OBJECT ORIENTED PROGRAMMING
UNIT2: OO PROGRAMMING. OBJECTS

JESUITAS
Sagrado Corazón
Jesuitas · Logroño

Index

Introduction to OOP

- Before OOP, computer programs were usually described under the simplest definition you've learned: *sets of instructions that are listed in a file and handled in some kind of reliable order* (structured/procedural programming).

Introduction to OOP

- The programs you create with Java can be thought of as objects, just like physical objects that exist in the real world.
- Fundamental concepts:
  - object
  - class
  - method
  - parameter
  - data type

How Object-Oriented Programming Works

- With this new paradigm, you can think of a computer program as a group of objects that interact with each other:
  - Objects
    - represent 'things' from the real world, or from some problem domain (example: "the red car down there in the car park")
  - Classes
    - represent all objects of a kind (example: "car")

Let´s try to imagine what classes are…

How Object-Oriented Programming Works

- In object-oriented programming, an object contains two things: *attributes* and *behavior*.
  - Attributes (fields) are things that describe the object and show how it is different from other objects. Also called *data* or *properties*.
    - The class defines what fields an object has, but each object stores its own set of values (the *state* of the object).
  - Objects have operations (behavior) which can be invoked (Java calls them methods).
    - Methods may have parameters to pass additional information needed to execute.

How Object-Oriented Programming Works

- You create objects in Java by using a class as a "template" that represents something in the reality.
- A class is a master copy of the object that is consulted to determine which attributes and behavior an object should have.

How Object-Oriented Programming Works

- Examples:
  - We can think of a class called *bird*.
    - A bird can have these attributes:
      - Name
      - Color of their plumes
      - Age
      - If they are domestic pets or not
    - Behaviour
      - Hunt
      - Sing
  - *My parrot called "Dorian" (identity), will own to the class bird, which is green, of age 2 and domestic (state) → Dorian will be an object (instance) of the class bird, which will be able to Hunt and Breed (behaviour).*

How Object-Oriented Programming Works

- Other examples of an object:
  - You have already used *String* object type, in order to have character chains ("Hello world").
  - Every time you declare a *String* type, you are creating an object of the *String class*.
  - This class contains attributes that determine what a String object is and behavior that controls what String objects can do.
- With object-oriented programming, a computer program is a group of objects that work together to get something done.

How Object-Oriented Programming Works

- When you write an application OO, what you are doing is defining the classes (with attributes and behaviour), and when you run the program objects (*instances*) are created.

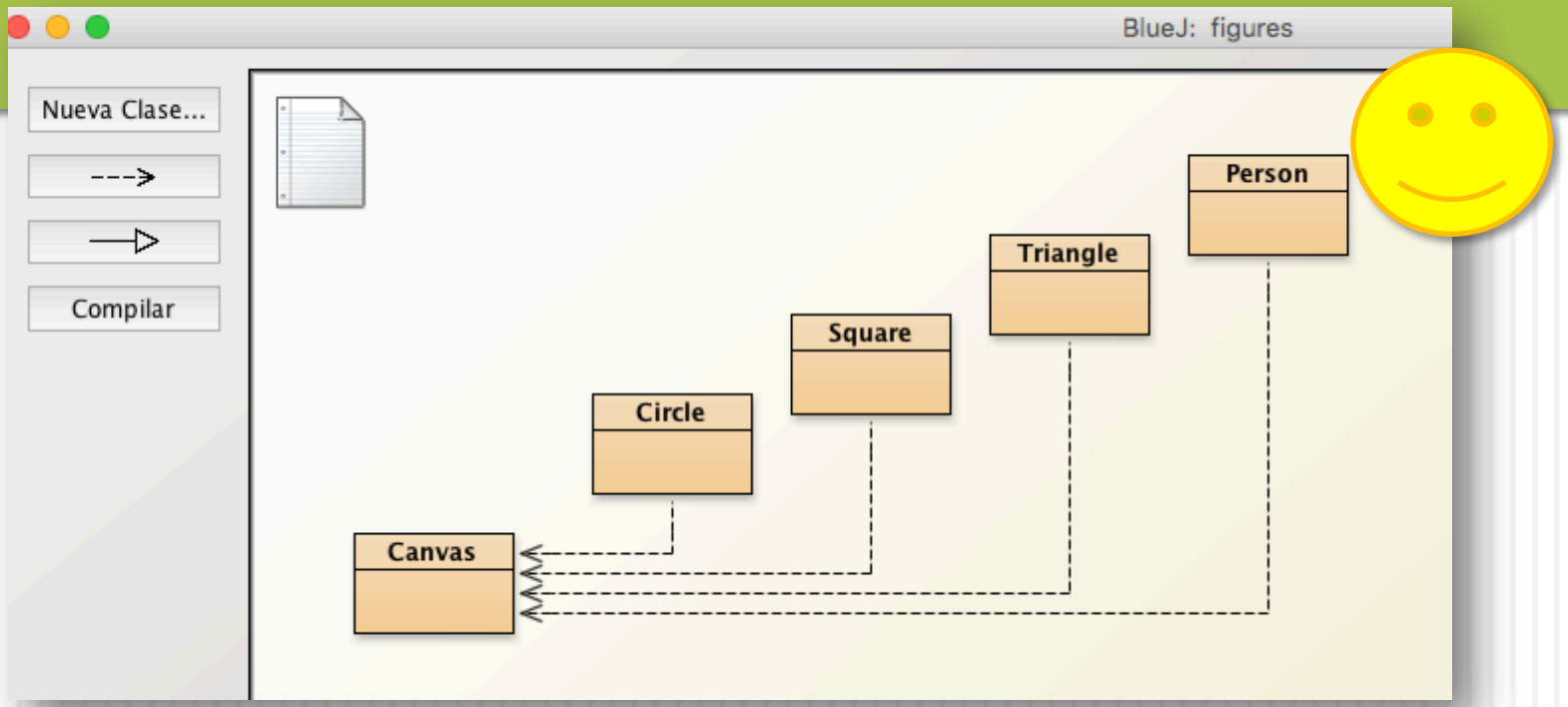
Advantages of using Objects

- *Reusability*: When you create classes, with its functionality defined "inside", that object could be used for another program.
- *Easier to debug*: OO applications are devided into pieces , so if you change one class, you know it´s not dependent on anything else.
- *Modularity*: The code of an object can be rewritten without the need of touching other code from other objects of your application.

Practice

A2.1: Follow these steps:

1) Install Bluej in your computers.
2) Download figures.zip from Moodle. Unzip it.
3) Start BlueJ and open “figures”.

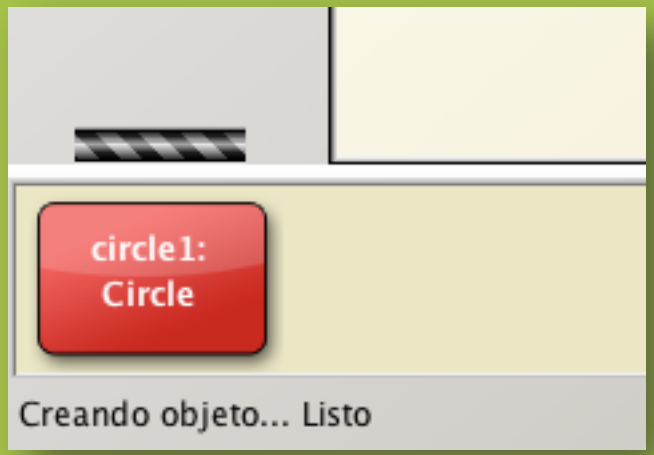


Practice

A2.1 (cont.):

4) Right-click on the Circle class and choose "new Circle()".

5) The system asks you for a "name of the instance"; click OK—the default name supplied is good enough for now. You will see a red rectangle toward the bottom of the screen labeled "circle1".

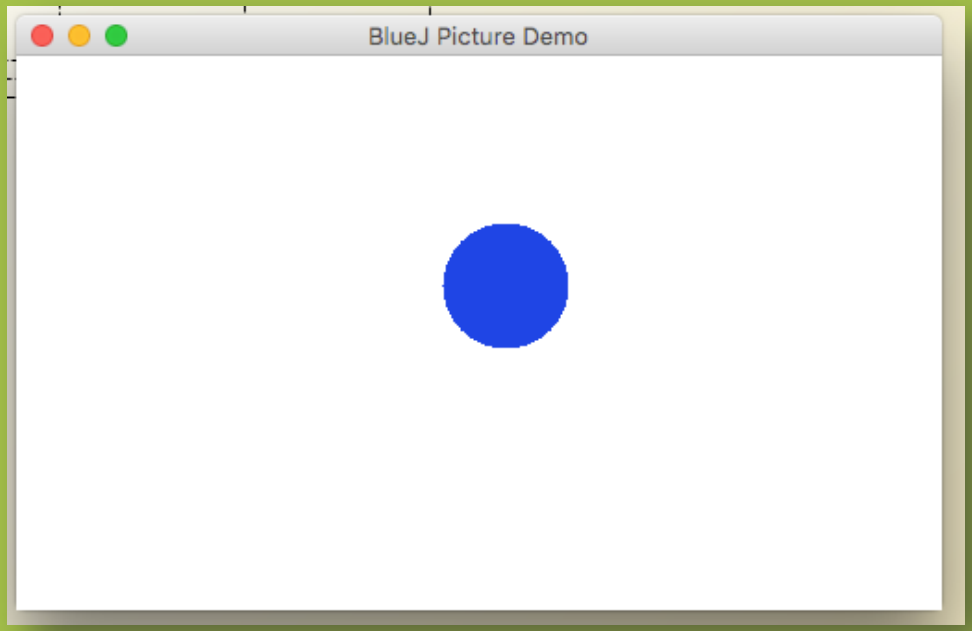


6) Create another circle. Then create a square.

Practice

A2.1 (cont.):

6) Right-click on one of the circle objects (not the class!), and you will see a pop-up menu with several operations. Choose "makeVisible" from the menu.



7) What happens if you call moveDown twice? Or three times? What happens if you call makeInvisible twice?

Practice

A2.1 (cont.):
8) Now invoke the *moveHorizontal* method. You will see a dialog appears that prompts you for some input. Type in 50 and click OK. You will see the circle move 50 pixels to the right.
9) After that, Try invoking the *moveVertical*, *slowMoveVertical*, and *changeSize*.

- The additional values that some methods require are called **parameters.**
- A method indicates what kinds of parameters it requires in its deffinition:

```
public void moveHorizontal(int distance)
```

- That is called the **signature** of a method.

Practice

A2.1 (cont.):
10) Invoke the *changeColor* method on one of your circle objects and enter the string "red". This should change the color of the circle. Try other colors.
11) Invoke the *changeColor* method, and write the color into the parameter field without the quotes. What happens?
12) Create two circle objects. Make them visible, then move them around on the screen using the "move" methods. Make one big and yellow; make another one small and green.

- You have been able to create **multiple instances** from the same class.

Practice

A2.1 (cont.):

13) Using Bluej you are able to *inspect* each of the objects you create. Therefore, make a change in the **state** of an object (for example, by calling the moveLeft method) and check the new object state in the inspector.

- The set of values of all attributes defining an object is also referred to as the object's **state**

Practice

A2.1 (cont.):

14) Would you be able to reproduce these images, using a set of objects.

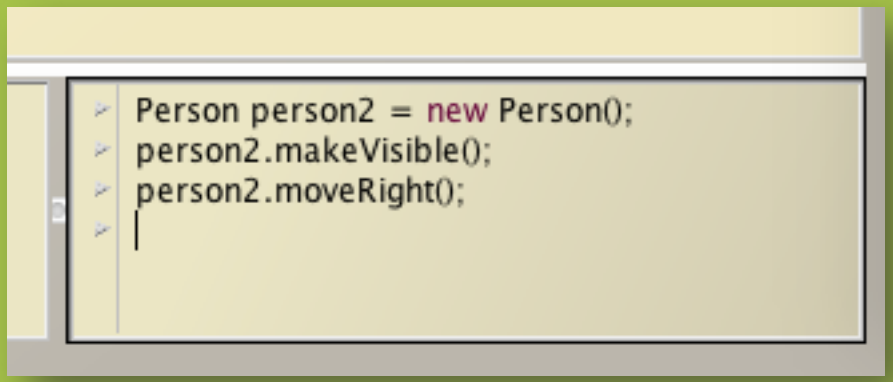
- In this part you have been able to create different kind of **objects/instances** in order to make your drawing.

Practice

A2.1 (cont.):

15) The object creation can also be using the command line that comes in Bluej. In order to do it, select *Show Code Pad* from the *Ver* menu. This should display a new pane next to the object bench in your main BlueJ window. This pane is the Code Pad. You can type Java code here.

In the Code Pad, type the code shown below to create a person object and call its *makeVisible* and *moveRight* methods.



Class definition

- Structure of a class definition:

*[xxx] class **N**ameOfTheClass [xxx]* {

[Atributtes (0…N)]

[Methods (0…N)]

}

Class definition

- Example:

The public statement means that the class is available for use by "the public"—in other words, by any program that wants to use Modem objects.

attributes

methods

```
import java.awt.*;
import java.awt.geom.*;

/**
 * A circle that can be manipulated and that draws itself on a canvas.
 *
 * @author
 * @version
 */

public class Circle
{
    private int diameter;
    private int xPosition;
    private int yPosition;
    private String color;
    private boolean isVisible;

    /**
     * Create a new circle at default position with default color.
     */
    public Circle()
    {
        diameter = 68;
        xPosition = 230;
        yPosition = 90;
        color = "blue";
    }

    /**
     * Make this circle visible. If it was already visible, do nothing.
     */
    public void makeVisible()
    {
        isVisible = true;
        draw();
    }

    /**
     * Make this circle invisible. If it was already invisible, do nothing.
     */
```

Characteristics of OOP

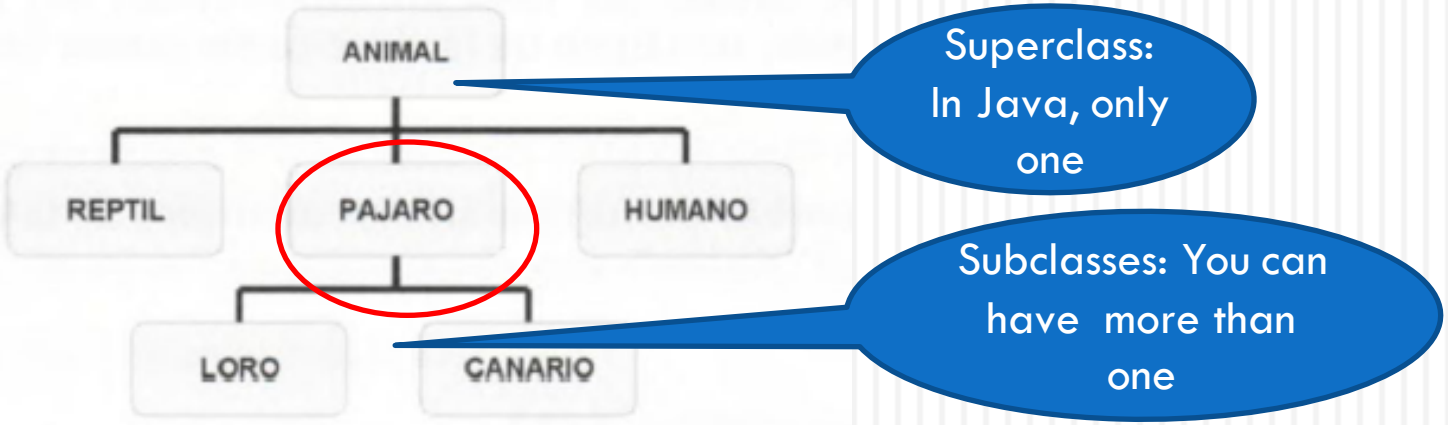
A. Abstraction:
   - RAE definition: "*separar por medio de una operación intelectual las cualidades de un objeto para considerarlas aisladamente o para considerar el mismo objeto en su pura esencia o noción*".
   - If we program OOP, we visualize abstractly the features of objects that are going to be part of our program, with their attributes and methods.

B. Encapsulation: When you program OO, you see objects for the behaviour you want them externally to have.
   - Example: In the bird class, you want it to sing, so you will call the method sing for the object "Dorian", and it will sing (execute the code in order to sing).
   - A lot of times, you will be using methods from objects that you do not know inside how they do it, but you know the result expected. This is called the *interface*.

Characteristics of POO

C. Inheritance:

- It is the way one object can inherit behaviour and attributes from other objects that are similar to it.
- When you start creating objects for use in other programs, you will find that some new objects you want are a lot like other objects that have already been developed.
- This system of classes is called a *class hierarchy*. Example:



- When one class inherits from a superclass, it obtains all its attributes and methods. Moreover, the class can extend its attributes and functionality.

Characteristics of POO

D. Polymorphism:

- It is the ability of an object to take many forms: the way you can create different ways of the same method, so this method will offer different behaviours.
- Example: método *setSueldo()* → *overloading methods*

Properties and methods

- All the properties (attributes) created must belong to a class.
- The attributes can be Java primitive types (char, int,…) or objects of another class.
  - Example: A “car” class can have an object of type “engine” as an attribute.

Properties and methods

- Example:

Pajaro.java

```
public class Pajaro {
    // Attributes or properties
    private char color;
    private int edad;
    // Methods
    public void setEdad (int e){edad =e;}
    public void printEdad () {System.out.println(edad);}
    public void setColor (char c) {color=c;}
}
```

Test.java

```
class Test {
    public static void main (String[] args){
        Pajaro p;
        p= new Pajaro();
        p.setEdad(5);
        p.printEdad();
    }
}
```

Practice

A2.2: Follow these steps:

1) Open the lab-classes

2) Create an object of class Student.

3) Create some student objects. Call the *getName* method on each object. Explain what is happening.

4) Create an object of class *LabClass*. As the signature indicates, you need to specify the maximum number of students in that class (an integer). Call the numberOfStudents method of that class. What does it do?

Practice

A2.2: (Cont.)

5) Look at the signature of the *enrollStudent* method. You will notice that the type of the expected parameter is *Student*. Make sure you have two or three students and a *LabClass* object on the object bench, then call the *enrollStudent* method of the LabClass object. Add one or more other students.

6) Call the printList method of the LabClass object.

```
Class list:
Ines, student ID: 1, credits: 0
Maria, student ID: 2, credits: 0
Number of students: 2
```

Practice

A2.2: (Cont.)

7) Now click on *Herramientas/ Reset Java Virtual Machine*.

8) After that, create three students with the following details:

Snow White, student ID: A00234, credits: 24

Lisa Simpson, student ID: C22044, credits: 56

Charlie Brown, student ID: A12003, credits: 6

Then enter all three into a lab and print a list to the screen.

8) Use the inspector on a *LabClass* object to discover what fields it has.

9) Set the *instructor*, *room*, and *time for a lab*, and print the list to the terminal window to check that these new details appear. Check also the inspector again.