



# Classes and Objects

---

**2023-2024**

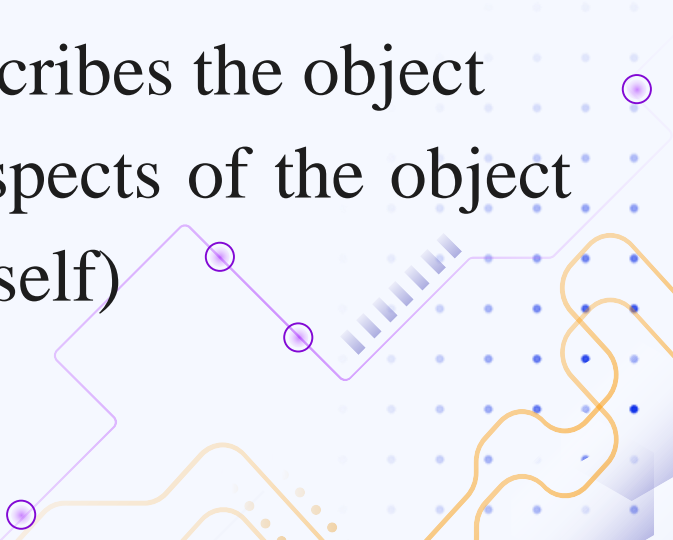


**Object** : an encapsulation of data along with functions that act upon that data.

**An object consists of:**

- ❑ **Name:** the variable name we give it
  - ❑ **Member data :** the data that describes the object

---

  - ❑ **Member functions:** behavior aspects of the object  
(functions related to the object itself)
- 

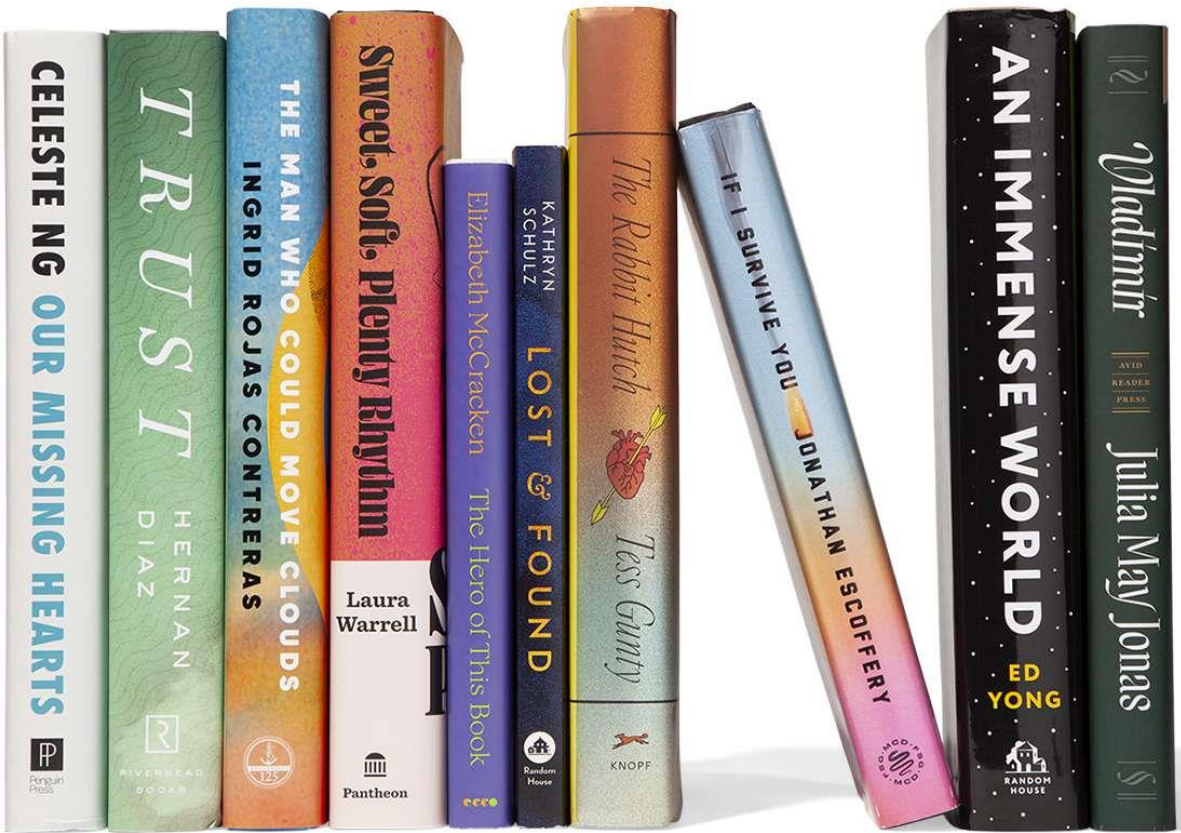
**Class:** a blueprint for objects. A class is a user-defined type that describes what a certain type of object will look like.

A class description consists of a declaration and a definition. Usually these pieces are split into separate files.

---

An object is a single instance of a class. You can create many objects from the same class type.







**Objects** are omnipresent in our world (human, book, etc.). An object is a symbolic, self-contained container that contains information and mechanisms about a subject, manipulated in a program. The subject is often something tangible (concrete)

**In OOP: object = identity + attributes + behaviors**

**The identity** must make it possible to unambiguously identify the object (address/reference or name).

**attributes** are stored in properties (variables) and

**Behaviors** are implemented using **methods** (procedures/functions).

# Attributes

**Attributes:** These are data characterizing the object. These are variables storing state information about the object,

**Example:** the title of a *book*, for the *Book class*.

These variables (attributes) can be primitive types (int, float...) or objects. They are declared like other variables.

Example: for the Book class

```
int nbPages; // number of pages
```

```
String Title; // Title of the book
```



# Behavior

**Behavior:** is characterized by the methods, that is to say the set of actions (called operations) that the object is able to carry out. These operations make it possible to make the object react to external requests (or to act on other objects). In addition, operations are closely linked to attributes, because their actions can depend on the attribute values, or modify them

**In Java,** the behavior of a class of objects is defined by what we call methods, blocks of instructions that can be compared to functions in other languages.

# Relation Object-Class

the object becomes an instance of its class, and of the class as well as of the type of this object. Each of the object's attributes will be "typed" as indicated in its class, and all methods affecting the object will be only those provided for in its class.

Each instance (object) of a class has its own set of attributes in memory independent of other objects of this class.

# Constructor

**A constructor** is a method responsible for constructing the object (allocating memory and initializing the object's attributes). This constructor is never called directly; it is taken into account when requesting creation of the object with new:

**Class Object =new Class()** ●

Example:

**Rectangle rect1 =new Rectangle()**

# Constructor creation

We can create the constructor method as follows:

- ✓ A constructor must always be a public method
- ✓ You must give the same name as your class to the constructor
- ✓ A constructor does not return anything, that is to say you will **not add a return** in this method.
- ✓ ~~You should not put *void*, even if the constructor returns nothing.~~

## Example: Rectangle class

```
public class Rectangle {  
    float longr;  
    float larg;  
    public float surface(){  
  
        return longr*larg;  
    }  
    public float perimetre(){  
        return (longr+larg)*2;  
    }  
    public void afficher(){  
        System.out.println("longueur= "+longr+", largeur="+ larg);  
    }  
}
```

## Example: constructors

```
public class Rectangle {  
    float longr;  
    float larg;  
  
    public Rectangle(){  
  
    }  
    public Rectangle(float longr, float larg){  
        this.longr=longr;  
        this.larg= larg;  
    }  
}
```