1.

q. Explain Java and its Characteristics.

Ans:- The Java is a programming language and also a platform too. For better understanding the Java as programming language and Java as a platform, lets first understand platform and programming language.

Platform

A platform is a major piece of software, as an operating system, an operating environment, or a database under which various smaller application programs can be designed to run.

Characteristics of Java

The primary objective of Java programming language creation was to make it portable, simple and secure programming language. Apart from this, there are also some excellent features which play an important role in the popularity of this language. The features of Java are known as Java buzzwords.

(i) Simple
(ii) Object-Oriented
(iii) Distributed
(iv) Robust
(v) Secure
(vi) system independence

(vii) Portability
(viii) Interpreted
(ix) High performance
(x) Multithreaded
(xi) Dynamic

① Simple :-

Java is very easy to learn, and its syntax is simple, clean and easy to understand. According to sun microsystem, Java language is a simple programming language because;

(a) Java syntax is based on c++ (so easier for programmers to learn it after c++).

(b) Java has removed many complicated and rarely-used features, for example :- explicit pointers, Operator overloading etc...

② Object-Oriented :-

Java is an Object-Oriented programming language. Everything in Java is an Object. Object-Oriented means, we Organize our software as a combination of different types of Objects that incorporate both data and behavior.

(iii) System (platform) Independent :-

Java is platform independent because it is different from other language like C, C++, etc. which are complied into platform specific machines while Java it is a write once, run anywhere language. A platform is the hardware or software in which a program runs.

(iv) Secured :-

Java is best known for its security. With Java, we can develop virus-free system.

Java is secured because, no explicit pointer.

Java programs runs inside a virtual machine sandbox.

(v) Dynamic :-

Java is a dynamic language. It supports the dynamic loading of classes. It means classes are loaded on demand. It also supports functions from its native language i.e C and C++.

(vi) Robust :-

The English meaning of Robust is strong. Java is Robust because :

(a) It uses strong management.

(b) There is a lack of pointers that avoids security problems.

(VII) Architecture - neutral :-

Java is architecture neutral because there are no implementation dependent features. for example the size of primitive type is fixed.

(VIII) Portable :-

Java is portable because it facilitates you to carry the Java bytecode to any platform. It doesn't require any implementation.

(IX) High - performance :-

Java is faster than other traditional interpreted programming language because Java bytecode is "close" to native code. It is still a little bit slower than a compiled language (eg. C++). Java is an interpreted language that is why it is slower than compiled language. eg. C, C++ etc.

(X) Distributed :-

Java is distributed because it facilitates users to create distributed applications in Java.

6. What are constants in Java? Explain it with suitable example.

Ans:- Constant is a value that cannot be changed after assigning it. Java does not directly support the constants. There is an alternative way to define the constants in Java by using the non-access modifiers static and final.

In Java, to declare any variable as constant, we use static and final modifiers. It is also known as non-access modifiers. According to the Java naming convention the identifier name must be in capital letters.

Example

```
import java.util.Scanner;
public class Constant
{
  // declare Constant
private static final double PRICE = 234.90;
  public static void main (String[] args)
  {
  int unit;
  double total_bill;
System.out.print (" Enter the number of units
      you have used: ");
```

```
Scanner sc = new Scanner(System.in);
    Unit = sc.nextInt();
total_bill = PRICE * Unit;
System.out.println("The total amount you
have to deposit is:" + total_bill);
    }
    }
```

2. Explain the operators used in Java.

Ans:- Operators are special sign or symbol, which performs an operation or evaluation on one or more operands. Operands are value or variable declare within program. for example 10+5 where '+' sign is an operator which indicates addition operation between 10 and 5 operands. There are different types of operators supported by Java are :-

(a) Arithmetic operator
(b) Comparison operator
(c) Logical operator
(d) Assignment operator
(e) Conditional operator

(a) Arithmetic operator :-

An arithmetic operator is a mathematical function that takes two operands and performs a calculation on them. '+', '-', '*', '/', % (modulus), ++ (increment), -- (decrement) are the arithmetic operators.

(b) Comparison operator :-

Comparison operators can compare numbers or string and perform evaluations. Expressions that use comparison

operators do not return a number value as do arithmetic expression. Comparison expressions return either 1, which represents true or 0, which represents false.
==, !=, <, >, <=, >= are the comparison operator.

(c) Logical Operator :-
A logical operator is a symbol or word used to connect two or more expressions such that the value of the compound expression produced depend.
&&, ||, != are logical operator.

(d) Assignment Operator :-
An operator which is used to store a value into a particular variable is called assignment operator in java.
=, +=, -=, *=, /=, %= are the Assignment operator.

(e) Conditional Operator :-
Conditional operator also known as ternary operator. The conditional statements are the decision making statement which depend upon the output of expression. It is represented by two symbol i.e ? and : (colon). As conditional operator works on three operand. so, it is

also known as ternary operator.

Syntax

(condition) ? (statement 1 if condition is true) : (statement 2 if condition is false).

3 a. Explain the concept of two-dimensional Array in Java.

Ans:- The two dimensional Array in Java is nothing but an Array of Arrays. In Java two dimensional Array, data stored in row and columns, and we can access the record using both the row index and column index (like an Excel File).

If the data is linear, we can use the one Dimensional Array. However, to work with multi-level data, we have to use the multi-Dimensional Array. Two Dimensional Array in Java is the simplest form of Multi-Dimensional Array.

Example

```
import java.util.Scanner;
public Class Array2D
{
```

```
public static void main(String[] args)
{
Scanner sc = new Scanner(System.in);
System.out.println("Enter size of array");
int a = sc.nextInt();
int b = sc.nextInt();
int arr[][] = new int[a][b];
System.out.println("Enter elements of
  array");
for(int i=0; i<a; i++)
  {
for(int j=0; j<b; j++)
  {
arr[i][j] = sc.nextInt();
  }}
System.out.println("Element of array
  are:\n");
for(int i=0; i<a; i++)
  {
for(int j=0; j<b; j++)
  {
System.out.println(arr[i][j]+" ");
  }
System.out.print("\n");
  }
  }
  }
```

4. What is the use of package in Java? Explain the process of creation of packages.

Ans:- Packages are used in Java in order to prevent naming conflicts, to control access, to make searching locating and usage of classes, interfaces, enumerations and annotations easier etc. A package can be defined as a grouping of related types (classes, interfaces, enumerations and annotations) providing access protection and namespace management.

Example

```
package mypack;
public class firstClass
{
public void show()
{
system.out.println("I am first class");
}
}
```

```
package mypack;
public class secondClass
{
public void show()
```

```
{
System.out.println("I am second class");
}
}
```

```
package mypack;
public class Thirdclass
{
public void show()
{
System.out.println("I am third class");
}
}
```

```
import mypack.Firstclass;
import mypack.Secondclass;
import mypack.Thirdclass;
class packImmethod
{
public static void main(String[] args)
{
Firstclass f = new Firstclass();
Secondclass s = new Secondclass();
Thirdclass T = new Thirdclass();
F.show();
S.show();
T.show();
}
}
```

6a. How will you implement an interface? Explain with an example.

Ans:- An interface is used as "superclass" whose properties are inherited by a class. A class can implement one or more than one interface by using a keyword implements followed by a list of interfaces separated by commas.

When a class implements an interface, it must provide an implementation of all methods declared in the interface and all its super interfaces.

Example

```
interface InterfaceA
{
void display();
int x=10, y=15;
void sum();
}
class ClassB implement InterfaceA
{
public void display()
{
system.out.println("This is display");
}
public void sum()
{
```

```
System.out.println("sum is :" +(x+y));
    }}
class classc implements InterfaceA
            {
public void display()
            {
system.out.println("this is display");
            }
public void sum()
            {
system.out.println("sum is:" +(x+y));
            }
public static void main(String[] args)
            {
Interface A ob = new classc();
    ob.display();
    ob.sum();
        }
        }
```

7.9. Explain polymorphism in Java.

Ans:- Polymorphism is the ability of an object to take on different forms. In Java, polymorphism refers to the ability of a class to provide different implementations of a method, depending on the type of object that is passed to the method.

To put it simply, polymorphism in Java allows us to perform the same action in many different ways. Any Java Object that can pass more than one IS-A test is polymorphic in Java. Therefore, all the Java Objects are polymorphic as it has passed the IS-A test for their own type and for the class Object.

Example

```
class Shapes
    {
    public void area()
        {
        System.out.println("The formula for area of");
        }
    }
class Triangle extends Shapes
{
  public void area() {
    System.out.println("Triangle is 1/2 * base * height");
}}
class Circle extends Shapes {
  public void area() {
  System.out.println("Circle is 3.14 * radius * radius");
}}
class Main {
public static void main(String[] args) {
  Shapes myshape = new Shapes();
```

```
Shapes myTriangle = new Triangle();
Shapes myCircle = new Circle();
myShape.area();
myTriangle.area();
myCircle.area();
    }
}
```

b. Explain basic data types used in Java.

Ans:- A data type in programming, is a classification that specifies which type of value a variable has that is made as per convenience and circumstances and what type of mathematical, relational or logical operations can be applied to it without causing an error.

Basic data types used in Java are explain below:-

(1) int :-

It is a 32-bit signed two's complement integer.

Size :- 4 byte (32 bits)

Value :- -2,147,483,648 to 2,147,483,647 (inclusive).

(iii) Boolean :-

Boolean data type represents only one bit of information either true or false which is intended to represent the two truth values of logical and Boolean algebra.

(iv) byte :-

The byte data type is an 8-bit signed two's complement integer.

Size : 1 byte (8 bits)

value :- -128 to 127.

(v) float :-

The 'float' data type is a single-precision 32-bit.

Size : 4 byte (32 bits)

value :- upto 7 decimal digits.

(vi) double :-

The double data type is a double precision 64-bit.

Size : 8 bytes or 64 bits

values : upto 16 decimal digits

(vii) Char :-

The char data type is a single 16-bit Unicode character.

size : 2 byte (16 bits)

8. What is abstract? Write the simple Java program that reads data from one file and writes data to another file.

Ans:- The abstract keyword is a non-access modifier, used for classes and methods.

- Abstract class :- is a restricted class that cannot be used to create object (to access it, it must be inherited from another class.
- Abstract method :- Can only be used in an abstract class, & it does not have a body.

```
import java.io.*;
class File {
public static void main(String[] args)
        {
File inf = new File("in.dat");
File outf = new File("out.dat");
 FileReader ins = null;
 FileWriter outs = null;
 try {
  ins = new FileReader(inf);
  outs = new FileWriter(outf);
```

```
int ch;
while ((ch = ins.read()) != -1) {
    outs.write(ch);
}
}
Catch (IOException e) {
System.out.println(e);
System.exit(-1);
System. }
    finally {
    try {
ins.close();
outs.close();
}
Catch (IOException e) { }
    }
    }
    }
```

9. Write short notes on:

(a) Dynamic Binding

↳ Binding refers to the linking of a procedure call to the code to be executed in response to the call. The Dynamic Binding (also called late binding) means that the code associated with a given procedures call is not known until the time of the call at run time.

(b)

Super classes and Sub classes

Superclasses (Base class)

↳ In an object-oriented programming language, a base class is an existing class from which the other classes are determined and properties are inherited. It is also known as base class or parent class. In general, the class which acquires the base class can hold all its member and some further data as well.

Syntax

```
class base_classname
{
:
};
```

Sub classes

↳ A sub class is a class that is constructed from a base class or an existing class. It has a tendency to acquire all the methods and properties of a super class. It is also known as a derived class or child class.

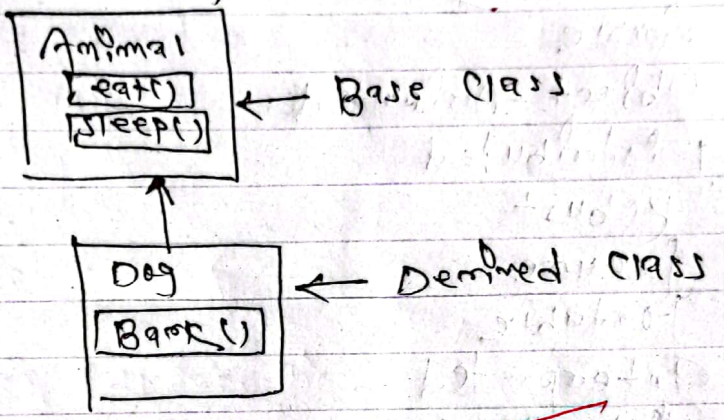
Syntax:

~~base class~~

class derived_class name : access_mode base_class_name

{

}



(c.) this Keyword

↳ The this keyword refers to the current object in a method or constructor. The most common use of the this keyword is to eliminate the confusion between class attributes and parameters with the same name (because a class attribute is shadowed by a method or constructor parameter). If you omit the keyword in the example above, the output would be "0" instead of "5".

(d) Java "White paper" buzzwords

↳ The authors of Java have written an influential white paper that explains their design goals and accomplishments. Their paper is organized along the following eleven buzzwords:

(i) Simple
(ii) Object-Oriented
(iii) Distributed
(iv) Robust
(v) Secure
(vi) Portable
(vii) Interpreted
(viii) High performance
(ix) Multithreaded
(x) Dynamic
(xi) System independence

07
12/08