Set-5

Regd NO:19BQ1A05P0
Name: V. Tejaswini
Sec: CSE-D ①

1) List and explain java buzzwords. which factors are making Java famous language.

Ans The following are the list of buzzwords

- Simple
- portable
- object-oriented
- Robust
- Multithreaded
- Architecture-neutral
- Interpreted
- High perfomence
- Distributed
- Dynamic

Simple:

Java was designed to be easy for the professional programmes to learn and we effectively. Java inherits the C/C++ syntax and many of the object-oriented features of C++, most programmes have little trouble learning java.

Secure:

Java achieved protection by confinning an applet to the java execution environment and not allowing it access to othe parts of the computer. The ability to downloaad applets that no harm will be done and no security will be breaked is considered to be the single most innovative aspect of java.

Portable:

It is not practical to have different versions as there are many different types of computer and operating systems

(2)

therefore some means of generating portable executable code was needed.

Object - Oriented

Borrowing liberally from many seminal object software environments, Java manages to strike a balance between the purists "Everything is an object" paradigm. the object model in java is simple and easy to extend. while primitive types, such as integer are kept as high-performance non objects.

Robust :

As the program must execute reliably in variety of systems. The ability to create robust programs was given a high priority in the design of java. Java is a strictly type language checks your code at compile time However it also checks at runtime

Multi thread :

It is used to meet the real-world requirement of creating interactive networked programs, java supports multithreaded programming which allows you to write programs that do many things simultaneously. Java run-time system enables you to construct smooth running interactive systems.

Architecture - Neutral :

A central issue for the java designers was that of code longevity and probability. operating system upgrades, processor upgrades and changes in core system resources can all combine to make a program malfunction. In an attempt to alter this situation, the goal "write once, run anywhere, anytime, forever was accomplished.

Interpreted and high perfomance :

Java enables the certain of cross-platform program compling into an intermediate representation called Java byte code. Java byte code was carefully designed so that it would be eassy to translate directly into native machine for very high perfomence by using a just-in-time compiler

Distributed :
Java is designed for the distributed environment of the Internet because it handles TCP/IP protocols. Java also supports. Remote Method Invocation (RMI) which enables a program to methods across a network.

Dynamic :
Java programs carry with them substantial amounts of runtime type information that is used to verify and resolve access to objects at runtime. This makes it possible to dynamically link code in a safe and expedient manner. This is crucial to the robustne of the java environment, in which small fragments of bytecode may be dynamically updated on running system.

Factors Making Java Famous Language :

1) Inertia; It became the standard programming language for most enterprise software development
2) Huge ecosystem of frameworks and libraries
3) Design patterns
4) carrer paths
5) Free to use
6) Software tools
7) Scientific Applications
8) Creating and designing Android, web Applications.

2) What are the benefits of inheritance? Explain various forms of inheritence with suitable code segments

Ans. Advantages of inheritence :

→ Inheritence minimizes the ammount of duplicate code in an application by sharing common code amongst several subclasses
→ Result in a better organization of code and smaller, simpler compilations units

→ Makes application code more flexible to change.

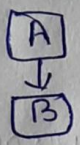
→ Extending the base class logic as per logic of the derived class.

→ Class can decide to keep some data private so that be altered by the derived class.

→ we will be able to override the methods of base class so that implementation of base class method can be designed in the derived class.

various types of inheritance are as follows:

1) Single Inheritence: when a class extends another one class only, then it is called single Inheritence.



Class B extends only one class which is B

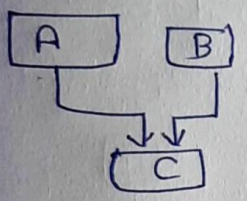
Here A is a parent class of B and B will be a child class of A. The following is an example program single inheritence in Java.

code:

```
Class A
{
public void methodA()
{
   System.out.println("Parent class method");
} }
}
Class B extend A
{
   public void methodBB()
   {
    System.Outprintln("Child class method");
    public static void main(String args[])
    {  B obj = new B();
    Obj.methodA(); //calling super class method
```

```
        obj.methodB(); // calling local method
    }
}
```

2) **Multiple Inheritence:** It refers to concept of one class extending (or Inheriten) more than one base class. Multiple Inheritence is a feature of object oriented concept, where a class can inherit properties of more than one present class. The problem occurs when there exist methods with same signature in both super classes and subclass and thereby compiler cannot determine which class method to be called and gets the priority.



→ Multiple inheritence is very rarely used in software projects. Using multiple inheritence leads to problems and complexity when further extending class

Code

```
class Parent1
{
    void Fun()
    {
        System.out.println("Parent1");
    }
}
class Parent2
{
    void Fun()
    {
        System.out.println("Parent 2");
    }
}
// Error: Test is inheriting from multiple classes
class Test extends Parent1, Parent2
{
    public static void main(String args[])
    {
        Test t = new Test();
        t.Fun();
    }
}
```

Output :- Compiler error // complications as whether to call parent1's func() or parent2's fun() method */

3) Multilevel Inheritence :-

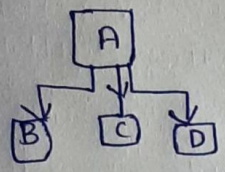
When a class extends a class, which extends another class then this is called multilevel inheritance

Here c is subclass or child class of B and B is child class of A

Code :-

```
class X
{
 public void methodX()
 { System.out.println("class X method");
 }
}
class Y extends X
{ public void methodY()
  {
   System.out.println("class Y method");
  }
}
 Class Z extends Y
  { public void methodZ()
    {
     System.out.println(" class Z method");
    }
    public static void main(String args[])
    {
     Z obj = new Z();
     obj.methoX(); //calling grand parent class method
     Obj.methodY(); //calling parent class method
     obj.methodZ(); //calling local method.
```

4) Hierarchical Inheritence:- In this, one class is inherited by many sub classes.



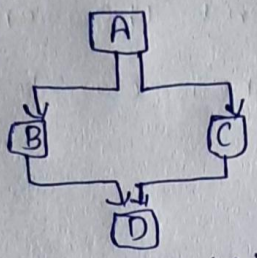
Here class B,C,D inherit the some class A. A is parent class (or Base class) of B, C & D

Code

```
class A
{
  public void methodA()
  { System.out.println("method of class A");
  }
}
class B extends A
{
  public void method B()
  {
    System.out.println("method of class B");
  }
}
class C extends A
  {public void methodC()
    {
      System.out.println("method of class C");
    }
  }
  class D extends A
  {
    public void methodD()
      {
        system.out.println("method of class D");
      }
  }
  class Example
  {public static void main (String args[])
    {
      B obj1 = new B();
      C obj2 = new C();
      D obj3 = new D();
      obj1.methodA();
      obj2.methodA();
      obj3.methodA();   /*All classes can access method of
                          class A*/
  }}
```

5, Hybrid Inheritence :

Hybrid Inheritence is a combination of single and multiple inheritence (or) more than one types of inheritance.



Hybrid inheritence

(3) Define a class named movieMagic with the following description

Instance variables (data members) :

Int year - to store the year of release of a movie

String title - to store the title of the movie

float rating - to store the popularity rating of the movie

(minimum rating = 0.0 and maximum rating = 5.0)

Member methods :-

(i) movieMagic() :- Default constructor to initialize numeric data members to 0 and string data members to " "

(ii) void accept() : To input and store year, title and rating

(iii) void display() : To display the title of a movie and a message based on the rating as per the table below.

Rating	msg to be display
0.0 to 2.0	Flop
2.1 to 3.4	semi-hit
3.5 to 4.5	Hit
4.6 to 5.0	Superhit

Rating	Mesage to displayed
0.0 to 2.0	Flop
2.1 to 3.4	Semi-hit
3.5 to 4.5	Hit
4.6 to 5.0	superhit

Program

```
public class movieMagic{
    private int year;
    private String title;
    private Float rating;
    public movieMagic() {
        this.year = 0;
        this.title = " ";
        this.rating = 0;
    }
    public void accepst (int year, String title, Float rating){
        this.year = year;
        this.title = title;
        this.rating = rating;
    }
    public void display(){
        if(rating >=0.0 && rating <=2.0)
            System.out.println("Flop");
        else if (rating >=2.1 && rating <=3.4)
            System.out.println("semi hit");
        elseif(rating >=4.6 && rating <=4.5)
            Sytem.out.println("please enter positivenumbers");
        else
            System.out.println("please enter positive numbers");
    }
    public static void main(String args[]){
        movieMagic ob = new movieMagic();
        ob.accept(2000, "ABCB", 4.4F);
        ob.display();
    }
}
```

Output: Hit

4) Write a class to overload a function num_calc() as follows;

i) void num_calc (int num, char ch) with one integer argument and one character argument, computes the square of integer argument if choice ch is 'S' otherwise finds its cube.

ii) void num_calc (int a, int b, char ch) with two integer arguments and one character argument It computes the product of integer arguments IF ch is 'P' else adds the integers.

iii) void num_calc (String S1, String S2) with two string arguments, which prints whether the strings are equal or not

Program

```
class OverloadDemo{
        void num_calc(int,num,charch){
            if (ch=='S')
                System.out.println(num*num);
            else
                System.out.println(num*num*num);
        }
        void num_calc(int a, int b, charch){
            if(ch==('P')
                System.out.println(a*b);
            else
                System.out.println(a+b);
        }
        void num_calc(String s1, String s2){
            if(S1==S2)
                System.out.println("strings are equal");
            else
                System.out.println("Strings are not equal");
        }
class Overload{
        public static void main(String args[]){
            OverloadDemo ob= new OverloadDemo();
            ob.num_calc(2,'a');
            ob.num_calc(5,6,'p');
            Ob.num_calc("abcd","ABCD");
        }
}
```

O/P :- Strings are not equal.