Most Popular Java Programming Interview Questions

Write a Java Program to reverse a string without using String inbuilt function.

```
public class Reversestr{

        public static void main(String[] args) {

                String str = "malayalam";
                StringBuilder str2 = new StringBuilder();
                str2.append(str);
                str2 = str2.reverse();     // used string builder to reverse
               System.out.println(str2);
                }

}
```

o/p

malayalam

Write a Java Program to reverse a string without using String inbuilt function reverse().

```
public class Revnotusinginfunc{
      public static void main(String[] args) {
            String str = "malayalam";
            char chars[] = str.toCharArray();
            // converted to character array and printed in reverse order
            for(int i= chars.length-1; i>=0; i--) {
                  System.out.print(chars[i]);
            }
      }
}
```

o/p

malayalam

Another method

```
import java.util.Scanner;

public class ReverseSplit {

	public static void main(String[] args) {
		String str;
		Scanner in = new Scanner(System.in);
		System.out.println("Enter your String");
		str = in.nextLine();
		String[] token = str.split("");
		    //used split method to print in reverse order
		for(int i=token.length-1; i>=0; i--)
		{
			System.out.print(token[i] + "");
		}

	}

}
```

Another method without using split() [and using charAt()]

```
original = in.nextLine();
		int length = original.length();
		for(int i=length-1; i>=0; i--) {
			reverse = reverse + original.charAt(i);
			 }
		System.out.println(reverse);
```

Write a Java Program to find whether a number is prime or not.

```
import java.util.Scanner;

public class Prime {

	public static void main(String[] args) {
		int temp, num;
		boolean isPrime = true;
		Scanner in = new Scanner(System.in);
		num = in.nextInt();
		in.close();
		for (int i = 2; i<= num/2; i++) {
			temp = num%i;
			if (temp == 0) {
				isPrime = false;
				break;
```

```
                }
            }
            if(isPrime)
                System.out.println(num + "number is prime");
                else
                    System.out.println(num + "number is not a prime");
        }

}
```

Write a Java Program to find whether a string or number is palindrome or not.

```
import java.util.Scanner;

public class Palindrome {
    public static void main (String[] args) {
        String original, reverse = "";
        Scanner in = new Scanner(System.in);
        int length;
        System.out.println("Enter the number or String");
        original = in.nextLine();
        length = original.length();
        for (int i =length -1; i>;=0; i--) {
            reverse = reverse + original.charAt(i);
        }
        System.out.println("reverse is:" +reverse);

        if(original.equals(reverse))
            System.out.println("The number is palindrome");
        else
            System.out.println("The number is not a palindrome");

    }
}
```

Write a Java Program for the Fibonacci series.

```
import java.util.Scanner;

public class Fibonacci {
    public static void main(String[] args) {
        int num, a = 0,b=0, c =1;
        Scanner in = new Scanner(System.in);
        System.out.println("Enter the number of times");
        num = in.nextInt();
        System.out.println("Fibonacci Series of the number is:");
        for (int i=0; i<num; i++) {
            a = b;
            b = c;
            c = a+b;
```

```
                System.out.println(a + "");
                //if you want to print on the same line, use print()
            }
        }
}
```

Write a Java Program to iterate ArrayList using for-loop, while-loop, and advance for-loop.

```
import java.util.*;

public class arrayList {
    public static void main(String[] args) {
        ArrayList list = new ArrayList();
        list.add("20");
        list.add("30");
        list.add("40");
        System.out.println(list.size());
        System.out.println("While Loop:");
        Iterator itr = list.iterator();
        while(itr.hasNext()) {
            System.out.println(itr.next());
        }
        System.out.println("Advanced For Loop:");
        for(Object obj : list) {
            System.out.println(obj);
    }
        System.out.println("For Loop:");
        for(int i=0; i<list.size(); i++) {
            System.out.println(list.get(i));
        }
}
}
```

Write a Java Program to find the duplicate characters in a string.

```
public class DuplicateCharacters {
        public static void main(String[] args) {
                String str = new String("Sakkett");
                int count = 0;
                char[] chars = str.toCharArray();
                System.out.println("Duplicate characters are:");
                for (int i=0; i<str.length();i++) {
                        for(int j=i+1; j<str.length();j++) {
                                if (chars[i] == chars[j]) {
                                    System.out.println(chars[j]);
                                        count++;
                                        break;
                                }
                        }
                }
        }
```

}

Write a Java Program to find the second-highest number in an array.

```
//package codes;
public class SecondHighestNumberInArray {
public static void main(String[] args)
	{
		int arr[] = { 100,14, 46, 47, 94, 94, 52, 86, 36, 94, 89 };
		int largest = 0;
		int secondLargest = 0;
		System.out.println("The given array is:");
		for (int i = 0; i < arr.length; i++)
		{
		   System.out.print(arr[i] + "\t");
		}
		for (int i = 0; i < arr.length; i++)
		{
			if (arr[i] > largest)
			{
				secondLargest = largest;
				largest = arr[i];
			}
			else if (arr[i] > secondLargest)
			{
				secondLargest = arr[i];
			}
		}
		System.out.println("\nSecond largest number is:" + secondLargest);
		System.out.println("Largest Number is: "  +largest);
	}
}
```

Write a Java Program to check Armstrong number.

```
class Armstrong{
  public static void main(String[] args)  {
     int c=0,a,temp;
     int n=153;//It is the number to check Armstrong
     temp=n;
     while(n>0)
     {
     a=n%10;
     n=n/10;
      c=c+(a*a*a);
      }
      if(temp==c)
      System.out.println("armstrong number");
      else
		System.out.println("Not armstrong number");
     }
```

```
}
```

Write a Java Program to remove all white spaces from a string with using replace().

```
class RemoveWhiteSpaces
{
	public static void main(String[] args)
	{
		String str1 = "Saket Saurav	is a QualityAna	list";

		//1. Using replaceAll() Method

		String str2 = str1.replaceAll("\\s", "");

		System.out.println(str2);

		    }
}
}
```

Write a Java Program to remove all white spaces from a string without using replace().

```
class RemoveWhiteSpaces
{
	public static void main(String[] args)
	{
		String str1 = "Saket Saurav	is an Autom ation Engi ne	er";

		char[] chars = str1.toCharArray();

		StringBuffer sb = new StringBuffer();

		for (int i = 0; i < chars.length; i++)
		{
			if( (chars[i] != ' ') && (chars[i] != '\t') )
			{
				sb.append(chars[i]);
			}
		}
		System.out.println(sb);		//Output :
	}
}
```

Creation of Simple class and object

```
   public class Puppy {
            int puppyAge;

   public Puppy(String name) {
      // This constructor has one parameter, name.
      System.out.println("Name chosen is :" + name );
   }

   public void setAge( int age ) {
      puppyAge = age;
   }

   public int getAge( ) {
      System.out.println("Puppy's age is :" + puppyAge );
      return puppyAge;
   }

   public static void main(String []args) {
      /* Object creation */
      Puppy myPuppy = new Puppy( "tommy" );

      /* Call class method to set puppy's age */
      myPuppy.setAge(2);

      /* Call another class method to get puppy's age */
      myPuppy.getAge();

      /* You can access instance variable as follows as well */
      System.out.println("Variable Value :" + myPuppy.puppyAge);
   }
}
```

o/p

```
Name chosen is :tommy
Puppy's age is :2
```

```
Variable Value :2
```

Constructor in Java

n Java, a constructor is a block of codes similar to the method. It is called when an instance of the class is created. At the time of calling constructor, memory for the object is allocated in the memory

```
class Main {
  private int x;
    // constructor
    private Main(){
          System.out.println("Constructor Called");
          x = 5;
    }
    public static void main(String[] args){
          // constructor is called while creating object
          Main obj = new Main();
          System.out.println("Value of x = " + obj.x);
    }
}
```

Rules for creating Java constructor

There are two rules defined for the constructor.

1.Constructor name must be the same as its class name

2.A Constructor must have no explicit return type

3.A Java constructor cannot be abstract, static, final, and synchronized

Method overloading and method overriding

In Method Overloading, Methods of the same class shares the same name but each method must have a **different number of parameters or parameters having different types and order.**

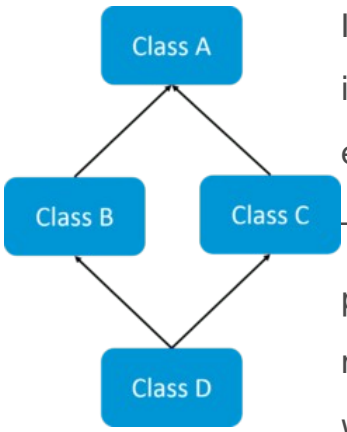
```
class Adder {

Static int add(int a, int b)
{
return a+b;
}
Static double add( double a, double b)
{
return a+b;
}
public static void main(String args[])
{
System.out.println(Adder.add(11,11));
System.out.println(Adder.add(12.3,12.6));
```

```
}
}
```

In Method Overriding, the subclass has the same method with the same name and **exactly the same number and type of parameters and same return type as a superclass.(it is used in inheritance)**

```
class Car {
        void run(){
                System.out.println(“car is running”);
                        }
Class Audi extends Car{
                void run()
                        {
                        System.out.prinltn("Audi is running safely with 100km");
                        }
                public static void main( String args[])
                      {
                       Car b=new Audi();
                       b.run();
                        }
                                }
                }
```

What is multiple inheritance? Why Is it not supported by Java?



If a child class inherits the property from multiple classes is known as multiple inheritance. Java does not allow to extend multiple classes.

The problem with multiple inheritance is that if multiple parent classes have the same method name, then at runtime it becomes difficult for the compiler to decide which method to execute from the child class.

Diamond problem : Class D has the property of Class A twice (A->B and A->C)

Therefore, Java doesn’t support multiple inheritance. The problem is commonly referred to as **Diamond Problem.**

Solution : Implementation of Interface in Java

Encapsulation in Java

- **Make the instance variables private** so that **they cannot be accessed directly from outside the class.** You can only set and get values of these variables through the methods of the class.

- Have **getter and setter methods in the class to set and get the values of the fields.**

```
class EncapsulationDemo{
    private int ssn;
    private String empName;
    private int empAge;

    //Getter and Setter methods
    public int getEmpSSN(){
        return ssn;
    }

    public String getEmpName(){
        return empName;
    }

    public int getEmpAge(){
        return empAge;
    }

    public void setEmpAge(int newValue){
        empAge = newValue;
    }

    public void setEmpName(String newValue){
        empName = newValue;
    }

    public void setEmpSSN(int newValue){
        ssn = newValue;
    }
}
public class EncapsTest{
    public static void main(String args[]){
         EncapsulationDemo obj = new EncapsulationDemo();
         obj.setEmpName("Mario");
         obj.setEmpAge(32);
         obj.setEmpSSN(112233);
         System.out.println("Employee Name: " + obj.getEmpName());
         System.out.println("Employee SSN: " + obj.getEmpSSN());
         System.out.println("Employee Age: " + obj.getEmpAge());
    }
}
```

Advantages of encapsulation

1.It improves maintainability and flexibility and re-usability:

Abstract class in Java

A class which is declared with the abstract keyword is known as an abstract class in Java. It can have abstract and non-abstract methods (method with the body).

Abstraction is a process of **hiding the implementation details** and **showing only functionality** to the user.

```
abstract class Animal {
                        abstract void makeSound();
                    }
```

```
class Dog extends Animal {
    public void makeSound() {
        System.out.println("Bark bark.");
    }
}

class Cat extends Animal {
    public void makeSound() {
        System.out.println("Meows ");
    }
}

class Main {
    public static void main(String[] args) {
        Dog d1 = new Dog();
        d1.makeSound();

        Cat c1 = new Cat();
        c1.makeSound();
    }
}
```

Output:

```
Bark bark
Meows
```

The `makeSound()` method cannot be implemented inside `Animal`. It is because every animal makes different sounds. So, all the subclasses of `Animal` would have different implementation of `makeSound()`.

We cannot implement `makeSound()` in `Animal` in such a way that it can be correct for all subclasses of `Animal`. So, the implementation of `makeSound()` in `Animal` is kept hidden.

Interface

```
interface  Polygon {

    void getArea();

  // calculate the perimeter of a Polygon
    default void getPerimeter(int... sides) {
        int perimeter = 0;
        for (int side: sides) {
            perimeter += side;
        }

    System.out.println("Perimeter: " + perimeter);
    }
}

class Triangle implements Polygon {
    private int a, b, c;
    private double s, area;

// initializing sides of a triangle
    Triangle(int a, int b, int c) {
        this.a = a;
        this.b = b;
        this.c = c;
        s = 0;
    }
// calculate the area of a triangle
    public void getArea() {
        s = (double) (a + b + c)/2;
        area = Math.sqrt(s*(s-a)*(s-b)*(s-c));
        System.out.println("Area: " + area);
    }
}
class Main {
    public static void main(String[] args) {
        Triangle t1 = new Triangle(2, 3, 4);

// calls the method of the Triangle class
        t1.getArea();

// calls the method of Polygon
        t1.getPerimeter(2, 3, 4);
    }
}
```

```
//Area Of Rectangle and Triangle using Interface

interface Area
{
float compute(float x, float y);
}

class Rectang implements Area
{
public float compute(float x, float y)
{
return(x * y);
}
}

class Triang implements Area
{
public float compute(float x,float y)
{
return(x * y/2);
}
}

class Multinheritance
{
public static void main(String args[])
{
Rectang rect = new Rectang();
Triang tri = new Triang();
Area area;
area = rect;
System.out.println("Area Of Rectangle = "+ area.compute(1,2));

area = tri;
System.out.println("Area Of Triangle = "+ area.compute(10,2));
}
}
/* * OUTPUT **

Area Of Rectangle = 2.0
Area Of Triangle = 10.0
*/
```

Inheritance

```
class Teacher {
   String designation = "Teacher";
   String collegeName = "Beginnersbook";
   void does(){
        System.out.println("Teaching");
   }
}

public class PhysicsTeacher extends Teacher{
   String mainSubject = "Physics";
   public static void main(String args[]){
        PhysicsTeacher obj = new PhysicsTeacher();
        System.out.println(obj.collegeName);
        System.out.println(obj.designation);
        System.out.println(obj.mainSubject);
        obj.does();
   }
}
```