# Java: Constructors

## Quick summary

- constructors construct objects from classes
- use `new` keyword: Class object_name = new Class();
- have same name as class and look just like method
- usually the constructor initializes the object with specific arguments
- constructors don’t have a return type
- a class can have multiple constructors, differentiated by arguments
- best practice is for classes to always have a constructor, even if no arguments

## Sample Java projects

- constructor_sample1
- constructor_sample2

## About constructors

Constructors are used to create new objects from classes. Constructors use the same name as the class. When you create a new object from a constructor, you usually pass in arguments to set values for some of the class’s instance variables.

Constructors have the following characteristics:

- Constructors don't have a return type (e.g., `void`).
- Constructors have same name as the class.
- Constructors just like a method in its format, except that it has the same name as the class.
- Constructors can take arguments, like methods.
- It's a best practice to include a constructor with each class.
- When there are multiple constructors for the same class (they will aloo have the same name), the arguments specified for each constructor become the distinguishing point. The constructor that is used is the one whose parameters match the incoming arguments.

## Sample constructor

Here's how you create a constructor:

```
public class MyConstructor {



public MyConstructor() {

System.out.println("constructor is running!");

}



}
```

Then in your `main` method, you call the constructor as follows:

```
public class App {



public static void main(String[] args) {

MyConstructor lilconstructor1 = new MyConstructor();



}



}
```

You can even use shorthand to call the constructor, like this:

```
public class App {



public static void main(String[] args) {



// shorthand way of calling constructor

new MyConstructor();



}



}
```

(One thing I'm not sure about is what the name of the object is when you use the constructor this way.)

Constructors can take arguments:

```
public class MyConstructor {
```

```
public String name;


public MyConstructor(String name) {

this.name = name;

System.out.println("My name is " + name");

}


}
```

The `this.name = name` is important. It sets the value of the instance variable `name` to that of the constructor you just created.

You can then pass in the argument to the constructor like this:

```
package constructors;


public class App {


public static void main(String[] args) {

// longhand way of calling constructor

MyConstructor lilconstructor1 = new MyConstructor("Henry");


// shorthand way of calling constructor

//new MyConstructor("Henry");


}
```

```
}
```

You can identify constructors by their arguments. Suppose you have 2 constructors for your class, as follows:

```
public class MyConstructor {


public MyConstructor() {

System.out.println("constructor1 is running!");

}


public MyConstructor(String name) {

System.out.println("constructor2 is running!" + name);

}


}
```

If you pass a string argument when you call the constructor, the `MyConstructor2` will run instead of `MyConstructor1`. Depending on the parameters you're passing, Java will choose to run the correct constructor.

The syntax for instantiating a class is the same as calling a constructor. Even if your class doesn't have a constructor, the default constructor is made using the `new` keyword.

Person john = new Person();

The constructor for an object is simply the word `new` before the class name. You are actually “constructing” an object using the constructor when you instantiate a class. However, we often just say “creating an object” or “instantiating the class” instead of using the constructor to create an object.

You can have multiple constructors (overloading). The constructor that runs is determined by the arguments that match the constructor’s parameters.

it’s a best practice to always use a constructor with a class even if it doesn’t take an argument.

## Default constructors

All classes rely on constructors in order to create objects. If a class doesn’t have a constructor, it uses the `new` keyword in order to create the object.

## No-argument constructors

If a constructor doesn’t take any arguments, it’s considered a “no-argument constructor.” Usually when constructors take arguments, they initialize the new object with certain values.

## Constructor methods

Constructors are really methods. By default, it’s `new ClassName()`;. But you can create your own:

```
public class Olive {



public Olive() {

System.out.println("Constructor of " + this.name)
```

```
}


}
```

Constructor methods never return a value. The accessor method should be public. The constructor has the same name as the class.