1

public static void main (String args[])

{

System.out.println (100+100+”Hello”);

System.out.println (“Hello World”+ 100 +100);

}

Output:

200Hello

Hello World100100

2

Explain **JVM, JRE & JDK**

JVM – Java Virtual Machine

JRE – Java Runtime Environment (for executing programs)

JDK – Java Development Kit (for debugging & development)

3

Explain **public static void main(String args[])**

4

What are the various access identifiers?

Access Specifiers are predefined keywords used to help JVM understand the scope of a variable, method, and class. We have four access specifiers.

* Public Access Specifier
* Private Access Specifier
* Protected Access Specifier
* Default Access Specifier

Public, Private, Protected, Default

5

### How many types of constructors are used in Java?

There are two [types of constructors in Java](https://www.simplilearn.com/tutorials/java-tutorial/constructor-in-java).

Parameterized Constructors: Parameterized constructor accepts the parameters with which users can initialize the instance variables. Users can initialize the class variables dynamically at the time of instantiating the class.

Default constructors: This type doesn’t accept any parameters; rather, it instantiates the class variables with their default values. It is used mainly for object creation.

Types: Default, Parameterized, Copy

6

Differentiate b/w method overloading & method over-riding. With example.

Differentiate b/w method & constructor overloading

The process of creating multiple method signatures using one method name is called Method Overloading in Java. Two ways to achieve method overloading are:

1. Varying the number of arguments
2. Changing the return type of the Method

|  |  |
| --- | --- |
| Overloading | Overriding |
| Two or more methods having the same name but different parameters or signature | Child class redefining methods present in the base class with the same parameters/ signature |
| Resolved during compile-time | Resolved during runtime |

Will the program run if we write static public void main?

Yes, the program will successfully execute if written so. Because, in Java, there is no specific rule for the order of specifiers

### Differentiate between instance and local variables.

For instance, variables are declared inside a class, and the [scope of variables in javascript](https://www.simplilearn.com/tutorials/javascript-tutorial/scope-of-variables-in-javascript) is limited to only a specific object.

A local variable can be anywhere inside a method or a specific block of code. Also, the scope is limited to the code segment where the variable is declared.

Write a Java Program to check Prime number.

public class Main {

public static void main(String[] args) {

int n = 11;

if (n <= 1) {

System.out.println("The number is not prime");

return;

}

int count = 0;

int i = 1;

while (i <= n / 2) {

if (n % i == 0) {

count++;

}

i++;

}

if (count > 1) {

System.out.println("The number is not prime");

} else {

System.out.println("The number is prime");

}

}

}

## Find Prime Number Using Recursion

class Main {

static boolean isPrime(int n, int i) {

if (n <= 2) return (n == 2) ? true : false;

if (n % i == 0) return false;

if (i \* i > n) return true;

return isPrime(n, i + 1);

}

public static void main(String[] args) {

int n = 11;

if (isPrime(n, 2)) {

System.out.println("The number is prime");

}

else {

System.out.println("The number is not prime");

}

}

}

### Write a Java Program to print Fibonacci Series using Recursion.

class FibonacciExample2{

 static int n1=0,n2=1,n3=0;

 static void printFibonacci(int count){

    if(count>0){

         n3 = n1 + n2;

         n1 = n2;

         n2 = n3;

         System.out.print(" "+n3);

         printFibonacci(count-1);

     }

 }

 public static void main(String args[]){

  int count=10;

  System.out.print(n1+" "+n2);//printing 0 and 1

  printFibonacci(count-2);//n-2 because 2 numbers are already printed

 }

 }

#### **What is an abstract class?**

An abstract class is a class that consists of abstract methods. These methods are basically declared but not defined. If these methods are to be used in some subclass, they need to be exclusively defined in the subclass.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

### ****What is exception handling?****

Exception handling in Object-Oriented Programming is a very important concept that is used to manage errors. An exception handler allows errors to be thrown and caught and implements a centralized mechanism to resolve them.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**What are the main features of OOPs?**

* Inheritance
* Encapsulation
* Polymorphism
* Data Abstraction

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_