Classes and Objects

- The concept of an object:
  - Objects are the instances created from the class.
  - An object of a class is also called an instance of that class.
  - Once we have the class, we can create as many objects (instances) as we want.

Classes and Objects

- The concept of an object:
  - These objects will have:
    - Different identities.
    - States defined by the same attributes but with values that can be different.
    - Identical behaviour (same methods).

Classes and Objects

- **Class = Mold = Model**
  - **Describes the structure of the state (attributes and their types)**
  - **defines the behaviour of the object (its methods) and its interface (messages accepted by the object)**
- **Instance = Casting = Object**
  - **Has a state (attribute values) that corresponds to the structure described by the class**
  - **Only responds to messages allowed by the class (interface)**

Classes and Objects

- **class : abstract**
  - There is no student called "student"
- **instance : concrete**
  - The student “Halimi Amina born on March 17, 1990”

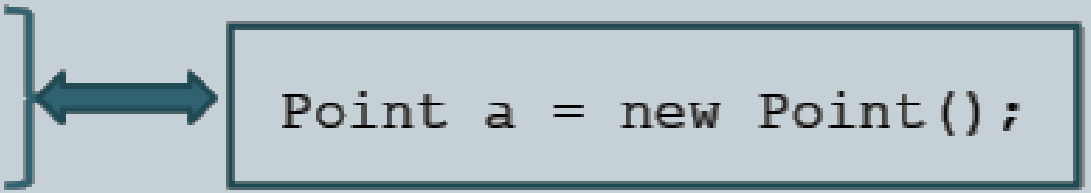
Classes and Objects

- Creation of an object:
  - To create a new object, we need to allocate memory space for it.
    - Each object will have its space where the values of its attributes will be stored.
    - If we create 1000 objects of type Point, we will have 1000 locations for the variable x and 1000 locations for the variable y.
    - The methods, on the other hand, are not duplicated (unnecessary).

Classes and Objects

- **Example: the Point class**
  - **JAVA**

```
Point a;
a = new Point ();
```



Classes and Objects

- **Example: the Point class**
  - To create a space for the object itself, the keyword "new" must be used.

```
a = new Point(2, 5);
```

  - In this statement, the "new" operator creates an object of type Point and provides its reference.
  - This reference is assigned to the previously declared variable "a."
  - Additionally, it initializes the attributes x and y to the values 2 and 5, respectively.

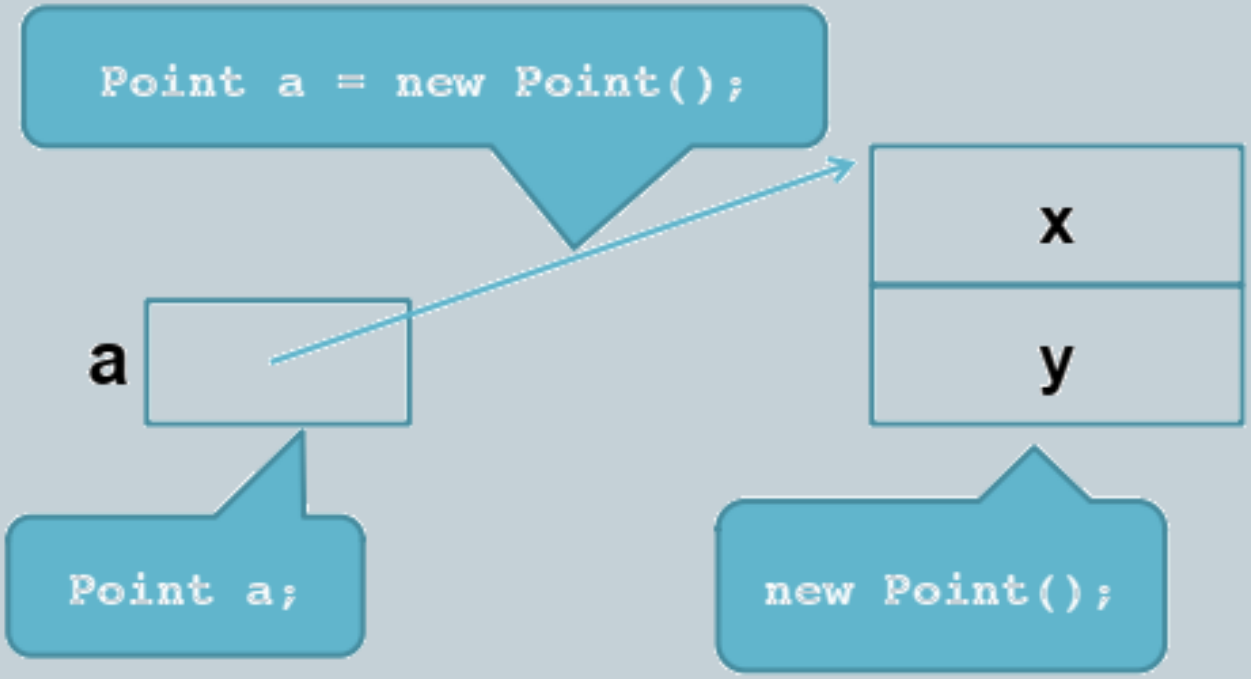
Classes and Objects

- **Example: the Point class**
  - **We can create objects of the Point class and apply its public methods init, move, and display as needed.**
  - **Before creating an object, it must first be declared.**

    Point a;
  - **This declaration does not reserve space for an object of type Point but only space for a reference to an object of type Point.**

Classes and Objects

- **Example of the Point class**



Classes and Objects

- The concept of a constructor
  - Point class with a constructor
    - The general template for defining a class in Java is

```
class Point
{
// field definition instructions
private int x; // abscissa
private int y; // ordered

// instructions for definition of class methods

public Point ( int absc , int ord )
{x = absc ;
y = ord ;
}

public void move ( int dx , int dy )
{x = x + dx ;
y = y + dy ;
}

public void display ()
{
System. out.println ("I am a point with coordinates :"+x+" "+y);
}
}
```

This is the constructor for the Point method. Now the init method becomes useless. We can delete it

Classes and Objects

- **Definition:**
  - A constructor is a special method used to construct a new object.
  - The object will adhere to the structure defined in the class, meaning it will have the attributes and behaviour defined in the class. This requires reserving memory space to store the state.
  - The constructor is often used to initialize the different attributes of the object.
    - **It has the same name as the class.**
    - **It does not return any value.**
    - **It has no return type, not even void.**

Classes and Objects

- **Some rules:**

1. **A class may not have any constructor. In this case, objects are created using a default constructor without arguments, as in the statement `a = new Point();`.**

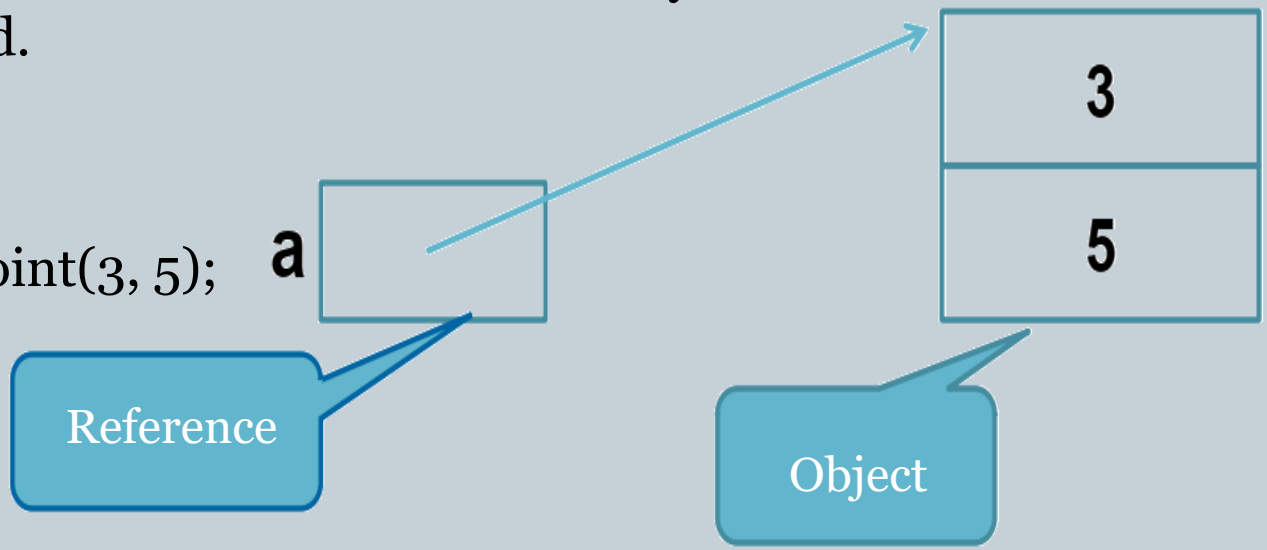
2. **A class may have multiple constructors.**

3. **Once a class has at least one constructor, the default constructor can no longer be used. Unless a constructor without arguments has been defined.**

Classes and Objects

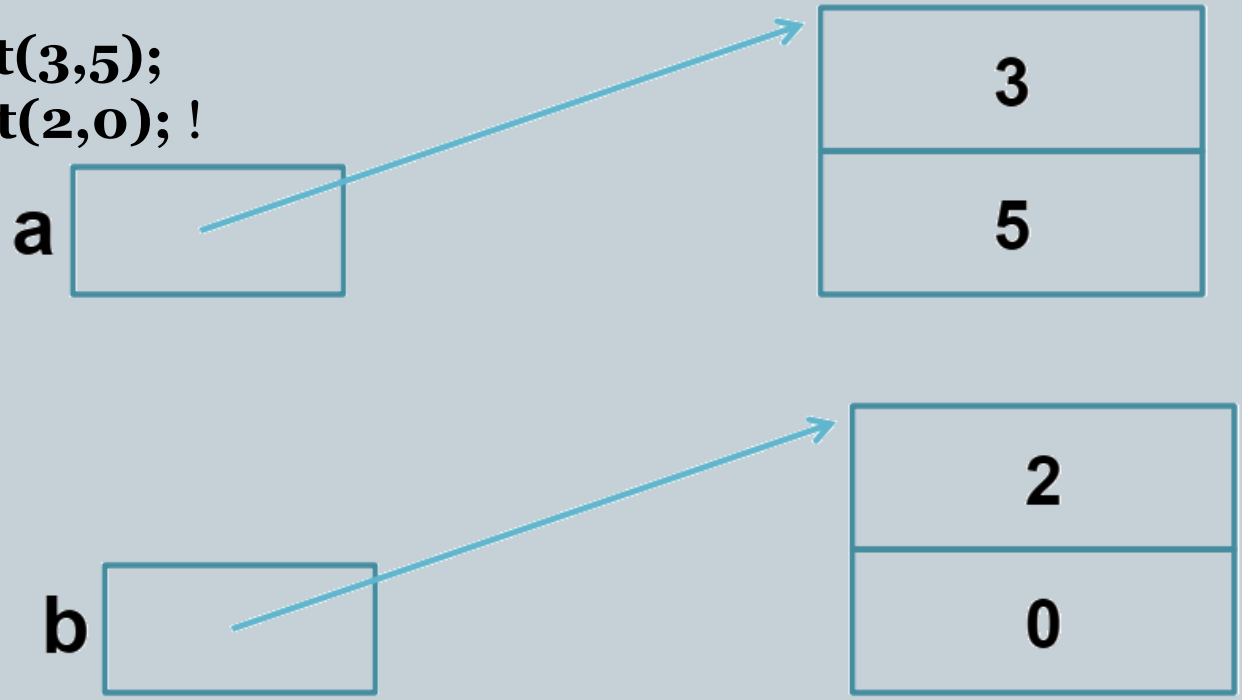
- The concept of reference:
  - To use objects, they need to be referenced.
  - The reference allows access to the object but is not the object itself. It contains the address of the memory location where the object is stored.
  - Example:

Point a = new Point(3, 5);



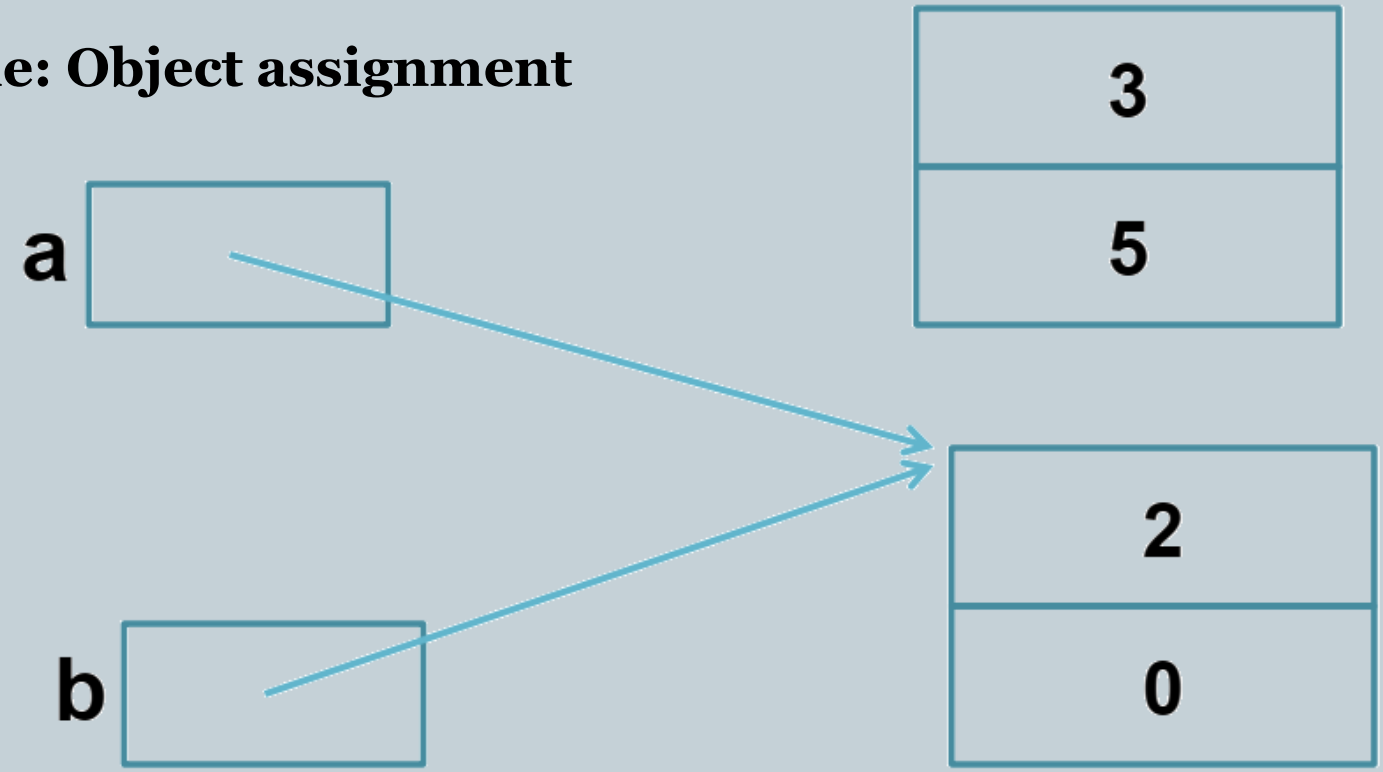
Classes and Objects

- The concept of reference:
  - **Example**
    Point a,b ;
    a = **new Point(3,5);**
    b = **new Point(2,0);** !



Classes and Objects

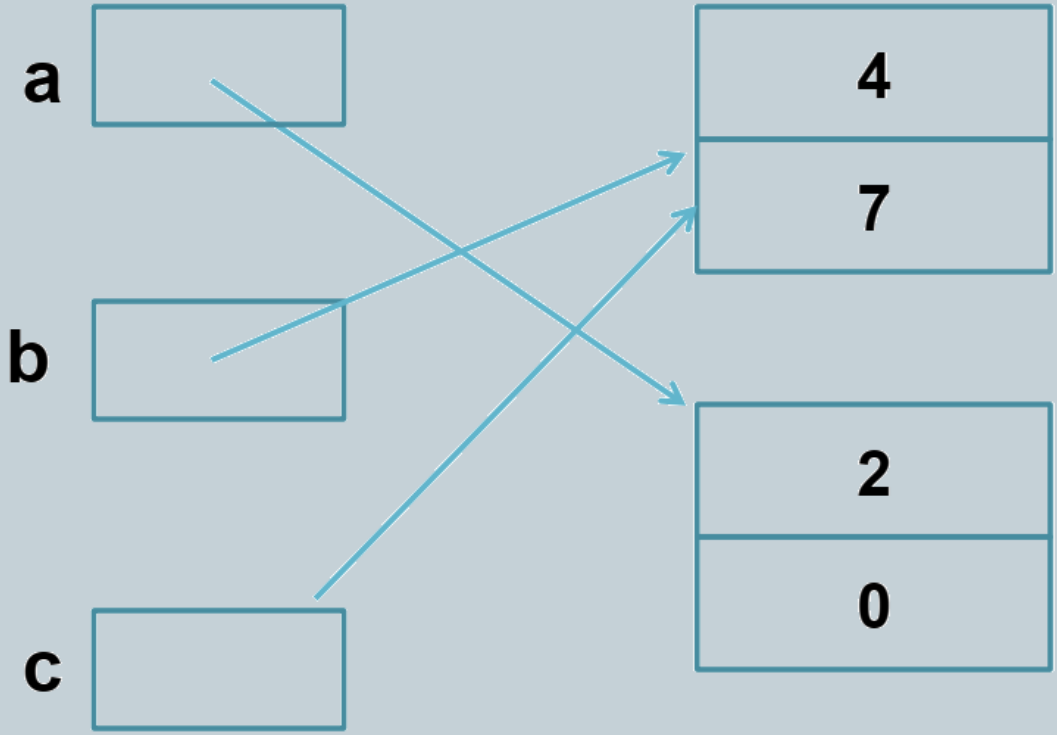
- The concept of reference:
  - **Example: Object assignment**
    a = b;



Classes and Objects

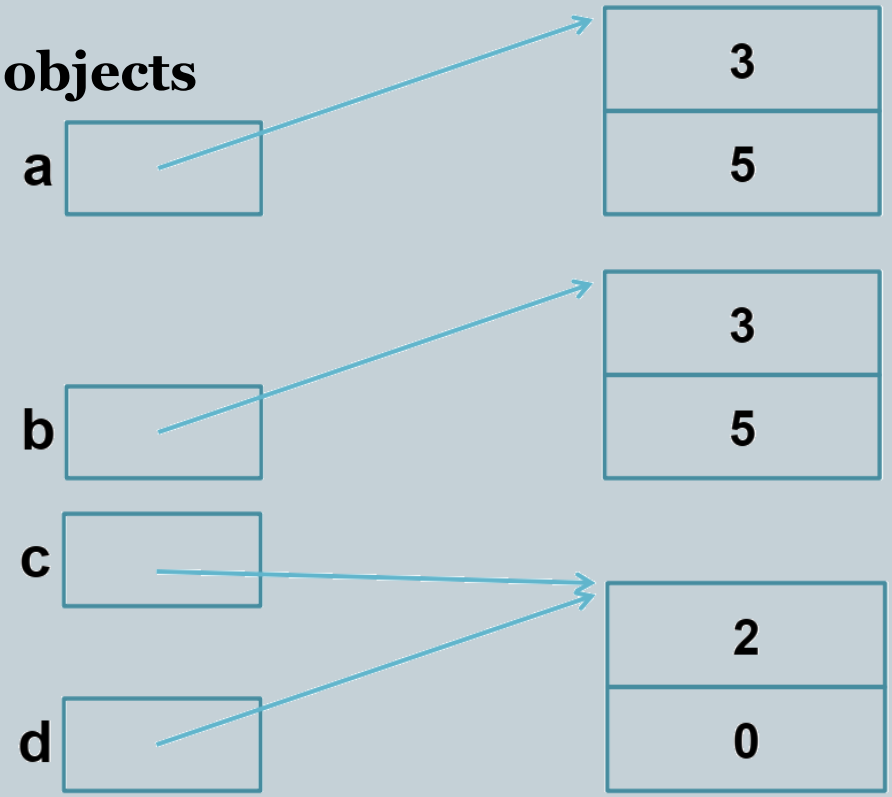
- The concept of reference:
  - **Example : Comparison of objects**

```
Point a, b, c;
a = new Point(4,7);
b = new Point(2,0);
c = a;
a = b;
b = c;
```



Classes and Objects

- The concept of reference:
  - **Example : Comparison of objects**
    a == b?
    c == d ?



Classes and Objects

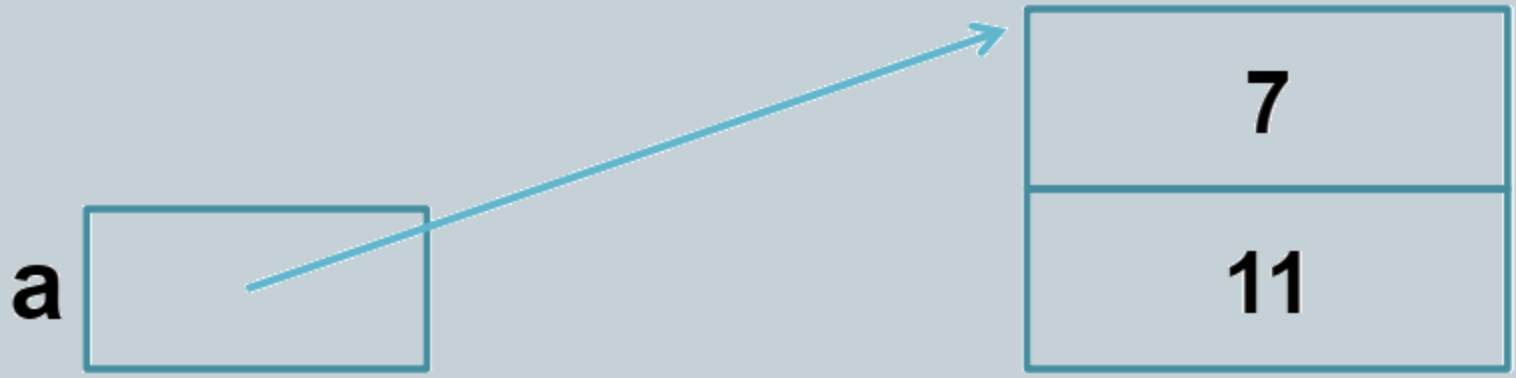
- Self-reference:
  - In the execution of one of its methods, an object may need to send itself a message (to access one of its attributes or invoke one of its methods).
    - For this, it uses the keyword `this`.
    - The use of the keyword `this` also helps to avoid ambiguity.

Example: the Point constructor

```
public Point(int x, int y) {
    this.x = x;
    this.y = y;
}
```

Classes and Objects

- Utilization of an object:
  - Once the object is created, any method can be applied to it.
    - Example: a.move(4, 6);
    - (suppose that `a` is already initialized with the value 3 in the x field and the value 5 in the y field)



Classes and Objects

- Creation and use of an object
  - Example: Circle class

Let's imagine that we want to create a Circle class to represent circles defined by a centre (an object of type Point) and a floating-point radius.

Circle Class
centre radius
Move() Display()

Point Class
X Y
Init() Move() Display()

  - The functionalities of this class are limited to:
    - Displaying the characteristics of a circle (coordinates of the center point and the radius)
    - Moving the center

Classes and Objects

- Creating and use of an object
  - **Example class circle : code in java**

```
public class Circle

 { private Point centre;
   private float radius;

public Circle ( int x, int y, float r)
   { centre = new Point (x, y);
   radius = r;
   }

public void display ()
   { // display the center and radius of the circle
   }
   }
```

Classes and Objects

- **Some special methods:**
  - **Among the methods that a class may have, we distinguish:**
    - **Constructors**
      - **Accessor methods (Getter): provide the values of certain private fields without modifying them.**
        - **Often, names of the form `getXXX` are used.**
      - **Mutator methods (Setter): modify the values of certain private fields.**
        - **Often, names of the form `setXXX` are used.**

Classes and Objects

- Some design rules:
  - For good design, there are some rules that are not mandatory but highly recommended to follow.
    1. Respect the principle of encapsulation by declaring all fields as private.
    2. Rely on the notion of a contract, which considers that a class is characterized by:
       A. The headers of its public methods (interface)
       B. The behaviours of these methods.
       C. The rest (fields, private methods, and body) is called the implementation and should remain private.
       D. The contract defines what the class does, while the implementation describes how it does it.

Classes and Objects

- The Garbage Collector
  - To create an object, a Java program uses the new operator.
  - To free up the memory occupied by the object, a destructor is invoked (finalizers in Java).
  - Java ensures automatic memory management through the Garbage Collector, which operates on the following principle:
    - At any moment, we know the number of references to an object.
    - When an object is no longer referenced (there are no references to it), we are certain that the program will no longer be able to access it. It is then possible to free up the corresponding space.