## What is a constructor used for?

A constructor in Java is a **special method** that is used to initialize objects. The constructor is called when an object of a class is created. It can be used to set initial values for object attributes:

Example

Create a constructor:

// Create a MyClass class

public class MyClass {

int x; // Create a class attribute

// Create a **class constructor** for the MyClass class

public MyClass() {

x = 5; // Set the initial value for the class attribute x

}

public static void main(String[] args) {

MyClass myObj = new MyClass(); // Create an object of class MyClass (This will **call the constructor**)

System.out.println(myObj.x); // Print the value of x

}

}

// Outputs 5

## Constructor Parameters

Constructors can also take parameters, which is used to initialize attributes.

The following example adds an int y parameter to the constructor. Inside the constructor we set x to y (x=y). When we call the constructor, we pass a parameter to the constructor (5), which will set the value of x to 5:

### Example

public class MyClass {

int x;

public MyClass(int y) {

x = y;

}

public static void main(String[] args) {

MyClass myObj = new MyClass(5);

System.out.println(myObj.x);

}

}

// Outputs 5

## How do constructors differ from other methods in a class?

The important **difference between constructors and methods** is that **constructors** initialize objects that are being created with the new operator, while **methods** perform operations on objects that already exist. **Constructors** can't be called directly; they are called implicitly when the new keyword creates an object

## Describe the difference between private and public

## public means you can access it anywhere while private means you can only access it inside its own class. Just to note all private, protected or public modifier are not applicable to local variables in Java. a local variable can only be final in java.

## how a programmer decides which parts of a class should be private and which public.?

## All your implementation details should be private, and the public part should be a small, concise, well-defined interface for using your class. The keyword here is Encapsulation. In OOP you want to use private variables in order to enforce proper encapsulation of your objects/classes.

## Part 2

## My program UML class diagram

## 