

Theory: Constructor

🕒 15 minutes 0 / 5 problems solved

Skip this topic

Start practicing

8469 users solved this topic. Latest completion was 14 minutes ago.

Constructors are special methods that initialize a **new object** of the class. A constructor of a class is invoked when an instance is created using the keyword `new`.

A constructor is different from other methods in that:

- it has the same name as the class that contains it;
- it has no return type (not even `void`).

Constructors initialize **instances** (objects) of the class. They set values to the fields when the object is created. Also, constructors can take parameters for initializing fields by the given values.

§1. Using constructors

Here is a class named `Patient`. An object of the class has a name, an age, and a height. The class has a three-argument constructor to initialize objects with specific values.

```
1 class Patient {
2
3     String name;
4     int age;
5     float height;
6
7     public Patient(String name, int age, float height) {
8         this.name = name;
9         this.age = age;
10
11         this.height = height;
12     }
13 }
```

Let's go further and create some instances of the class using the constructor we've written:

```
1 Patient patient1 = new Patient("Heinrich", 40, 182.0f);
2 Patient patient2 = new Patient("Mary", 33, 171.5f);
```

Now we have two patients, Heinrich and Mary, with the same fields, but the values of those fields are different.

§2. Keyword this

In the example above, `Patient` constructor takes three parameters:

```
1 this.name = name;
2 this.age = age;
3 this.height = height;
```

To initialize the fields, the keyword `this` is used, which is a reference to the current object. Usually, `this` keyword is used when an instance variable and a constructor or a method variable share the same name. This keyword helps to disambiguate these instances.

If you write something like `name = name`, it means that you're assigning the `name` variable to itself, that, of course, doesn't make any sense. Frankly speaking, you may distinguish two objects simply by assigning another name to the variable, like `name = newName`. It is not prohibited, but it is considered bad practice since these variables point to the same thing. These are the

Current topic:

`Constructor` Stage 6 ...

Topic depends on:

✓ `Declaring a method` Stage 3 ...

✓ `Defining classes` Stage 3 ...

Topic is required for:

`Access modifiers` Stage 6 ...

`Static members` ...

`Multiple constructors` ...

`Fields and methods in enum` ...

`Inheritance` Stage 7 ...

`Generic programming` ...

`Class files and Bytecode` ...

`Introduction to JPA` ...

Table of contents:

[↑ Constructor](#)

[§1. Using constructors](#)

[§2. Keyword this](#)

[§3. Default and no-argument constructor](#)

[§4. To sum up](#)

[Feedback & Comments](#)

reasons why `this` keyword is extremely useful to work with constructors, fields and methods. The absence of extra variables makes the code look clearer and less overloaded.

§3. Default and no-argument constructor

The compiler automatically provides a default no-argument constructor for any class without constructors.

```
1 class Patient {
2
3     String name;
4     int age;
5     float height;
6 }
```

We can create an instance of the class `Patient` using the no-argument default constructor:

```
1 Patient patient = new Patient();
```

In this case, all fields will be filled with the default values of their types.

If you define a specific constructor, the default constructor will not be created.

We can also define a constructor without any arguments, but use it to set default values for fields of a class. For example, we can initialize `name` with `"Unknown"`:

```
1 class Patient {
2
3     String name;
4     int age;
5     float height;
6
7     public Patient() {
8         this.name = "Unknown";
9     }
10 }
```

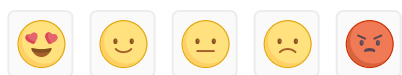
Such no-argument constructors are useful in cases when any default value is better than `null`.

§4. To sum up

- Any Java class has a constructor to initialize objects;
- A constructor has the same name as the class containing it;
- A constructor has no return type, not even `void`;
- If a class has no explicit constructors, the Java compiler automatically provides a default no-argument constructor;
- If we want to introduce new variables to denote the same thing, make the code clearer and less loaded with extra variables, the keyword `this` is used.

 Report a typo

662 users liked this theory. 10 didn't like it. What about you?



Start practicing

[Comments \(15\)](#)

[Hints \(0\)](#)

[Useful links \(0\)](#)

[Show discussion](#)

