NOORUL ISLAM COLLEGE OF ENGINEERING, KUMARACOIL

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

OBJECT ORIENTED PROGRAMMING

CLASS : THIRD SEMESTER CSE

Prepared By,

S.Vinila Jinny

Lecturer, Dept. of CSE

UNIT I

1. State the characteristics of procedure oriented programming.

- Emphasis is on algorithm.
- Large programs are divided into smaller programs called functions.
- Functions share global data.
- Data move openly around the system from function to function.
- Functions transform data from one form to another.
- Employs top-down approach in program design.

2. What are the features of Object Oriented Programming?

- Emphasis is on data rather than procedure.
- Programs are divided into objects.
- Data structures are designed such that they characterize the objects.
- Functions that operate on the data of an object are tied together.
- Data is hidden and cannot be accessed by external functions.
- Objects may communicate with each other through functions.
- New data and functions can easily be added whenever necessary.
- Follows bottom-up approach.

3. Distinguish between Procedure Oriented Programming and Object Oriented Programming.

Procedure Oriented Programming	**Object Oriented Programming**
• Emphasis is on algorithm.	• Emphasis is on data rather than procedure.
• Large programs are divided into smaller programs called functions.	• Programs are divided into objects.
• Functions share global data.	• Functions that operate on the data of an object are tied together.
• Data move openly around the system from function to function.	• Data is hidden and cannot be accessed by external functions.
• Employs top-down approach in program design.	• Follows bottom-up approach.

4. Define Object Oriented Programming (OOP).

Object Oriented Programming is an approach that provides a way of modularizing programs by creating partitioned memory area for both data and functions that can be used as templates for creating copies of such modules on demand.

5. List out the basic concepts of Object Oriented Programming.

- Objects
- Classes
- Data Abstraction and Encapsulation
- Inheritance
- Polymorphism
- Dynamic Binding
- Message Passing

6. Define Objects.

Objects are the basic run time entities in an object oriented system. They are instance of a class. They may represent a person, a place etc that a program has to handle. They may also represent user-defined data. They contain both data and code.

7. Define Class.

Class is a collection of objects of similar data types. Class is a user-defined data type. The entire set of data and code of an object can be made a user defined type through a class.

8. Define Encapsulation and Data Hiding.

The wrapping up of data and functions into a single unit is known as data encapsulation. Here the data is not accessible to the outside world.

The insulation of data from direct access by the program is called data hiding or information hiding.

9. Define Data Abstraction.

Abstraction refers to the act of representing the essential features without including the background details or explanations.

10. Define data members and member functions.

The attributes in the objects are known as data members because they hold the information. The functions that operate on these data are known as methods or member functions.

11. State Inheritance.

Inheritance is the process by which objects of one class acquire the properties of objects of another class. It supports the concept of hierarchical classification and provides the idea of reusability. The class which is inherited is known as the base or super class and class which is newly derived is known as the derived or sub class.

12. State Polymorphism.

Polymorphism is an important concept of OOPs. Polymorphism means one name, multiple forms. It is the ability of a function or operator to take more than one form at different instances.

13. List and define the two types of Polymorphism.

- Operator Overloading – The process of making an operator to exhibit different behaviors at different instances.
- Function Overloading – Using a single function name to perform different types of tasks. The same function name can be used to handle different number and different types of arguments.

14. State Dynamic Binding.

Binding refers to the linking of procedure call to the code to be executed in response to the call. Dynamic Binding or Late Binding means that the code associated with a given procedure call is known only at the run-time.

15. Define Message Passing.

Objects communicate between each other by sending and receiving information known as messages. A message to an object is a request for execution of a procedure. Message passing involves specifying the name of the object, the name of the function and the information to be sent.

16. List out some of the benefits of OOP.

- Eliminate redundant code
- Saves development time and leads to higher productivity
- Helps to build secure programs
- Easy to partition work
- Small programs can be easily upgraded to large programs
- Software complexity can easily be managed

17. Define Object Based Programming language.

Object Based Programming is the style of programming that primarily supports encapsulation and object identity. Languages that support programming with objects are known as Object Based Programming languages. They do not support inheritance and dynamic binding.

18. List out the applications of OOP.

- Real time systems
- Simulation and modeling

- Object oriented databases
- Hypertext, Hypermedia and expertext
- AI and expert systems
- Neural networks and parallel programming
- Decision support and office automation systems
- CIM/CAM/CAD systems

19. Define C++.

C++ is an object oriented programming language developed by Bjarne Stroustrup. It is a super set of C. Initially it was known as "C with Classes". It is a versatile language for handling large programs.

20. What are the input and output operators used in C++?

The identifier cin is used for input operation. The input operator used is >>, which is known as the extraction or get from operator. The syntax is,

```
cin >> n1;
```

The identifier cout is used for output operation. The input operator used is <<, which is known as the insertion or put to operator. The syntax is,

```
cout << "C++ is better than C";
```

21. What is the return type of main ()?

The return type of main () is integer i.e. main () returns an integer type value to the operating system. Therefore, every main () should end with a return (0) statement. It's general format is,

```
int main ()
{
        ...........
        return 0;
}
```

22. List out the four basic sections in a typical C++ program.

- Include files
- Class declaration
- Member functions definition
- Main function program

23. Define token. What are the tokens used in C++?

The smallest individual units in a program are known as tokens. The various tokens in C++ are keywords, identifiers, constants, strings and operators.

24. Define identifier. What are the rules to be followed for identifiers?

Identifiers refer to the names of variables, functions, arrays, classes etc created by the programmer. The rules are as follows:

- Only alphabetic characters, digits and underscores are permitted
- The name cannot start with a digit
- Uppercase and lowercase letters are distinct
- A declared keyword cannot be used as a variable name

25. State the use of void in C++.

The two normal uses of void are

- To specify the return type of the function when it is not returning a value
- To indicate an empty argument list to a function

26. Define an enumeration data type.

An enumerated data type is a user defined data type that provides a way for attaching names to numbers thereby increasing comprehensibility of the code. The enum keyword is used which automatically enumerates a list of words by assigning them values 0, 1, 2…

E.g.) enum shape {circle, square, triangle};

27. Define constant pointer and pointer to a constant.

The constant pointer concept does not allow us to modify the value initialized to the pointer.

e.g.) char * const ptr = "GOOD";

The pointer to a constant concept doesn't allow us to modify the address of the pointer.

E.g.) int const * ptr = &n;

28. What are the two ways of creating symbolic constants?

- Using the qualifier const
- Defining a set of integer constants using enum keyword

29. Define reference variable. Give its syntax.

A reference variable provides an alias or alternate name for a previously defined variable. It must be initialized at the time of declaration. Its syntax is given by,

data-type & reference-name = variable-name;

30. List out the new operators introduced in C++.

- **::** Scope resolution operator
- **::*** Pointer to member declarator
- **->*** Pointer to member operator
- **.*** Pointer to member operator
- **delete** Memory release operator
- **endl** Line feed operator
- **new** Memory allocation operator

- **setw** Field width operator

31. What is the use of scope resolution operator?

A variable declared in an inner block cannot be accessed outside the block. To resolve this problem the scope resolution operator is used. It can be used to uncover a hidden variable. This operator allows access to the global version of the variable. It takes the form,

:: variable-name

32. List out the memory differencing operator.

- **::*** To declare a pointer to the member of the class
- **->*** To access a member using object name and a pointer to that member
- **.*** To access a member using a pointer to the object and a pointer to that member

33. Define the 2 memory management operators.

- **new** Memory allocation operator
  The **new** operator can be used to create objects of any data-type. It allocates sufficient memory to hold a data object of type data-type and returns the address of the object.
  Its general form is,
  Pointer variable=**new** data-type;
- **delete** Memory release operator
  When a data object is no longer needed it is destroyed to release the memory space for reuse.
  The general form is,
  delete pointer variable;

34. List out the advantages of new operator over malloc ().

- It automatically computes the size of the data object.
- It automatically returns the correct pointer type.
- It is possible to initialize the objects while creating the memory space.
- It can be overloaded.

35. Define manipulators. What are the manipulators used in C++?

Manipulators are operators that are used to format the data display. The manipulators used in C++ are

- **endl** – causes a linefeed to be inserted
- **setw** – provides a common field width for all the numbers and forces them to be printed right justified

36. What are the three types of special assignment expressions?

• Chained assignment	e.g., x = y = 10;
• Embedded assignment	e.g., x = (y = 50) + 10;
• Compound assignment	e.g., x + = 10;

37. Define implicit conversion.

Whenever data types are mixed in a expression, C++ performs the conversions automatically. This process is known as implicit or automatic conversion.

e.g., m = 5 + 2.75;

38. Define integral widening conversion.

Whenever a char or short int appears in an expression, it is converted to an int. This is called integral widening conversion.

39. What are the control structures used in C++?

- Sequence structure (straight line)
- Selection structure (branching)
    - if – else (two way branch)
    - switch (multiple branch)
- Loop structure (iteration or repetition)
    - do – while (exit controlled)
    - while (entry controlled)
    - for (entry controlled)

40. Define Function Prototyping.

The function prototype describes the function interface to the compiler by giving details such as the number and type of arguments and type of return values. It is the declaration of a function in a program. It is in the following form,

type function – name (argument – list);

where argument – list -> types and names of arguments to be passed to the function

41. What is call by reference?

When we pass arguments by reference, the formal arguments in the called function become the aliases to the actual arguments in the calling function. Here the function works on the original data rather than its copy.

```
e.g., void swap (int &a, int &b)
      {
         int t = a;
         a = b;
         b = t;
      }
```

42. What are inline functions?

An inline function is a function that is expanded in line when it is invoked. Here, the compiler replaces the function call with the corresponding function code. The inline function is defined as,

```
inline function-header
{
        function body
}
```

43. List out the conditions where inline expansion doesn't work.

- For functions returning values, if a loop, a switch, or a goto exists
- For functions not returning values, if a return statement exists
- If functions contain static variables
- If inline functions are recursive

44. Why do we use default arguments?

The function assigns a default value to the parameter which does not have a matching argument in the function call. They are useful in situations where some arguments always have the same value.

e.g., float amt (float P, float n, float r = 0.15);

45. State the advantages of default arguments.

The advantages of default arguments are,

- We can use default arguments to add new parameters to the existing function.
- Default arguments can be used to combine similar functions into one.

46. Define function overloading.

A single function name can be used to perform different types of tasks. The same function name can be used to handle different number and different types of arguments. This is known as function overloading or function polymorphism.

47. List out the limitations of function overloading.

We should not overload unrelated functions and should reserve function overloading for functions that perform closely related operations.

UNIT II

1. State the difference between structures and class.

By default the members of a structure are public whereas the members of a class are private.

2. Define a class.

A class is a way to bind the data and its function together. It allows the data to be hidden from external use. The general form of a class is,

```
class class_name
{
        private:
                variable declarations;
                function declaration;
        public:
                variable declarations;
                function declaration;
};
```

3. List the access modes used within a class.

- Private – The class members are private by default. The members declared private are completely hidden from the outside world. They can be accessed from only within the class.
- Public – The class members declared public can be accessed from any where.
- Protected – The class members declared protected can be access from within the class and also by the friend classes.

4. How can we access the class members?

The class members can be accessed only when an object is created to that class. They are accessed with the help of the object name and a dot operator. They can be accessed using the general format,

```
Object_name.function_name (actual_arguments);
```

5. Where can we define member functions?

Member functions can be defined in two places:

- Outside the class definition – The member functions can be defined outside the class definition with the help of the scope resolution operator. The general format is given as,

```
return_type class_name :: function_name (argument declaration)
{
        function body
}
```

- Inside the class definition – The member function is written inside the class in place of the member declaration. They are treated as inline functions.

6. What are the characteristics of member functions?

The various characteristics of member functions are,

- Different classes can use the same function name and their scope can be resolved using the membership label.
- Member functions can access the private data of a class while a non-member function cannot.
- A member function can call another member function directly without using a dot operator.

7. How can an outside function be made inline?

An outside function can be made inline by just using the qualifier 'inline' in the header line of the function definition.

The general format is,

```
inline return_type class_name :: function_name (argument declaration)
{
        function body
}
```

8. What are the properties of a static data member?

The properties of a static data member are,

- It is initialized to zero when the first object is created and no other initialization is permitted.
- Only one copy of that member is created and shared by all the objects of that class.
- It is visible only within the class, but the life time is the entire program.

9. What are the properties of a static member function?

- A static member function can access only other static members declared in the same class.
- It can be called using the class name instead of objects as follows,

  class_name :: function_name;

10. How can objects be used as function arguments?

An object can be used as a function argument in two ways,

- A copy of the entire object is passed to the function. (Pass by value)
- Only the address of the object is transferred to the function. (Pass by reference)

11. Define friend function?

An outside function can be made a friend to a class using the qualifier 'friend'. The function declaration should be preceded by the keyword friend. A friend function has full access rights to the private members of a class.

12. List out the special characteristics of a friend function.

- It is not in the scope of a class in which it is declared as friend.
- It cannot be called using the object of that class.
- It can be invoked without an object.
- It cannot access the member names directly and uses the dot operator.
- It can be declared as either public or private.
- It has the objects as arguments.

13. Define Constructor.

A constructor is a special member function whose task is to initialize the objects of its class. It has the same name as the class. It gets invoked whenever an object is created to that class. It is called so since it constructs the values of data members of the class.

14. List some of the special characteristics of constructor.

- Constructors should be declared in the public section.
- They are invoked automatically when the objects are created.
- They do not have return types
- They cannot be inherited.

15. Give the various types of constructors.

There are four types of constructors. They are

- Default constructors – A constructor that accepts no parameters
- Parameterized constructors – The constructors that can take arguments
- Copy constructor – It takes a reference to an object of the same class as itself as an argument
- Dynamic constructors – Used to allocate memory while creating objects

16. What are the ways in which a constructor can be called?

The constructor can be called by two ways. They are,

- By calling the constructor explicitly

  e.g., integer int1 = integer (0, 100);
- By calling the constructor implicitly

  e.g., integer int1 (0, 100);

17. State dynamic initialization of objects.

Class objects can be initialized dynamically. The initial values of an object may be provided during run time. The advantage of dynamic initialization is that various initialization formats can be used. It provides flexibility of using different data formats.

18. Define Destructor.

A destructor is used to destroy the objects that have been created by a constructor. It is a special member function whose name is same as the class and is preceded by a tilde '~' symbol.

19. Give the general form of an operator function.

The general form of an operator function is given as,

```
return-type class-name :: operator op (arglist)
{
        function body
}
```

Where,

Return-type -> type of value returned
operator -> keyword
op -> operator being overloaded

20. List some of the rules for operator overloading.

- Only existing operators can be overloaded.
- We cannot change the basic meaning of an operator.
- The overloaded operator must have at least one operand.
- Overloaded operators follow the syntax rules of the original operators.

21. What are the types of type conversions?

There are three types of conversions. They are

- Conversion from basic type to class type – done using constructor
- Conversion from class type to basic type – done using a casting operator
- Conversion from one class type to another – done using constructor or casting operator

22. What are the conditions should a casting operator satisfy?

The conditions that a casting operator should satisfy are,

- It must be a class member.
- It must not specify a return type.
- It must not have any arguments.

UNIT III

1. What are the types of inheritance?

The various types of inheritance are,

- Single inheritance
- Multi-level inheritance
- Multiple inheritance
- Hierarchical inheritance
- Hybrid inheritance

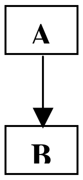
2. Give the syntax for inheritance.

The syntax of deriving a new class from an already existing class is given by,

```
class derived-class : visibility-mode base-class
{
        body of derived class
}
```

3. Define single inheritance.

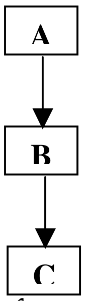
In single inheritance, one class is derived from an already existing base class.



Here A is the base class and B is the derived class.

4. Define multi-level inheritance.

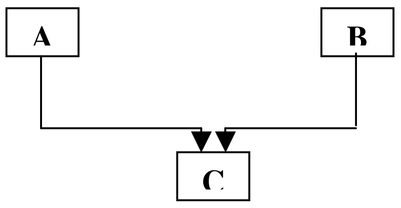
In multi-level inheritance, a new class is derived from a class already derived from the base class.



Here, class B is derived from class A and class C is further derived from the derived class B.

5. Define multiple inheritance.

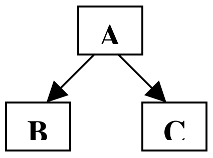
In multiple inheritance, a single class is derived from more than one base class.



Here class C is derived from two base classes A and B.

6. Define Hierarchical inheritance.

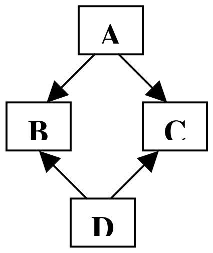
In hierarchical inheritance, more than one class is derived from a single base class.



Here class B and C are derived from class A.

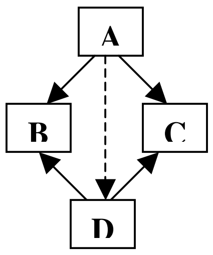
7. Define Hybrid inheritance.

Hybrid inheritance is defined as a combination of more than one inheritance.



Here, Classes A, B and C represent hierarchical inheritance. Classes A, B & D and classes A, C and D represent multilevel inheritance. Classes B, C and D represent multiple inheritance.

8. What is a virtual base class?



Here, class D inherits both classes B and C which are derived from the same base class A. Hence D has two copies of the properties of class A. This can be avoided by declaring classes B and C as virtual base classes.

9. What is an abstract class?

An abstract class is one that is not used to create objects. It is designed only to act as a base class to be inherited by other classes.

10. What are the types of polymorphism?

The two types of polymorphism are,

- Compile time polymorphism – The compiler selects the appropriate function for a particular call at the compile time itself. It can be achieved by function overloading and operator overloading.
- Run time Polymorphism - The compiler selects the appropriate function for a particular call at the run time only. It can be achieved using virtual functions.

11. Define 'this' pointer.

A 'this' pointer refers to an object that currently invokes a member function. For e.g., the function call a. show () will set the pointer 'this' to the address of the object 'a'.

12. What is a virtual function?

When a function is declared as virtual, C++ determines which function to use at run time based on the type of object pointed to by the base pointer, rather than the type of the pointer.

13. What is a pure virtual function?

A virtual function, equated to zero is called a pure virtual function. It is a function declared in a base class that has no definition relative to the base class.

14. How can a private member be made inheritable?

A private member can be made inheritable by declaring the data members as protected. When declared as protected the data members can be inherited by the friend classes.

UNIT IV

1. What are the various java features?

The various java features are,

- Compiled and interpreted
- Platform independent and portable
- Object oriented
- Robust and secure
- Distributed
- Familiar, small and simple
- Multithreaded and interactive
- High performance
- Dynamic and extensible

2. What are the two types of Java programs?

The two types of Java programs are,

- Stand alone applications
- Web applets

3. What are the steps involved in executing a stand alone java program?

The steps involved in executing a stand alone java program are,

- Compiling source code into byte code using javac compiler
- Executing the byte code program using java interpreter

4. What is the character set used in Java?

The character set used in Java is Unicode character set. It is a 16 – bit character coding system that supports more than 34,000 defined characters defined from more than 24 languages.

5. What is a token? What are the tokens used in java?

Smallest individual units in a program are known as tokens. The various tokens used in java are,

- Reserved keywords
- Identifiers
- Literals
- Operators
- Separators

6. What are the various java statements?

The various java statements used are,

- Empty statements
- Labeled statements

- Expression statements
- Selection statements
- Iteration statements
- Jump statement
- Synchronization statement
- Guarding statement

7. What are the steps involved in implementing a java program?

The steps involved in implementing a java program are,

- Creating the program
- Compiling the program
- Running the program

8. How does java achieve architecture neutrality?

The java compiler produces an intermediate code known as byte code for a machine that does not exist. This machine is called the java virtual machine and it exists only within the computer memory.

9. What is a constant? What are the types of constant used in java?

A constant is a fixed value that does not change during the execution of a program. The various constants used in java are,

- Integer constants
- Real constants
- Single character constants
- String constant
- Backslash character constant

10. List out the conditions a variable is subjected to.

The conditions a variable is subjected to are,

- They must begin with a digit
- Uppercase and lowercase are distinct
- It should not be a keyword
- White space is not allowed
- Variable names can be of any length

11. What are the kinds of java variables?

The three kinds of java variables are,

- Instance variables – They are created when the objects are instantiated and therefore they are associated with objects.
- Class variables – They are global to a class and belong to the entire set of the objects that class creates.
- Local variables – They are declared and used within methods.

12. Give the basic form of a class definition in java.

The basic form of a class definition is given by,

```
class class-name [extends superclassname]
{
        variable declaration;
        methods declaration;
}
```

13. What are the basic parts of a method declaration?

The basic parts of a method declaration are,

- The name of the method
- The return type of the method
- Parameter list
- Body of the method

14. What is inheritance? What are its types?

The mechanism of deriving a new class from an already existing class is known as inheritance. Its types are,

- Single inheritance
- Multi-level inheritance
- Hierarchical inheritance
- Hybrid inheritance

15. How does a subclass constructor invoke a super class constructor?

A subclass constructor invokes a super class constructor by using the super keyword. The super keyword is subjected to certain conditions,

- Super may bee used within a subclass constructor method
- It should be the first statement of the sub class method.
- It must match the order and type of the instance variables declared in the super class.

16. Define method overriding.

Method overriding is a mechanism in which the sub class method overrides the base class method. If the same function name is present in both the base class and the sub class then the sub class method overrides the base class method.

17. What are final variables, methods and classes?

In order to prevent the subclasses from overriding the members of a super class we can declare them as final using the keyword 'final'.

e.g, final int SIZE = 100;
final void show () {……..}
final class A {……}

18. What is the various visibility controls used in java?

The various visibility controls used in java are,

- Public access
- Private access
- Protected access
- Private protected access
- Friendly access

19. What are the rules of thumb?

The rules of thumb are,

- Use **public** if the field is to be visible everywhere.
- Use **protected** if the field is to be visible everywhere in the current package and also subclasses in the other packages.
- Use "default" if the field is to be visible everywhere in the current package only.
- Use **private protected** if the field is to be visible only in sub classes, regardless of packages.
- Use private if the field is not to be visible anywhere except in its own class.

20. What are the steps involved in creating an array?

Creation of an array involves three steps,

- Declare the array – two forms,
```
type array-name [];
type [] array-name;
```
- Create memory locations
```
array-name = new type [size];
```
- Put values into the memory locations
```
array-name [subscript] = value;
type array-name [] = { list of values};
```

UNIT V

1. What is an interface?

An interface is basically a kind of class. It contains abstract methods and final variable declarations. It does not specify any code to implement these methods and data fields contain only constants. It is used to implement multiple inheritance.

2. How do you implement an interface?

An interface can be implemented as follows,

```
class class-name implements interface-name
{
        variable declaration;
        methods declaration;
}
```

3. What are the steps involved in creating our own package?

The steps involved in creating our own package are,

- Declare the package at the beginning of the file.
- Define the class and declare it as public.
- Create a sub directory.
- Store the listing
- Compile the file

4. Define thread.

A thread is similar to a program that has a single flow of control. It is a tiny program or module that runs in parallel with others.

5. Define Multitasking or multithreading.

Multitasking or multithreading is the ability to execute several programs simultaneously. A program that contains multiple flows of control is known as a multithreaded program.

6. What are the ways to create a thread?

A thread can be created in two ways,

- By creating a thread class – extends Thread class
- By converting a class to a thread – implements Runnable interface

7. How do you stop and block a thread?

A thread can be stopped using the stop (). It can be blocked using sleep (), suspend () and wait () methods.

8. What are the states in the life cycle of a thread?

The various states are,

- Newborn
- Runnable
- Running
- Blocked
- Dead

9. Define synchronization?

Synchronization is the process by which only one class is allowed to access a member function at a time. It is done using the qualifier 'synchronized'.

e.g., synchronized void show () {……..}

10. Define error. What are its types?

An error is a wrong that can make a program go wrong. There are two types of errors namely,

- Compile time errors
- Run time errors

11. What is an exception? What are the steps involved in exception handling?

An exception is a condition that is caused by a run time error in a program. The steps involved in exception handling are,

- Hit the exception
- Throw the exception
- Catch the exception
- Handle the exception

12. Give the syntax of exception handling code.

The syntax of exception handling code is given by,

```
try
{
statements
}
catch ( Exception-type e)
{
statements
}
```

13. What is an applet? What are its types?

An applet is a small java program that is primarily used in internet computing. The types of applets are,

- Local applets
- Remote applets

14. What are the various states in an applet life cycle?

The various states in an applet life cycle are,

- Born or initialization state
- Running state
- Display state
- Idle state
- Dead or destroyed state

15. How do you create an executable applet?

- Move to the directory containing the source code and type the following command, javac classname.java
- The compiled output file is placed in the same directory as the source.
- If any error message is received, then we must check for it , correct them and compile it again.

Two Mark Questions

1) Give the evolution diagram of OOPS concept.

Machine language
˘
Procedure language
˘
Assembly language
˘
OOPS

2) Draw the structure of Procedure oriented language or Typical organization of Procedure oriented language.

3) What is Procedure oriented language?

Conventional programming, using high-level language such as COBOL, FORTRAN and C are commonly known as Procedure oriented language (POP). In POP number of functions are written to accomplish the tasks such as reading, calculating and printing.

4) Give some characteristics of procedure-oriented language.

❖ Emphasis is on doing things (algorithms).
❖ Larger programs are divided into smaller programs known as functions.
❖ Most of the functions share global data.
❖ Data move openly around the system from function to function.
❖ Employs *top-down* approach in program design.

Function-1 Function-2 Function-3
Function-4 Function-5
Function-6 Function-7 Function-8
Main program

5) Write any four features of OOPS.

❖ Emphasis is on data rather than on procedure.
❖ Programs are divided into objects.

- ❖ Data is hidden and cannot be accessed by external functions.
- ❖ Follows **bottom**-*up* approach in program design.

6) What are the basic concepts of OOS?

- ❖ Objects.
- ❖ Classes.
- ❖ Data abstraction and Encapsulation.
- ❖ Inheritance.
- ❖ Polymorphism.
- ❖ Dynamic binding.
- ❖ Message passing.

7) What are objects?

Objects are basic run-time entities in an object-oriented system. They may represent a person, a place, a bank account, a table of data or any item that the program has to handle. Each object has the data and code to manipulate the data and theses objects interact with each other.

8)What is a class?

The entire set of data and code of an object can be made a user-defined data type with the help of a class. Once a class has been defined, we can create any number of objects belonging to the classes.

Classes are user-defined data types and behave like built-in types of the programming language.

9) what is encapsulation?

Wrapping up of data and function within the structure is called as encapsulation.

10)What is data abstraction?

The insulation of data from direct access by the program is called as data hiding or information binding.

The data is not accessible to the outside world and only those functions, which are wrapped in the class, can access it.

11)What are data members and member functions?

Classes use the concept of abstraction and are defined as a list of abstract attributes such as size, weight, and cost and uses functions to operate on these attributes.

The attributes are sometimes called as data members because they hold information. The functions that operate on these data are called as methods or member functions.

Eg: int a,b; // a,b are data members

Void getdata ( ) ; // member function

12)What is dynamic binding or late binding?

Binding refers to the linking of a procedure to the code to be executed in response to the call. Dynamic binding means that the code associated with a given procedure call is not known until the time of the call at the run-time.

13)Write the process of programming in an object-oriented language?

- ❖ Create classes that define objects and their behavior.
- ❖ Creating objects from class definition.
- ❖ Establishing communication among objects.

14)Give any four advantages of OOPS.

- ❖ The principle of data hiding helps the programmer to build secure programs that cannot be invaded by code in other parts of the program.
- ❖ It is possible to have multiple instances of an object to co-exist without any interference.
- ❖ Object oriented programming can be easily upgraded from small to large systems.
- ❖ Software complexity can be easily managed.

15)What are the features required for object-based programming Language?

- ➢ Data encapsulation.
- ➢ Data hiding and access mechanisms.
- ➢ Automatic initialization and clear up of objects.
- ➢ Operator overloading.

16)What are the features required for object oriented language?

- ➢ Data encapsulation.
- ➢ Data hiding and access mechanisms.
- ➢ Automatic initialization and clear up of objects.
- ➢ Operator overloading.
- ➢ Inheritance.
- ➢ Dynamic binding.

17)Give any four applications of OOPS

- Real-time systems.
- Simulation and modeling.
- Object-oriented databases.
- AI and expert systems.

18) Give any four applications of c++?

- ➢ Since c++ allows us to create hierarchy-related objects, we can build special object-oriented libraries, which can be used later by many programmers.
- ➢ C++ are easily maintainable and expandable.
- ➢ C part of C++ gives the language the ability to get close to the machine-level details.
- ➢ It is expected that C++ will replace C as a general-purpose language in the near future.

19) What are tokens?

The smallest individual units in a program are known as tokens. C++ has the following tokens,

- o Keyword
- o Identifiers
- o Constants
- o Strings
- o Operator

20)What are keywords?

The keywords implement specific C++ language features. They are explicitly reserved identifiers and cannot be used as names fro the program variables or other user defined program elements.

Eg: go to, If, struct , else ,union etc.

21) Rules for naming the identifiers in C++.

- ❖ Only alphabetic characters, digits and underscore are permitted.
- ❖ The name cannot start with a digit.
- ❖ The upper case and lower case letters are distinct.
- ❖ A declared keyword cannot be used as a variable name.

22)What are the operators available in C++?

All operators in C are also used in C++. In addition to insertion operator << and extraction operator >> the other new operators in C++ are,

: Scope resolution operator
: : * Pointer-to-member declarator
->* Pointer-to-member operator
.* Pointer-to-member operator
delete Memory release operator
endl Line feed operator
new Memory allocation operator
setw Field width operator

23)What is a scope resolution operator?

Scope resolution operator is used to uncover the hidden variables. It also allows access to global version of variables.

Eg:

```
#include<iostream.h>
int m=10; // global variable m
void main ( )
{
int m=20; // local variable m
cout<<"m="<<m<<"\n";
cout<<": : m="<<: : m<<"\n";
}
```

output:
20
10 (: : m access global m)

Scope resolution operator is used to define the function outside the class.

Syntax:
Return type <class name> : : <function name>

Eg:
Void x : : getdata()

24) What are free store operators (or) Memory management operators?

New and Delete operators are called as free store operators since they allocate the memory dynamically.

New operator can be used to create objects of any data type.

Pointer-variable = new data type;

Initialization of the memory using new operator can be done. This can be done as,

Pointer-variable = new data-type(value)

Delete operator is used to release the memory space for reuse. The general form of its use is

Delete pointer-variable;

25) What are manipulators?

Setw, endl are known as manipulators.
Manipulators are operators that are used to format the display. The endl manipulator when used in an output statement causes a linefeed to be inserted and its effect is similar to that of the newline character"\n".

Eg:

Cout<<setw(5)<<sum<<endl;

26) What do you mean by enumerated datatype?

An enumerated datatype is another user-defined datatype, which provides a way for attaching names to numbers, thereby increasing comprehensibility of the code.
The syntax of an **enum** statement is similar to that of the struct statesmen.

Eg:

enum shape{ circle, square, triangle}
enum color{ red, blue, green, yellow}

27) What are symbolic constants?

There are two ways for creating symbolic constants in C++:

- Using the qualifier **constant.**
- Defining a set of integer constants using **enum** keyword.

The program in any way cannot modify the value declared as constant in c++.
Eg:
Const int size =10;
Char name [size];

28)What do you mean by dynamic initialization of variables?

C++ permits initialization of the variables at run-time. This is referred to as dynamic initialization of variables.
In C++ ,a variable can be initialized at run-time using expressions at the place of declaration as,

```
……..
…......
int n =strlen(string);
……..
float area=3.14*rad*rad;
```

Thus declaration and initialization is done simultaneously at the place where the variable is used for the first time.

29) What are reference variable?

A reference variable provides an alias(alternative name) for a previously defined variable.
For example , if make the variable **sum** a reference to the variable **total**, then **sum** and **total** can be used interchancheably to represent that variable.

Syntax:

Data-type &reference-name = variable-name

Eg:

float total = 100;
float sum = total;

30)What is member-dereferencing operator?

C++ permits to access the class members through pointers. It provides three pointer-to-member operators for this purpose,

: :* To declare a pointer to a member of a class.
* To access a member using object name and a pointer to the member
->* To access a member using a pointer to the object and a pointer to that member.

31)what is function prototype ?

The function prototype describes function interface to the compiler by giving details such as number ,type of arguments and type of return values
Function prototype is a declaration statement in the calling program and is of the following

Type function_name(argument list);

Eg float volume(int x,float y);

32)what is an inline function ?

An inline function is a function that is expanded in line when it is invoked. That is compiler replaces the function call with the corresponding function code.
The inline functions are defined as

Inline function-header
{
function body
}

33) Write some situations where inline expansion may not work

- ❖ for functions returning values, if loop, a **switch,** or a **goto** exists
- ❖ for functions not returning values ,if a return statement exists
- ❖ if function contain **static** variables
- ❖ if **inline** functions are recursive

34)what is a default argument ?

Default arguments assign a default value to the parameter, which does not have matching argument in the function call. Default values are specified when the function is declared.
Eg : float amount(float principle,int period,float rate=0.15)
Function call is
Value=amount(5000,7);
Here it takes principle=5000& period=7
And default value for rate=0.15
Value=amount(5000,7,0.34)
Passes an explicit value 0f 0.34 to rate
We must add default value from right to left

35) What are constant arguments ?

keyword is const. The qualifier const tells the compiler that the function should not modify the argument. The compiler will generate an error when this condition is violated. This type of declaration is significant only when we pass arguments by reference or pointers
eg: int strlen(**const** char *p);

36) How the class is specified ?

Generally class specification has two parts

- ❖ class declaration

It describes the type and scope of its member

- ❖ class function definition

It describes how the class functions are implemented

The general form is

```
Class class_name
{
private:
variable declarations;
function declaration;
public:
variable declaration;
function declaration;
};
```

37) How to create an object ?

Once the class has been declared, we can create variables of that type by using the classname

Eg:classname x; //memory for x is created

38) How to access a class member ?

object-name. function-name(actual arguments)

eg:x.getdata(100,75.5);

39) How the member functions are defined ?

Member functions can be defined in two ways

- ❖ outside the class definition

Member function can be defined by using scope resolution operator::

General format is

```
Return type class_name::function-name(argument declaration)
{
}
```

- ❖ Inside the class definition

This method of defining member function is to replace the function
declaration by the actual function definition inside the class. It is treated as inline function

```
Eg:class item
{
int a,b ;
void getdata(int x,int y)
{
a=x;
b=y;
};
```

40) What is static data member?

Static variable are normally used to maintain values common to the entire class.

Feature:

- ❖ It is initialized to zero when the first object is created. No other initialization is permitted
- ❖ only one copy of that member is created for the entire class and is shared by all the objects
- ❖ It is only visible within the class, but its life time is the entire class

- type and scope of each static member variable must be defined outside the class
- It is stored separately rather than objects

Eg: static int count//count is initialized to zero when an object is created.
int classname::count;//definition of static data member

41) What is static member function?

A member function that is declared as static has the following properties

- A static function can have access to only other static member declared in the same class
- A static member function can be called using the classname as follows classname ::function_name;

42) How the objects are used as function argument?

This can be done in two ways

- A copy of the entire object is passed to the argument
- Only address of the objects is transferred to the function

43) What is called pass by reference?

In this method address of an object is passed, the called function works directly on the actual arguments.

44) Define const member

If a member function does not alter any data in the class, then we may declare it as const member function as
Void mul(int ,int)const;

45) Define pointers to member

It is possible to take the address of a member of a class and assign it to a pointer. The address of a member can be obtained by applying the operator &to a “fully qualified” class member name. A class member pointer can be declared using the operator::*with the class name.

Eg: class A
{
int m;
public:
void show( );
};
pointer to member m is defined as
int A::*ip=&A::m;
A::*->pointer to member of A class
&A::m->address of the m member of A class

46) When the deferencing operator ->* is used?

It is used to access a member when we use pointer to both the object and the member.

47) When the deferencing operator .* is used?

It is used to access a member when the object itself is used as pointers.

48) Define local classes.

Classes can be defined and used inside a function or a block. such classes are called local classes. It can use global variables and static variables declared inside the function but cannot use automatic local variables.

Eg;

```
void test(int a)
{
…….
}
class student
{
………
};
student s1(a);
}
```

49) Define constructor

A constructor is a special member function whose task is to initialize the objects of its class. It is special because its name is same as class name. The constructor is invoked whenever an object of its associated class is created. It is called constructor because it constructs the values of data members of the class

Eg:

```
Class integer
{
……
public:
integer( );//constructor
………
}
```

50) Define default constructor

The constructor with no arguments is called default constructor

Eg:

```
Class integer
{
int m,n;
Public:
Integer( );
…….
};
integer::integer( )//default constructor
{
m=0;n=0;
}
```

the statement

integer a;

invokes the default constructor

51) Define parameterized constructor

constructor with arguments is called parameterized constructor

Eg;

```
Class integer
{ int m,n;
public:
```

integer(int x,int y)
{ m=x;n=y;
}
To invoke parameterized constructor we must pass the initial values as arguments to the constructor function when an object is declared. This is done in two ways
1.By calling the constructor explicitly
eg:
integer int1=integer(10,10);
2.By calling the constructor implicitly
eg:
Integer int1(10,10);
52) Define default argument constructor
The constructor with default arguments are called default argument constructor
Eg:
Complex(float real,float imag=0);
The default value of the argument imag is 0
The statement
complex a(6.0)
assign real=6.0 and imag=0
the statement
complex a(2.3,9.0)
assign real=2.3 and imag=9.0
53) What is the ambiguity between default constructor and default argument constructor ?
The default argument constructor can be called with either one argument or no arguments. when called with no arguments ,it becomes a default constructor. When both these forms are used in a class ,it cause ambiguity for a statement such as A a;
The ambiguity is whether to call A::A() or A::A(int i=0)
54) Define copy constructor
A copy constructor is used to declare and initialize an object from another object. It takes a reference to an object of the same class as an argument
Eg: integer i2(i1);
would define the object i2 at the same time initialize it to the values of i1.
Another form of this statement is
Eg: integer i2=i1;
The process of initializing through a copy constructor is known as **copy initialization**.
55) Define dynamic constructor
Allocation of memory to objects at time of their construction is known as dynamic constructor. The memory is allocated with the help of the **NEW operator**
Eg:
Class string
{
char *name;
int length;
public:
string( )

```
{
length=0;
name=new char[length +1];
}
void main( )
{
string name1(“Louis”),name3(Lagrange);
}
```
56) Define const object
We can create constant object by using const keyword before object declaration.
Eg: Const matrix x(m,n);
57) Define destructor
It is used to destroy the objects that have been created by constructor. Destructor name is same as class name preceded by tilde symbol(~)
Eg;
```
~integer()
{
}
```
A destructor never takes any arguments nor it does it return any value. The compiler upon exit from the program will invoke it.
Whenever **new** operator is used to allocate memory in the constructor, we should use delete to free that memory.
58) Define multiple constructors (constructor overloading).
The class that has different types of constructor is called multiple constructors
Eg:
```
#include<iostream.h>
#include<conio.h>
class integer
{
int m,n;
public:
integer( ) //default constructor
{
m=0;n=0;
}
integer(int a,int b) //parameterized constructor
{
m=a; n=b;
}
integer(&i) //copy constructor
{
m=i.m;
n=i.n;
}
void main()
```

```
{
integer i1; //invokes default constructor
integer i2(45,67);//invokes parameterized constructor
integer i3(i2); //invokes copy constructor
}
```

59) Write some special characteristics of constructor

- They should be declared in the public section
- They are invoked automatically when the objects are created
- They do not have return types, not even void and therefore, and they cannot return values
- They cannot be inherited, though a derived class can call the base class
- They can have default arguments
- Constructors cannot be virtual function

60) How the objects are initialized dynamically?

To call parameterized constructor we should the pass values to the object
ie,for the constructor integer(int a,int b)
it is invoked by integer a(10,18)
this value can be get during run time. i.e., for above constructor

```
int p,q;
cin>>p>>q;
integer a(p,q);
```

61)Define Inline Function?

Inline function is defined as a function definition such that each call to the function is in effect, replaced by the statements that define the function. It is expanded in line when it is invoked. The general form is

```
inline function-header
{
function body
}
```

62)Explain return by reference with an example.

A function can also return a reference. Consider the following function

```
int & max( int &x , int &y)
{ if(x>y)
return x;
else
return y;
}
```

Since the return type of max ( ) is int & the function returns reference to x or y (and not the values). Then a function call such as max (a , b) will yield a reference to either a or b depending on their values.
The statement
max (a , b) = -1;
is legal and assigns –1 to a if it is larger, otherwise –1 to b.

63) What are Friend functions? Write the syntax

A function that has access to the private member of the class but is not itself a member of the class is called friend functions.

The general form is
friend data_type function_name( );
Friend function is preceded by the keyword 'friend'.

64)Write some properties of friend functions.

- ❑ Friend function is not in the scope of the class to which it has been declared as friend. Hence it cannot be called using the object of that class.
- ❑ Usually it has object as arguments.
- ❑ It can be declared either in the public or private part of a class.
- ❑ It cannot access member names directly. It has to use an object name and dot membership operator with each member name. eg: (A . x)

65)What are virtual functions?

A function qualified by the 'virtual' keyword is called virtual function. When a virtual function is called through a pointer, class of the object pointed to determine which function definition will be used.

66)Write some of the basic rules for virtual functions

- ▪ Virtual functions must be member of some class.
- ▪ They cannot be static members and they are accessed by using object pointers
- ▪ Virtual function in a base class must be defined.
- ▪ Prototypes of base class version of a virtual function and all the derived class versions must be identical.
- ▪ If a virtual function is defined in the base class, it need not be redefined in the derived class.

67) What are pure virtual functions? Write the syntax.

A pure virtual function is a function declared in a base class that has no definition relative to the base class. In such cases, the compiler requires each derived class to either define the function or redeclare it as a pure virtual function. A class containing pure virtual functions cannot be used to declare any object of its own. It is also known as "donothing"
function.
The "do-nothing" function is defined as follows:
virtual void display ( ) =0;

68) What is polymorphism? What are its types?

Polymorphism is the ability to take more than one form. An operation may exhibit different behaviors in different. The behavior depends upon the type of data used.
Polymorphism is of two types. They are

- ❖ Function overloading
- ❖ Operator overloading

69) What is function overloading? Give an example.

Function overloading means we can use the same function name to create functions that perform a variety of different tasks.
Eg: An overloaded add ( ) function handles different data types as shown below.
// Declarations
i. int add(int a, int b); //add function with 2 arguments of same type
ii. int add(int a, int b, int c); //add function with 3 arguments of same type
iii. double add(int p, double q); //add function with 2 arguments of different type

//Function calls
add (3 , 4); //uses prototype (i.)
add (3, 4, 5); //uses prototype (ii.)
add (3 , 10.0); //uses prototype (iii.)

70) What is operator overloading?

C++ has the ability to provide the operators with a special meaning for a data type. This mechanism of giving such special meanings to an operator is known as Operator overloading. It provides a flexible option for the creation of new definitions for C++ operators.

71) List out the operators that cannot be overloaded.

- ❖ Class member access operator (. , .*)
- ❖ Scope resolution operator (::)
- ❖ Size operator (sizeof)
- ❖ Conditional operator (?:)

72) What is the purpose of using operator function? Write its syntax.

To define an additional task to an operator, we must specify what it means in relation to the class to which the operator is applied. This is done by Operator function , which describes the task. Operator functions are either member functions or friend functions. The general form is

```
return type classname :: operator (op-arglist )
{
function body
}
```

where *return type* is the type of value returned by specified operation.
Op-operator being overloaded. The *op* is preceded by a keyword operator. operator op is the function name.

73) Write at least four rules for Operator overloading.

- ➢ Only the existing operators can be overloaded.
- ➢ The overloaded operator must have at least one operand that is of user defined data type.
- ➢ The basic meaning of the operator should not be changed.
- ➢ Overloaded operators follow the syntax rules of the original operators. They cannot be overridden.

74) How will you overload Unary & Binary operator using member functions?

When unary operators are overloaded using member functions it takes no explicit arguments and return no explicit values.
When binary operators are overloaded using member functions, it takes one explicit argument. Also the left hand side operand must be an object of the relevant class.

75) How will you overload Unary and Binary operator using Friend functions?

When unary operators are overloaded using friend function, it takes one reference argument (object of the relevant class)
When binary operators are overloaded using friend function, it takes two explicit arguments.

76) How an overloaded operator can be invoked using member functions?

In case of Unary operators, overloaded operator can be invoked as
op object_name or object_name op

In case of binary operators, it would be invoked as
Object . operator op(y)
where op is the overloaded operator and y is the argument.

77) How an overloaded operator can be invoked using Friend functions?

In case of unary operators, overloaded operator can be invoked as
Operator op (x);
In case of binary operators, overloaded operator can be invoked as
Operator op (x , y)

78) List out the operators that cannot be overloaded using Friend function.

- ❖ Assignment operator =
- ❖ Function call operator ( )
- ❖ Subscripting operator [ ]
- ❖ Class member access operator →

79) What is meant by casting operator and write the general form of overloaded casting operator.

A casting operator is a function that satisfies the following conditions

- ➢ It must be a class member.
- ➢ It must not specify a return type.
- ➢ It must not have any arguments.

The general form of overloaded casting operator is

```
operator type name ( )
{
……….. // function statements
}
```

It is also known as conversion function.

80) Explain basic to class type conversion with an example.

Conversion from basic data type to class type can be done in destination class. Using constructors does it. Constructor takes a single argument whose type is to be converted.

Eg: Converting int type to class type

```
class time
{
int hrs,mins;
public:
…………
Time ( int t) //constructor
{
hours= t/60 ; //t in minutes
mins =t % 60;
}
};
```

Constructor will be called automatically while creating objects so that this conversion is done automatically.

81) Explain class to basic type conversion with an example.

Using Type Casting operator, conversion from class to basic type conversion can be done. It is done in the source class itself.

Eg: vector : : operator double( )
{
double sum=0;
for(int I=0;I<size;I++)
sum=sum+v[i] *u[i] ;
return sqrt (sum) ;
}
This function converts a vector to the corresponding scalar magnitude.
82) Explain one class to another class conversion with an example.
Conversion from one class type to another is the combination of class to basic and basic to class type conversion. Here constructor is used in destination class and casting operator function is used in source class.
Eg: objX = objY
objX is the object of class X and objY is an object of class Y. The class Y type data is converted into class X type data and the converted value is assigned to the obj X. Here class Y is the source class and class X is the destination class.
83) What is meant by inheritance?
Inheritance is the process by which objects of one class acquire the properties of another class. It supports the concept of hierarchical classification. It provides the idea of reusability. We can add additional features to an existing class without modifying it by deriving a new class from it.
84) What is meant by single inheritance?
If a single class is derived from a single base class is called single inheritance.
Eg:
Base class
Derived class
Here class A is the base class from which the class D is derived. Class D is the public derivation of class B hence it inherits all the public members of B. But D cannot access private members of B.
85) What is multiple inheritance?
If a class is derived from more than one base class, it is called multiple inheritance.
Eg: Base classes
Derived class
Here class C is derived from two base classes A & B.
86) What is hierarchical inheritance?
If a number of classes are derived from a single base class then it is called hierarchical inheritance.
Eg : Hierarchical classification of students in University
A
B
A
C
B
87) What is multilevel inheritance?
If a class is derived from a class, which in turn is derived from another class, is

called multilevel inheritance. This process can be extended to any number of levels.
Eg:
Base class Grand father
Intermediate
Base class Father
Derived class Child

88) What is hybrid inheritance?

It is the combination of one or more types of inheritance.
Multilevel
inheritance
Multiple
inheritance
The class result will have both the multilevel and multiple inheritances.
Student
Arts Engineering M e d i c a l
CSE ECE Civil
A
B
C
Student
Test
Result
Sports

89) What is meant by Abstract base class?

A class that serves only as a base class from which derived classes are derived. No objects of an abstract base class are created. A base class that contains pure virtual function is an abstract base class.

90) Write short notes on virtual base class.

A base class that is qualified as virtual in the inheritance definition. In case of multiple inheritance, if the base class is not virtual the derived class will inherit more than one copy of members of the base class. For a virtual base class only one copy of members will be inherited regardless of number of inheritance paths between base class and derived class.
Eg: Processing of students' results. Assume that class sports derive the roll number from class student. Class test is derived from class Student. Class result is derived from class Test and sports.
As a virtual base class As a virtual base class

91)Define Polymorphism?

Polymorphism is the feature that allows one interface to be used for a general class of actions.(ie) "one interface multiple methods".
This means that it is possible to design a generic interface to a group of related activites.This helps reduce complexity by allowing the same interface to be used to specify a general class of action.

92)Mention some of the Separators used in Java Programming?

( ) → Contain a list of parameters in method definition & invocation.
{ } → Contain the value of automatically initialized arrays.

[ ] → Declare array types.
; → Terminate statements.
. → Separate package name from sub packages.

93)What is boolean data type?

Java has simple type called boolean for logical values. It can have only one of
two possible values, true or false. This is the type returned by all relational operators like
a<b.boolean is also required by the conditional expression that governs the control
statements such as if for.

Student
Test
Result
Sports

Syntax: boolean variablename;

94)How dynamic initialization of variables is achieved in java?

Java allows variables to be initialized dynamically, using any expression valid at
the time the variable is declared.
double a= 3.0,b=4.0
double c=Math.sqrt(a * a + b * b);
here "c" is initialized dynamically to the length of hypotenuse.

95)What is meant by Widening conversion?

When one type of data is assigned to another type of variable ,an automatic
conversion will take place if the following conditions are met .
The two types are compatible.
The destination type is larger than the source type.
For example the int type is always large enough to hold to hold all byte values

96)What is meant by narrowing conversion?

We are explicitly making the value narrower so that it will fit into the target type.
The conversion is not performed automatically. To create a conversion between
two incompatible types, you must use a cast. A cast is simply an explicit type conversion.
Syntax: (target-type)value
target-type → the desired type to convert the specified value to.

97)State Type Promotion Rules?

All byte & short values are promoted to int. if one operand is long ,the whole
expression is promoted to long. If one operand is a float operand, the entire expression is
promoted to float. If any one operand is double, the result is double.
byte, short→ long→ float → double

98)How to create a one dimensional arrays?

A one dimensional array is a list of liked type variables. To create an array ,first
create an array variable of desired data type.
Syntax: type var-name[];

99)Here var-name is set to null. To link with an actual, physical array of integers, allocate using new.

bb: array-var=new type[size];

100)What is the use of ternary operator?

The ternary operator replaces if-then-else statements.
Syntax: expression1?expression2:expression3

Eg: ratio = denom = = 0 ? : num / denom;
If expression1 is true ,then expression2 is evaluated; otherwise expression3 is evaluated.
The result of ? operation is that of the expression evaluated.

101)Write down the syntax of switch statement?

```
Switch(expression){
case value1:
//statement sequence
break;
case value2:
//statement sequence
break;
.
.
.
case valueN:
//statement sequence
break;
default:
//default statement sequence
}
```

102)What are the iteration statements used in Java?

While: repeats a statement or block while its controlling expression is true.

```
Syntax: while(condition){
//body of loop
}
```

do-while: Executes its body atleast once

```
Syntax: do{
//body of loop
}while(condition);
```

for: consists of three portions initialization,conditon,termination.

```
Syntax: for(initialization,conditon,termination.){
//body
}
```

103)What is the difference between break & continue statements?

Break: We can force immediate termination of a loop, bypassing the conditional, the loop expression & any remaining code in the body of the loop. When a break statement is encountered in a loop, the loop is terminated & the program control resumes at the next statement following the loop.

Continue: useful to force early termination. it continue running the loop, but stop processing the remainder of the code in it's body for this particular iteration

104)What are the uses of break statements?

1.It terminates a statement sequence in switch statement.
2.It can be used to exit a loop.
3.it can be used as a civilized form of goto.

105)Define Constructors?

A constructor initializes an object immediately upon creation. It has the same names the class in which it resides & is syntactically similar to a method. Once defined the constructor is automatically called immediately after the object is created, before the new operator completes.

Syntax: class-var = new classname( );

106)Define parameterized Constructors?

To construct an object of various dimensions we can add parameters to the constructor.

Eg: Box mybox1= new Box(10,20,15);

Box mybox1= new Box(7,15);

107)What is the use of This keyword?

This is always a reference to the object on which the method was invoked. this can be used inside any method to refer to the current object.

```
Box(double w,double h,double d){
This. width = w;
This. height= h;
This. depth = d;
}
```

108)Define Garbage collection?

The technique used to handle the deal location automatically. When no references to an object exists, that object is assumed to be no longer needed,& the memory occupied by the object can be reclaimed. Garbage collection occurs sporadically during the execution of your program.

109)What is the use of finalize() method?

If an object is holding some non-java resource such as a file handle, or a window character font which might be freed before an object is destroyed. To handle such situation Java provides a mechanism called finalization. By using finalization we can define specific action that will occur when an objects just about to reclaimed by the Garbage collector.

Syntax: protected void finalize()

```
{
//finalization code
}
```

110)Define Method overloading?

In Java it is possible to define two or methods with the same class that share the same name, as long as the parameter declarations are different. When this is the case the methods are said to be overloaded &the process is referred to as method overloading.

```
void test(int a){
System.out.println("a: " +a);
}
void test(double a){
System.out.println("a: " +a);
}
void test(int a,int b){
System.out.println("a and b: " +a +" " +b);
}
```

111)What are the different ways of argument Passing?

Call-by-value: This method copies the value of an argument in to the formal parameter of the subroutine. Therefore changes made to the parameter of the subroutine have no effect on the argument used to call it.
Call-by-Reference: In this method ,a reference to an argument is passed to the parameter. Inside this subroutine ,this reference is used to access the actual argument specified in the call. This means that changes made to the parameter will affect the argument used to call the subroutine.
Define Recursion?
Recursion is the process of defining something in terms of itself.Recrsion is the attribute that allows a method to call itself, A method that call itself is said to be recursive.

```
int fact(int n){
int result;
if (n= =1) return 1;
result= fact(n-1) * n;
return result;
}
```

112)Mention some of the restrictions while using static keyword?
They an call other static methods.
They must only access the static data.
They cannot refer to **this** or **super** any way.
113)What is the use of final keyword?
It is used to prevent its contents from being modified. It is similar to const in C/C++.We must initialize a final variable when it is declared.
Eg: **final** int FILE_NEW=1;
114)How to call a Super class constructors?
A subclass can call a constructor method defined by its super class by use of the following form of **super.**
Super(parameter-list);
Parameter list→any parameters needed by the constructor in the superclass.super() must always be the first statement executed inside a subclass's constructor.
115)What are the uses of super keyword?
1.Using super we can call Super class constructor.
2.it acts like this, except that it always refers to the super class of the subclass in which it is used.
116)Define Method overriding?
When a method in a subclass has the same name & type signature as a method in the subclass is said to override the method in the superclass.when an overridden method is called from within a subclass it will always refer to the version of that method defined by the subclass.

```
class A{
…….
Void show(){
System.out.println("Super class")
}
class B extends A{
```

```
…….
Void show(){
System.out.println("sub class")
}
class override{
B sub = new B( );
Sub. show( );
}
```

117)Define Dynamic Method Dispatch?
Dynamic Method Dispatch is the mechanism by which a call to an overridden function is resolved at run time, rather than compile time. Java implements run time Polymorphism by means of Dynamic Method Dispatch.
Principle: A super class reference variable can refer to a sub class object. java uses this to resolve calls to overridden methods at run time. when ah overridden method is called through a super class reference, java determines which version of that method to execute based up on the type of the object being referred to at the time the call occurs. It is the type of the object being referred to that determines which version of an overridden method will be executed.
118)What are the uses of final keyword?
1. to create the equivalent of named constant.
2. To Prevent overriding
3. To prevent inheritance.
119)How to define a Package?
Include package statement as the first statement in a java sourcefile.The package statement defines a name space in which classes are stored.
Syntax: package pkg;
Pkg → name of the package.
We can create a hierarchy of classes. For that separate each package name from the one above it by use of a period.
Syntax: package pkg1[pkg2[.pkg3]];
120)How to import packages?
Java includes the import statements to bring certain classes ,or entire packages in to visibility. import statements occur immediately following the package statements & before any class definitions.
Syntax: import pkg1[.pkg2].(class namel *);
Pkg1 →name of the Top level package.
pkg2 →name of the subordinate package inside the outer package separated by a dot.
121)Write down the syntax for defining Interface?
An interface is defined similar to a class
Syntax: access interfacename{
Return type method-name1(parameter list);
Return type method-name2(parameter list);
type final –varname1 = value;
type final –varname1 = value;
//…

Return type method-nameN(parameter list);
type final –varname1 = value;
}

122)What are the steps to be followed while implementing interfaces?

To implement an interface, include the implements clause in a class definition,&
then create the methods defined by the interface.

Syntax: access class classname[extends super class]
Implements interface[,interface…]]{
//class body
}

access → either public or not used.

123)Write down the fundamentals of Exception Handling?

A java exception is an object that describes an exceptional condition that has
occurred in a piece of code. When an exceptional condition arises an object representing
that exception is created & thrown in the method that caused error.
Java Exception handling is managed through five keywords:try,catch,throw,
throws, and finally.

Syntax: try{
//block of code to monitor errors
}
catch(Expression type1 exob){
//exception handler for Exception type1
}
catch(Expression type1 exob){
//exception handler for Exception type1
}
//….
finally{
//block of code to be executed before try block ends.
}

124)Define Multithreaded Programming?

A Multithreaded program contains two or more parts that can run concurrently.
Each part of such a program is called a thread ,And each thread defines a separate path of
execution. Thus Multi threading is a specialized form of multitasking.

125)Mention some of the methods defined by Thread class?

Method	**Meaning**
getName	Obtain a thread's name
getPriority	Obtain a thread's priority
isAlive	Determine if a thread is still running
join	wait for a thread to terminate
run	Entry point for the thread
sleep	Suspend a thread for a period of time.

126)What are the two ways for creating a Thread

By implementing the runnable interface.
Class Newthread implements Runnable
By Extending the Thread class

Class Newthread extends Thread

Sixteen Mark Questions

1.What are the Features of Oop's & how are they implemented in C++?

- Objects.
- Classes.
- Data abstraction and Encapsulation.
- Inheritance.
- Polymorphism.
- Dynamic binding.
- Message passing.

2. Explain about inline function?

An inline function is a function that is expanded in line when it is invoked. That is compiler replaces the function call with the corresponding function code.

The inline functions are defined as

```
Inline function-header
{
function body
}
```

The situations where inline expansion may not work are

- for functions returning values, if loop, a **switch,** or a **goto** exists
- for functions not returning values ,if a return statement exists
- if function contain **static** variables
- if **inline** functions are recursive

3. Explain Function Overloading?

Function overloading means we can use the same function name to create functions that perform a variety of different tasks.

Eg: An overloaded add ( ) function handles different data types as shown below.

```
// Declarations
iv. int add( int a, int b); //add function with 2 arguments of same type
v. int add( int a, int b, int c); //add function with 3 arguments of same type
vi. double add( int p, double q); //add function with 2 arguments of
different type
//Function calls
i. add (3 , 4);
ii. add (3, 4, 5);
iii. add (3 , 10.0);
```

4. Explain about Operator Overloading?

C++ has the ability to provide the operators with a special meaning for a data type. This mechanism of giving such special meanings to an operator is known as Operator overloading. It provides a flexible option for the creation of new definitions for C++ operators.

The operators that cannot be overloaded are.

- Class member access operator (. , .*)
- Scope resolution operator (::)
- Size operator (size of)
- Conditional operator (?:)

The purpose of using operator function is to define an additional task to an operator, we must
specify what it means in relation to the class to which the operator is applied. This is done by Operator
function , which describes the task. Operator functions are either member functions or friend functions.
The general form is
return type classname :: operator (op-arglist)
{
function body
}
where *return type* is the type of value returned by specified operation.
Op-operator being overloaded. The *op* is preceded by a keyword operator. operator op is the function name. The rules for Operator overloading are

- Only the existing operators can be overloaded.
- The overloaded operator must have at least one operand that is of user
defined data type.
- The basic meaning of the operator should not be changed.
- Overloaded operators follow the syntax rules of the original operators.

They cannot be overridden.

5. Explain overloading Unary & Binary operator?

When unary operators are overloaded using member functions it takes no explicit arguments and return no explicit values.
When binary operators are overloaded using member functions, it takes one
explicit argument. Also the left hand side operand must be an object of the relevant class.
When unary operators are overloaded using friend function, it takes one reference argument (object of the relevant class)
When binary operators are overloaded using friend function, it takes two explicit arguments. The operator can be invoked using member functions as follows
In case of Unary operators, overloaded operator can be invoked as
op object_name or object_name op
In case of binary operators, it would be invoked as
Object . operator op(y)
where op is the overloaded operator and y is the argument.
The overloaded operator can be invoked using Friend function as
In case of unary operators, overloaded operator can be invoked as
Operator op (x);
In case of binary operators, overloaded operator can be invoked as
Operator op (x, y)
The operators that cannot be overloaded using Friend function.

- Assignment operator =
- Function call operator ( )
- Subscripting operator [ ]
- Class member access operator →

6. Explain about Type conversions?

The three types of data conversion are

i. Conversion from basic type to class type.
ii. Conversion from class type to basic type.
iii. Conversion from one class type to another class type.
A casting operator is a function that satisfies the following conditions

- It must be a class member.
- It must not specify a return type.
- It must not have any arguments.

The general form of overloaded casting operator is

```
operator type name ( )
{
……….. // function statements
}
```

It is also known as conversion function.
i. Basic to class type conversion
Conversion from basic data type to class type can be done in destination class.
Using constructors does it. Constructor takes a single argument whose type is to be converted.
Eg: Converting int type to class type

```
class time
{
int hrs,mins;
public:
………….
Time ( int t) //constructor
{
hours= t/60 ; //t in minutes
mins =t % 60;
}
};
```

Constructor will be called automatically while creating objects so that this conversion is done automatically.
ii. Basic type conversion with an example.
Using Type Casting operator, conversion from class to basic type conversion can be done. It is done in the source class itself.
Eg: vector : : operator double( )

```
{
double sum=0;
for(int I=0;I<size;I++)
sum=sum+v[ i ] *u[ i ] ;
return sqrt ( sum ) ;
}
```

This function converts a vector to the corresponding scalar magnitude.
iii. One class to another class conversion with an example.
Conversion from one class type to another is the combination of class to basic and basic to class type conversion. Here constructor is used in destination class and casting operator function is used in source class.

Eg: objX = objY
objX is the object of class X and objY is an object of class Y. The class Y type
data is converted into class X type data and the converted value is assigned to the obj X.
Here class Y is the source class and class X is the destination class.

7. Explain inheritance?

Inheritance is the process by which objects of one class acquire the properties of
another class. It supports the concept of hierarchical classification. It provides the idea of
reusability. We can add additional features to an existing class without modifying it by
deriving a new class from it.

i. single inheritance
If a single class is derived from a single base class is called single inheritance.
Eg:
Base class
Derived class
Here class A is the base class from which the class D is derived. Class D is the
public derivation of class B hence it inherits all the public members of B. But D cannot
access private members of B.

ii. multiple inheritance
If a class is derived from more than one base class, it is called multiple inheritance.
Eg: Base classes
Derived class
Here class C is derived from two base classes A & B.

iii. Hierarchical inheritance
If a number of classes are derived from a single base class then it is called
hierarchical inheritance.
Eg : Hierarchical classification of students in University

A
B
A
C
B

Student
Arts Engineering M e d i c a l
CSE ECE Civil

iv.Multilevel inheritance
If a class is derived from a class, which in turn is derived from another class, is
called multilevel inheritance. This process can be extended to any number of levels.
Eg:
Base class Grand father
Intermediate
Base class Father
Derived class Child

v. Hybrid inheritance
It is the combination of one or more types of inheritance. The class result will
have both the multilevel and multiple inheritances.

Multilevel
inheritance
Multiple
inheritance

8. Define constructor

A constructor is a special member function whose task is to initialize the objects of its class. It is special because its name is same as class name. The constructor is invoked whenever an object of its associated class is created. It is called constructor because it constructs the values of data members of the class

Eg:

```
Class integer
{
……
public:
integer( );//constructor
………
}
```

A
B
C
Student
Test
Result
Sports

The different types of constructor are

i. default constructor

The constructor with no arguments is called default constructor

Eg:

```
Class integer
{
int m,n;
Public:
Integer( );
…….
};
integer::integer( )//default constructor
{
m=0;n=0;
}
```

the statement

integer a;

invokes the default constructor

ii. parameterized constructor

constructor with arguments is called parameterized constructor

Eg;

```
Class integer
```

{ int m,n;
public:
integer(int x,int y)
{ m=x;n=y;
}
To invoke parameterized constructor we must pass the initial values as arguments to the constructor function when an object is declared. This is done in two ways
1.By calling the constructor explicitly
eg:
integer int1=integer(10,10);
2.By calling the constructor implicitly
eg:
Integer int1(10,10);
iii. Default argument constructor
The constructor with default arguments are called default argument constructor
Eg:
Complex(float real,float imag=0);
The default value of the argument imag is 0
The statement
complex a(6.0)
assign real=6.0 and imag=0
the statement
complex a(2.3,9.0)
assign real=2.3 and imag=9.0
iv. Copy constructor
A copy constructor is used to declare and initialize an object from another object. It takes a reference to an object of the same class as an argument
Eg: integer i2(i1);
would define the object i2 at the same time initialize it to the values of i1.
Another form of this statement is
Eg: integer i2=i1;
The process of initializing through a copy constructor is known as **copy initialization**.
v.Dynamic constructor
Allocation of memory to objects at time of their construction is known as dynamic constructor. The memory is allocated with the help of the **NEW operator**
Eg:
Class string
{
char *name;
int length;
public:
string( )
{
length=0;
name=new char[length +1];
}

```
void main( )
{
string name1("Louis"),name3(Lagrange);
}
```
use delete to free that memory.

9. Explain about Multiple constructors (constructor overloading)?

The class that has different types of constructor is called multiple constructors

Eg:
```
#include<iostream.h>
#include<conio.h>
class integer
{
int m,n;
public:
integer( ) //default constructor
{
m=0;n=0;
}
integer(int a,int b) //parameterized constructor
{
m=a; n=b;
}
integer(&i) //copy constructor
{
m=i.m;
n=i.n;
}
void main()
{
integer i1; //invokes default constructor
integer i2(45,67);//invokes parameterized constructor
integer i3(i2); //invokes copy constructor
}
```
The special characteristics of constructor are

- **T**hey should be declared in the public section
- They are invoked automatically when the objects are created
- They do not have return types, not even void and therefore, and they cannot return values
- They cannot be inherited, though a derived class can call the base class
- They can have default arguments
- Constructors cannot be virtual function

10. Explain virtual functions

A function qualified by the 'virtual' keyword is called virtual function. When a virtual function is called through a pointer, class of the object pointed to determine which function definition will be used.

The rules for virtual functions are

▪ Virtual functions must be member of some class.
▪ They cannot be static members and they are accessed by using object pointers
▪ Virtual function in a base class must be defined.
▪ Prototypes of base class version of a virtual function and all the derived class versions must be identical.
▪ If a virtual function is defined in the base class, it need not be redefined in the derived class.

Pure virtual functions

A pure virtual function is a function declared in a base class that has no definition relative to the base class. In such cases, the compiler requires each derived class to either define the function or redeclare it as a pure virtual function. A class containing pure virtual functions cannot be used to declare any object of its own. It is also known as "donothing"
function.

The "do-nothing" function is defined as follows:

virtual void display ( ) =0;

11. Explain the features of Java?

- Compiled and Interpreted
- Platform-Independent and portable
- Object oriented
- Robust and Secure
- Distributed
- Familiar, simple and small
- Multithreaded and Interactive
- High Performance
- Dynamic and Extensible

12. Describe the structure of Java program?

- Documentation section
- Package statement
- Import statements
- Interface statements
- Class definitions
- Main method class

13. Explain about Inheritance in Java?

- Defining a subclass
- Subclass constructor
- Single Inheritance
- Multilevel Inheritance
- Hierarchical Inheritance

Eg:

```
Class A
{
---------
---------
}
```

```
class B extends A
{
---------
---------
}
class C extends B
{
-------
--------
}
```

14. Explain about Interfaces in Java?

Java does not support multiple inheritances. It is achieved using interfaces in Java.

- Defining Interfaces
- Extending Interfaces
- Implementing Interfaces
- Accessing Interface variables

Eg :

```
Interface Area
{
final static flaot pi=3.14f;
float compute(float x,float y);
}
```

15. Explain about Packages?

Packages are java's way of grouping variety of classes or interfaces together.
Packages act as containers for classes

- Java API Packages
- Using System packages
- Creating Packages
- Accessing a package
- Using a package

Eg:

```
Package p1;
Public class A
{
public void display()
{
System.out,println("Class A");
}
}
```

16. Explain about Threads in Java?

A thread is a program that has a single flow of control

- Creating threads
- Extending the thread class

Eg:

```
Class Mythread extends Thread
{
```

```
------
------
}
```

• Using Runnable Interface

Eg:

```
Class MyThread implements Runnable
{
-------
-------
}
```

• Stopping and Blocking a thread

17. Explain about Strings in Java?

• Strings can be created using

▪ String class

String s= new String(“Hello”);

▪ StringBuffer class

StringBuffer s= new StringBuffer(“Hello”);

• String arrays

• String methods

18. Explain about Thread lifecycle?

• Newborn state

• Runnable state

• Running state

• Blocked state

• Dead state

19. Explain about Exception handling in Java?

• Exceptions

• Syntax of Exceptions handling state

• Multiple catch statements

• Using finally statement

Eg:

```
Class A
{
public static void main (String args[])
{
int a=10;
int b=5;
int c=5;
int x,y;
try
{
x=a/(b-c);
}
catch(ArithmeticException e)
{
System.out.println(“Division by zero”);
}
y=a/b+c;
System.out.println("y="+y);
}
}
```

20. Explain about Applet Lifecycle?

- Initialization state
- Running state
- Idle state or stopped state
- Dead state
- Display state

21. How Applets are prepared and executed?

- Writing Applets

Eg:

```
Import java.awt.*;
Import java.applet.*;
Public class A extends Applet
{
public void paint( Graphics g)
{
g.drawString("Hello",10,100);
}
}
```

- Building Applet code
- Designing a web page

```
<html>
<title>Applet</title>
<body>
<applet code= A.class width=400 height=200 </applet>
</body>
</html>
```

- Running the Applet
- Using appletviewer
- Using web browser

22. Explain about Visibility Controls?

- Public access
- Friendly access
- Protected access
- Private access
- Private protected access

23. Explain about Methods overriding and methods overloading?

Methods that have same name but different parameter lists and different definitions is called Method overloading.

Eg:

```
class Room
```

```
{
int width;
int length;
Room(int x,int y) // Constructor
{
length= x;
width = y;
}
Room(int x)
{
length=breadth=x;
}
}
```
When a method is called the method defined in the subclass is invoked and executed instead of the one in the superclass. This is called overriding.
Eg:
```
class superclass
{
void methodsuper()
{
System.out.println("superclass method");
}
}
class subclass extends superclass
{
void methodsuper()
{
System.out.println("overriding method of superclass");
}
}
class override
{
public static void main(String args[])
{
subclass sb=new subclass();
sb.methodsuper();
}
}
```
24 Explain about operators in Java?
- Arithmetic operators
- Integer arithmetic
- Real arithmetic
- Mixed mode arithmetic
- Relational operators
- Logical operators
- Assignment operators

- Increment and Decrement operators
- Conditional Operator
- Bitwise operator
- Special operators
- Instanceof operator
- Dot operator

25 Explain about Constructors in Java?

Constructors enable an object to initialize itself when it is created.

Eg: class Rect
```
{
int width;
int length;
Rect(int x,int y) // Constructor
{
length= x;
width = y;
}
int rectArea()
{
return(length *width);}
}
```

26. Explain the various Java tokens?

- Java character set
- Keywords
- Identifiers
- Literals
- Operators
- Separators

27. How will you implement a Java program?

- Creating the Java program

```
class sample
{
public static void main(String args[])
{
System.out.println("Hello");
}
}
```

Save the program as sample.java

- Compiling the program

The program is compiled using the statement and it will create a class file named sample.class.

```
Javac sample.java
```

- Running the program

```
Java sample
```

This statement will execute the program and display the output.

28. What are the features of simple Java program?

- Class declaration
- Opening brace
- Main Line
- Output line

Eg:

```
class sample // Class declaration
{ // Opening brace
public static void main(String args[]) // Main Line
{
System.out.println(“Hello”); // Output line
}
}
```