Hall Ticket NO :- | 1 | 9 | B | Q | 1 | A | 0 | 5 | K | 0 |

Name : Shaik Heena Kousar Branch : CSE Subject : Java

Set-5

1) List and explain Java buzzwords. Which factors are making Java famous language?

Following are the list of Java buzzwords

- Simple
- Secure
- portable
- Object oriented
- Robust
- Multithreaded
- Architecture-neutral
- Interpreted
- High performance
- Distributed
- Dynamic

Simple :

Java was designed to be easy for the professional programmer to learn and use effectively. If you already understand the basic concepts of object-oriented programming learning Java will be even easier. Most programmers have little trouble in learning Java.

19BQ1A05K0

Secure :

Java provides a "firewall" between a networked application and your computer. When a Java Compatible Web browser is used, downloading can be done safely without fear of viral infection.

portable :

Java provides a way to download programs dynamically to all the various types of platforms connected to the Internet. It helps in generating portable executable Code.

Object Oriented :

Java is true object oriented language. Almost "Everything is an Object". All program code and data reside within Objects and classes. the Object model in Java is simple and easy to extend. Java comes with an extensive set of classes, arranged in packages that can be used in our programs.

Robust :

Since Java is platform independent, the programs need to be robust. programs written in Java should work in various operating systems and they should not fail or function differently. Java is strictly typed language. Java achieves robustness by checking the code both at compile time and runtime.

Multithreaded :

Java supports multithreaded programming which allows us to write programs that do many things simultaneously.

19B01A05K0

Architecture Neutral :

Java language and Java Virtual Machine helped in acheiving the goal of "Write once; run anywhere, any time, forever." Since changes like processors upgrade, operating systems upgrade could impact the programs which are dependent on architechture. Java is designed to be architechture neutral and hence these changes donot impact the programs.

Interpreted and Compiled :

Usually a computer language is either Compiled or Interpreted. Java combines both this approach and make it a two-stage system.

Compiled : Java enables creation of a cross platform programs by compiling into an intermediate representation called JavaByteCode

Interpreted : ByteCode is then interpreted, which generates machine code than can be directly executed by the machine that provides a Java Virtual machine.

High performance :

Java performance is high because of the use of bytecode. The byte code was used, so that it was easily translated into native machine code.

Distributed :

Java is designed for distributed environment of the internet because it handles TCP/IP protocols. It also supports RMI (Remote Method Invocation), which enables programs to invoke methods across a network.

19BQ1A05K0

Dynamic :

Java programs have lots of run-time information which is used in verifying and resolving access to objects at runtime. Due to this, it is possible to update small fragments of bytecode on a running system. Which means we need not stop and restart program for small code changes.

Factors which making Java famous language

1. Java is easy to learn
2. Java has Rich API
3. Java is User-Friendly language
4. Java is highly in Demand
5) Java has Adaptability and Stability
6) Java has powerful Development Tools
7) Java is FREE
8) powerful development tools eg. Eclipse, Netbeans
9) Java is an Object-Oriented programming language
10) Java allows developers to write once, run anywhere
11) Java has great collection of open source libraries
12) Java has Wonderful Comunity Support

These factors are making Java a famous language.

19BQIA05K0

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

Inheritance can be defined as the process where one class acquires the properties (methods and fields) of another. The class which inherits the properties of other is known as subclass and the class whose properties are inherited is known as superclass (Base class, or parent class)

Benefits of inheritance:

- A code can be used again and again
- Inheritance in Java enhances the properties of the class, which means that the property of the parent class will automatically be inherited by the super class (baseclass, parent class)
- The main advantages of Inheritance are code reusability and readability.
- Inheritance increases coupling between Superclass and Subclass (Derived class or child class)
- By using inheritance Redundancy (repetition) of the code is reduced or minimized so that we get consistence results and less storage cost.
- By using inheritance, Application performance is enhance (improved)

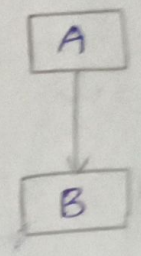
19B01A05K0

Various forms of Inheritance

Below are the various types of inheritance in java.

1) Single Inheritance

When a class extends another one class by only then we call it a single inheritance. The below flow diagram shows that class B extends only one class which is A. Here A is a parent class of B. And B would be a child class of A



Single Inheritance

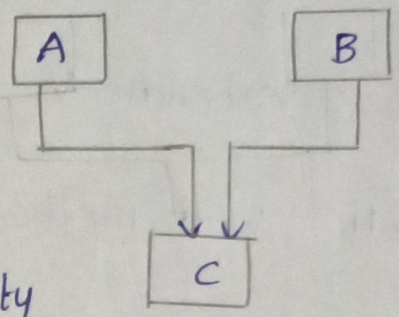
Single inheritance example

```
class A
{
  public void methodA()
  {
    System.out.println("Base class method");
  }
}
class B extends A
{
  public void methodB()
  {
    System.out.println("Child class method");
  }
  public static void main(String args[])
  {
    B obj = new B();
    obj.methodA(); // calling super class method
    obj.methodB(); // calling local method
  }
}
```

19BQ1A05K0

2) Multiple Inheritance :

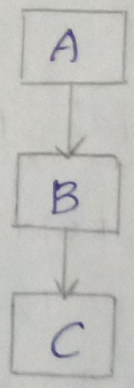
"Multiple Inheritance" refers to the concept of one class extending (or inherits) more than one base class. The inheritance we learnt earlier had a concept of one base class or parent. The problem with "multiple inheritance" is that the derived class will have to manage the dependency on two base class.



- Multiple Inheritance is very rarely used in software projects. Using Multiple Inheritance often leads to problems in hierarchy. This results in unwanted complexity when further extending the class.
- Java doesn't allow multiple Inheritance

3) Multilevel Inheritance

Multilevel Inheritance refers to a mechanism in OO technology where one can inherit from a derived class, thereby making this derived class the base class for the newclass. As you can see in below flow diagram C is subclass or child class of B and B is a child class of A



Multilevel Inheritance example

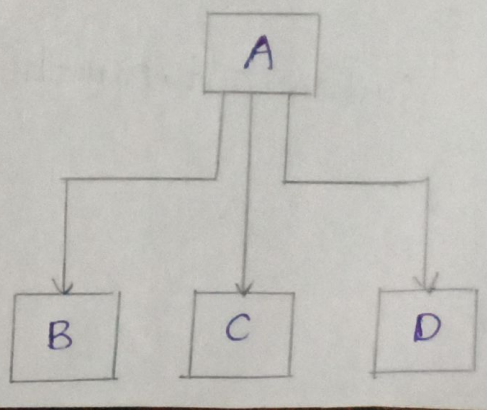
```
class X
{
  public void methodX( )
  {
    System.out.println("class X method");
  }
}
```

```
class Y extends X
{
public void method Y()
{
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.method X(); // calling grant parent class method
  obj.method Y(); // calling parent class method
  obj.method Z(); // Calling local method
 }
}
```

4) Hierarchial Inheritance

In such kind of Inheritance one class is inherited by many sub classes. In below example class B, C and D inherits the same class A. A is parent class(or base class) of B, C & D.

Example for Hierarchial Inheritance:




```
class A
{
 public void methodA()
 {
```

19BQ1A05K0

```
        System.out.println("method of class A");
    }
}
class B extends A
{
 public void methodB()
  {
     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 JavaExample
{
 public static void main(String args[])
 {
     B obj1 = new B();
     C obj2 = new C();
     D obj3 = new D();
                    // all classes can access the method of class A
     obj1.methodA();
     obj2.methodA();
     obj3.methodA();
  }
}
```

Output:

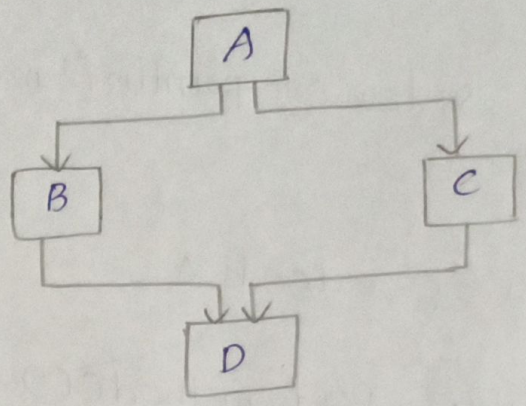
```
Method of class A
Method of class A
Method of class A
```

5) Hybrid Inheritance :

19BQ1A05K0

In simple terms you can say that Hybrid inheritance is a combination of single and Multiple inheritance. (or) more than one types of inheritance. A hybrid inheritance can be achieved in the java in a sameway as multiple inheritance can be!

The following example inherits (single and hierarchial inheritance) together to form hybrid inheritance




```
class C
{
  public void disp()
  {
    System.out.println("C");
  }
}
class A extends C
{
  public void disp()
  {
    System.out.println("A");
  }
}
class B extends C
{
  public void disp()
  {
    System.out.println("B");
  }
}
class D extends A
{
  public void disp()
  {
    System.out.println("D");
  }
  public static void main(String args[]){
      D obj = new D();
      obj.disp();
  }
}
```

In the example program

Class A and B extends class C
→ Hierarchial inheritance

Class D extends class A
→ Single inheritance

output :- D

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

19BQ1A05K0

Instance variables / data members :

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

String title - to store 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 member 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	Message to be 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	Super Hit

Write a main method to create an object of the class and call the above member methods

19BQ1A05K0

```
impost  java.util.Scanner;
class movieMagic
{
 int year;
 String title;
 float rating;
 movieMagic()
 {
   year = 0;
   rating = 0.0f;
   title = " ";
 }
 void accept()
 {
  Scanner sc = new Scanner(System.in);
  System.out.print("Enter the title of the movie: ");
  title = sc.next();
  System.out.print("Enter the year of its release: ");
  year = sc.nextInt();
  System.out.print("Enter the movie rating: ");
  rating = sc.nextFloat();
 }
 void display()
 {
   System.out.println("The title of the movie is : "+title);
   if(rating >=0.0 && rating <= 2.0)
   {
     System.out.println("Flop")
   }
```

19BQ1A05K0

```
else if (rating >= 2.1 && rating <= 3.4)
{
    System.out.println(" Semi-hit");
}
else if (rating >= 3.5 && rating <= 4.5)
{
    System.out.println("Hit");
}
else if (rating >= 4.6 && rating <= 5.0)
{
    System.out.println ("Super Hit");
}
else
{
    System.out.println ("Rating should be between 0.0 and 5.0");
}
}
public static void main(String args[])
{
  movieMagic ob = new movieMagic();
  ob.accept(2018, " Bahubali", 4.9f);
  ob.display();
}
}
```

1[19BQ1A05K0

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 find 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.

```
import java.io.*;
import java.util.*;
class Overloaded
{
    void num-calc(int num, char ch)
    {
        int s=0;
        if (ch == 's')
            s = num*num;
        else
            s = num*num*num;
        System.out.println("s = " + s);
    }
    void num-calc(int a, int b, char ch)
    {
        int s=0;
```

19BQ1A05K0

```
        if(ch==='p')
            s = a*b;
        else
            s = a+b;
        System.out.println("s = "+s);
    }
    void num_calc(String s1, String s2)
    {
        if(s1==s2)
            System.out.println("Strings are equal");
        else
            System.out.println("Strings are not equal");
    }
}
public static void main(String args[])
{
    Scanner sc = new Scanner(System.in);
    Overloaded ob = new Overloaded();
    ob.num_calc(3, 's');
    ob.num_calc(5, 7, 'q');
    ob.num_calc("Aman","Ayan");
}
}
```

Reference Books : The complete Reference, Introduction to Java programing

Internet links : https://beginnersbook.com/2013/03/inheritance-in-Java/