Instituto Superior de Engenharia do Porto
(ISEP)

Licenciatura em Engenharia de Telecomunicações e Informática (LETI)

Desenvolvimento de Software e Sistemas Móveis (DSSMV)


**DSSMV: PL: Week 3**

# Classes

Paulo Baltarejo Sousa and Carlos Filipe Freitas
{pbs,caf}@isep.ipp.pt
2025/26

# 1 Classes

## 1.1 Course

Create a Project and add a class called `Course` like this:

```
public class Course {
  public static String initials;
  public static int semester;
  public static String instructor;
  public static String evaluation;

  public void print(){
    System.out.println("Initials:" + initials);
    System.out.println("Semester:" + semester);
    System.out.println("Instructor:" + instructor);
    System.out.println("Evaluation:" + evaluation);
  }
}
```

Use `Main` to test `Course` class with the following code:

```
public class Main {

  public static void main(String[] args) {
    // write your code here
    Course dssmv = new Course();
    Course esoft = new Course();
    dssmv.initials ="DSSMV";
    dssmv.semester = 2;
    dssmv.instructor = "PBS";
    dssmv.evaluation = "Exam";

    esoft.initials = "ESOFT";
    esoft.semester = 2;
    esoft.instructor = "PAM";
    esoft.evaluation = "Frequency";

    dssmv.print();
    esoft.print();
  }
}
```

Run and comment the output.

## 1.2 AddressBookApplication

Create a Java command line project called `AddressBookApplication`. Use the MVC acrchitecture pattern. All `Model` class methods must be tested using JUnit Tests framework.

### 1.2.1 Address

Create a class called `Address` with folowing:

- Atributes:

  - `private int number`
  - `private String streetName`
  - `private String zipCode`
  - `private String city`
  - `private String country`
- Constructors (use `Alt+Insert` keys)
  - `public Address()`
  - `public Address(int number, String zipCode, String streetName,String city, String country)`
- Methods
  - Getters and setters (use `Alt+Insert` keys)
    - `public int getNumber()`
    - `public void setNumber(int number)`
    - `public String getStreetName()`
    - ...
  - `public void print()`: prints the `this` data.

### 1.2.2 Date

Create a class called `Date` with folowing:

- Atributes:
  - `private int day`
  - `private int month`
  - `private int year`
- Constructors
  - `public Date()`
  - `public Date(int day, int month, int year)`
- Methods
  - Getters and setters (use `Alt+Insert` keys)
    - `public int getDay()`

        * `public void setDay(int day)`
        * `public int getMonth()`
        * ...
    - `public void print()`: prints the `this` data.

### 1.2.3 Person

Create a class called `Person` with folowing:

- Atributes:
    - `private int id`
    - `private String name`
    - `private Address address`
    - `private Date birthDay`
- Constructors
    - `public Person()`
    - `public Person(int id, String name, Address address, Date birthDay)`
- Methods
    - Getters and setters (use `Alt+Insert` keys)
        * `public int getId()`
        * `public void setId(int id)`
        * `public String getName()`
        * ...
    - `public void print()`: prints the `this` data.

### 1.2.4 AddressBook

Create a class called `AddressBook` with folowing:

- Atributes:
    - `private String title`
    - `private ArrayList<Person> people`
- Constructors

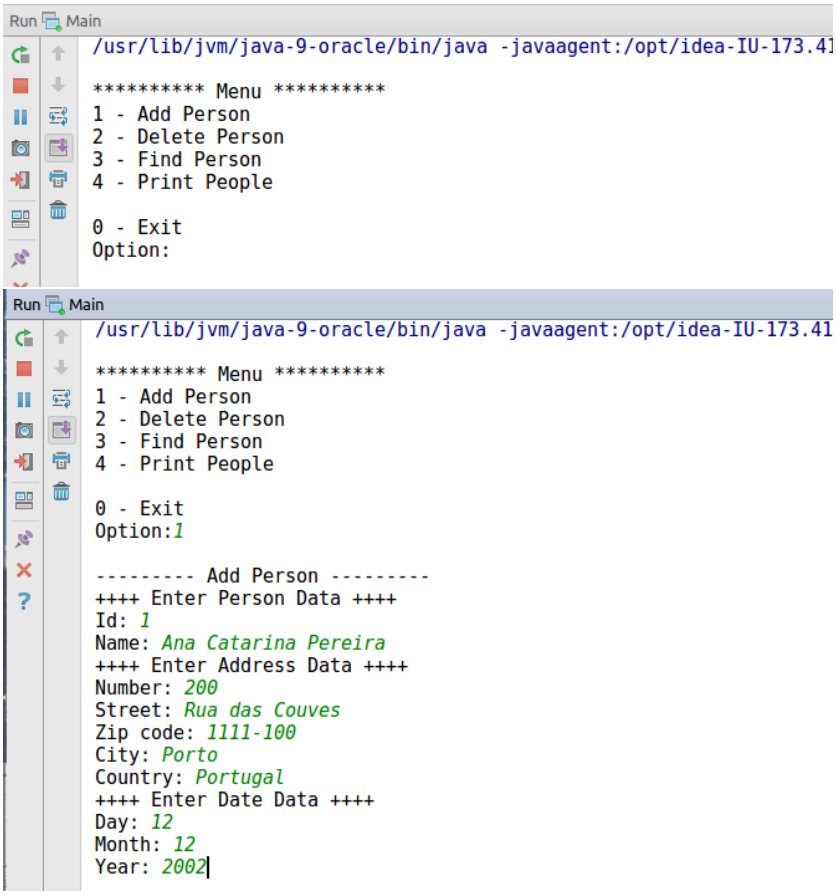
  - `public AddressBook()`
  - `public AddressBook(String title)`

- Methods
  - Getters and setters (use `Alt+Insert` keys)
    * `public String getTitle()`
    * `public void setTitle(String title)`
    * `public ArrayList<Person> getPeople()`
    * ...
  - `public void addPerson(Person p)`: adds an object `Person` to the `ArrayList<Person> people`.
  - `public Person getPerson(int id)`: given an `id` it returns an object `Person` from the `ArrayList<Person> people`.
  - `public Person deletePerson(int id)`: given an `id` it deletes an object `Person` from the `ArrayList<Person> people`.
  - `public void print()`: prints the `title` as well as all objects in the `ArrayList<Person> people`.

### 1.2.5 User Interface

Create a user interface for the `AddressBookApplication` with the following funcionalities:

- Add a person to the address book.
- Delete a person from the address book.
- Find a person in the address book.
- List all people in the address book.




### 1.2.6 Constraints

Change the `AddressBookApplication` implementation in order to fullfil the following constraints:

- Any date must be valid.
  - A year is a leap year if it is divisible by 4 but not by 100, or it is divisible by 400.
- The Zip Code must be in format `xxxx-xxx`, where each `x` is a number in the range $[0-9]$.
- The person identifier (`id`) must unique, that is, repeated person identifiers are not allowed.
  - Preferably, the person identifier (`id`) must be assigned automacally by the application.

# 2 Inheritance & Polymorphism

## 2.1 Shape, Circle and Rectangle

Consider the following classes, `Shape`, `Circle`, and `Rectangle`.

```
public class Shape {
  private String name;
  private String color;
  private boolean filled;
  private int x;
  private int y;

  public Shape() {

  }
  public Shape(String name, String color, boolean filled, int x, int y) {
    this.name = name;
    this.color = color;
    this.filled = filled;
    this.x = x;
    this.y = y;
  }

  public String getName() {
    return name;
  }

  public void setName(String name) {
    this.name = name;
  }

  public String getColor() {
    return color;
  }

  public void setColor(String color) {
    this.color = color;
  }

  public boolean isFilled() {
    return filled;
  }

  public void setFilled(boolean filled) {
    this.filled = filled;
  }

  public int getX() {
    return x;
  }

  public void setX(int x) {
    this.x = x;
  }

  public int getY() {
    return y;
  }

  public void setY(int y) {
    this.y = y;
  }

  public void draw()
  {
    System.out.println("Shape: [name:" + name +", color:"+color+", filled:"+filled+",origin:"+x+","+y+"]");
  }
}
```

```
public class Circle extends Shape{
  int radius;

  public Circle() {
  }
  public Circle(String name, String color, boolean filled, int x, int y, int radius) {
    super(name, color, filled, x, y);
    this.radius = radius;
  }

  public int getRadius() {
    return radius;
  }

  public void setRadius(int radius) {
    this.radius = radius;
  }

  @Override
  public void draw() {
    //super.draw();
    System.out.println("Shape: [name:" + super.getName() +", color:"+super.getColor()+", filled:"+super.
          isFilled()+",origin:"+super.getX()+","+super.getY()+",radius:"+radius+"]");

  }
}
```

```
public class Rectangle extends Shape{
  private int width;
  private int height;

  public Rectangle() {
  }

  public Rectangle(String name, String color, boolean filled, int x, int y, int width, int height) {
    super(name, color, filled, x, y);
    this.width = width;
    this.height = height;
  }

  public int getWidth() {
    return width;
  }

  public void setWidth(int width) {
    this.width = width;
  }

  public int getHeight() {
    return height;
  }

  public void setHeight(int height) {
    this.height = height;
  }

  @Override
  public void draw() {
    //super.draw();
    System.out.println("Shape: [name:" + super.getName() +", color:"+super.getColor()+", filled:"+super.
          isFilled()+",origin:"+super.getX()+","+super.getY()+",width:"+width+", height:"+height+"]");
  }
}
```

1. Which is the output (print on the screen) generated by the code of line 13?

```
1  ArrayList<Shape> shapes = new ArrayList<Shape>();
```

```
2
3   Shape shape = new Shape("Shape 1","Red", true,0,0);
4   shapes.add(shape);
5
6   Circle circle = new Circle("Circle 1","Green", true,0,0,5);
7   shapes.add(circle);
8
9   Rectangle rectangle = new Rectangle("Rectangle 1","Blue", true,0,0,3,6);
10  shapes.add(rectangle);
11
12  for (int i = 0; i< shapes.size();i++){
13    shapes.get(i).draw();
14  }
```

(a) `Shape: [name:Shape 1, color:Red, filled:true,origin:0,0]`
`Shape: [name:Circle 1, color:Green, filled:true,origin:0,0,radius:5]`
`Shape: [name:Rectangle 1, color:Blue, filled:true,origin:0,0,width:3, height:6]`

(b) `Shape: [name:Shape 1, color:Red, filled:true,origin:0,0]`
`Shape: [name:Circle 1, color:Green, filled:true,origin:0,0]`
`Shape: [name:Rectangle 1, color:Blue, filled:true,origin:0,0]`

(c) None, because the program has an error. The array list `shape` accepts only `Shape` class objects.

2. Which are the output(print on the screen) generated by the code of line 7?

```
1   public class Main {
2
3   public static void main(String[] args) {
4   // write your code here
5   Rectangle rectangle = new Rectangle("Rectangle 1","Blue", true,0,0,3,6);
6   Shape shape = rectangle;
7   shape.draw();
8   }
9   }
```

(a) `Shape: [name:Rectangle 1, color:Blue, filled:true,origin:0,0]`

(b) `Shape: [name:Rectangle 1, color:Blue, filled:true,origin:0,0,width:3, height:6]`

(c) None, because the program has an error. A `Rectangle` class object reference cannot be assigned to a `Shape` class object reference

3. Which are the output(print on the screen) generated by the code of line 7?

```
1   public class Main {
2
3   public static void main(String[] args) {
4   // write your code here
5   Shape shape = new Shape("Shape 1","Red", true,0,0);
```

```
6   Rectangle rectangle = (Rectangle) shape;
7   rectangle.draw();
8   }
9   }
```

(a) `Shape: [name:Shape 1, color:Red, filled:true,origin:0,0]`

(b) `Shape: [name:Shape 1, color:Red, filled:true,origin:0,0,width:0, height:0]`

(c) None, because the program has an error. A `Shape` class object reference cannot be assigned to a `Rectangle` class object reference