What is a Constructor?

As the name suggests, the constructor is used to *construct* the object of a class. It is a special method that outlines the steps that are performed when an instance of a class is created in the program.

> A constructor's **name** must be exactly the **same** as the name of its class.

The constructor is a special method because it **does not have a return type**. We do not even need to write `void` as the return type. It is a good practice to declare/define it as the first member method.

So, let's study the different types of constructors and use them to create class objects.

Default Constructor

The default constructor is the most basic form of a constructor. Think of it this way:

> In a default constructor, we define the default values for the data members of the class. Hence, the constructor creates an object in which the data members are initialized to their default values.

This will make sense when we look at the code below. Here, we have a **Date** class, with its default constructor, and we'll create an object out of it in our `main()`:

```
class Date {

  private int day;
  private int month;
  private int year;


  // Default constructor
  public Date() {
    // We must define the default values for day, month, and year
    day = 0;
    month = 0;
    year = 0;
  }

  // A simple print function
  public void printDate(){
    System.out.println("Date: " + day + "/" + month + "/" + year);
  }

}

class Demo {

  public static void main(String args[]) {
    // Call the Date constructor to create its object;
    Date date = new Date(); // Object created with default values!
    date.printDate();
```

Run Save Reset

Notice that when we created a `Date` object in **line 26**, we don't treat the constructor as a method and write this:

```
date.Date()
```

We create the object just like we create an `integer` or `string` object. It's that easy!

The default constructor does not need to be explicitly defined. Even if we don't create it, the JVM will call a default constructor and set data members to `null` or `0`.

> If you don't define any constructor, the Java compiler will insert a default constructor for you. Thus, once the class is compiled it will always at least have a no-argument constructor.

Parameterized Constructor

The default constructor isn't all that impressive. Sure, we could use `set` methods to set the values for `day`, `month` and `year` ourselves, but this step can be avoided using a **parameterized constructor**.

> In a parameterized constructor, we pass arguments to the constructor and set them as the values of our data members.

We are basically overloading the default constructor to accommodate our preferred values for the data members.

Let's try it out:

```
class Date {

  private int day;
  private int month;
  private int year;


  // Default constructor
  public Date() {
    // We must define the default values for day, month, and year
    day = 0;
    month = 0;
    year = 0;
  }

  // Parameterized constructor
  public Date(int d, int m, int y){
    // The arguments are used as values
    day = d;
    month = m;
    year = y;
  }

  // A simple print function
  public void printDate(){
    System.out.println("Date: " + day + "/" + month + "/" + year);
  }
}
```

Run Save Reset

This is much more convenient than the default constructor!