Exercises - Unit 4
Datatype classes and methods

Group I1E

Year 2017/2018

1. Given the following class `IntPoint`:

```
/**
 * Class IntPoint: defines points in a bi-dimensional integer space
 * with the following functionalities. <br>
 * @author IIP
 * @version 2017
 */
public class IntPoint {
   private int x; // abscissa component
   private int y; // ordinate component
   /** create a point (0,0). */
   public IntPoint() { x = 0; y = 0; }
   /** create a point (abs,ord). */
   public IntPoint(int abs, int ord) { x = abs; y = ord; }
   /** create a point (coord,coord). */
   public IntPoint(int coord) { x = coord; y = coord; }
   /** gets the point abcissa */
   public int abscissa() { return x; }
   /** gets the point ordenate */
   public int ordenate() { return y; }
   /** gets the distance from the point to the origin */
   public double distOrigin() { return Math.sqrt(x*x + y*y); }
   /** update components of the point to (abs, ord). */
   public void assign(int abs, int ord) { x = abs; y = ord; }
}
```

   (a) Write the **name** of its attributes, their datatypes and accesibility

   (b) Write the **header** of the constructor methods

   (c) Write the **name** of the rest of methods

   (d) Write a program class `IntPointTest` in whose `main` a `IntPoint` object `p1` is declared and created with coordinates (3,-2), and its distance to origin is shown on the screen.

2. Given the following `Circle` class:

```
/** Class Circle: defines a circle with a given radius, color, ... */
public class Circle {
     private double radius; private String color; private int centerX, centerY;
     /** create a Circle with radius 50, black, center in (100,100) */
     public Circle() {
       radius = 50; color = "black"; centerX = 100; centerY = 100;
     }
     /** create a Circle with the given radius, color, and center */
     public Circle(double r, String c, int cx, int cy) {
       radius = r; color = c; centerX = cx; centerY = cy;
     }
```

```
    /** consults the radius of the Circle. */
    public double getRadius() { return radius; }
    /** updates the radius of the Circle to newRadius. */
    public void setRadius(double newRadius) { radius = newRadius; }
    /** calculates the area of the Circle. */
    public double area() { return 3.14 * radius * radius; }
    /** calculate Circle perimeter */
    public double perimeter() { return 2 * 3.14 * radius; }
    // And more methods...
}
```

tell what is the error in the following program:

```
public class CircleTest {
   public static void main (String[] args) {
      Circle c = new Circle(2.5, "red", 1, 1);
      System.out.println("The radius of the circle is:" + c.radius);
   }
}
```

Describe how the program must be modified to obtain a correct result.

3. Fill in the code of the following `Square` class to obtain a functionality similar to the previously shown `Circle` class:

```
public class Square {
  private ...    // Attribute for side length
  private ...    // Attribute for color
  private ...    // Attribute for x coordinate of the center
  private ...    // Attribute for y coordinate of the center
  public Square(double side, String color, int cx, int cy) {
    ...
  }
  /** consults the side length of the Square */
  public double getSide() { ... }
  /** consults the color of the Square */
  public String getColor() { ... }
  /** consults the X center coordinate of the Square */
  public int getCenterX() { ... }
  /** consults the Y center coordinate of the Square */
  public int getCenterY() { ... }
  /** updates the side length of the Square to newSide */
  public void setSide(double newSide) { ... }
  /** updates the color of the Square to newColor */
  public void setColor(String newColor) { ... }
  /** updates the center of the Square */
  public void setCenter(int cx, int cy) { ... }
}
```

4. Given the class `Circle` previously described, rewrite the methods `area` and `perimeter` in order they use the proper constant of the `Math` class (in the package `java.lang`) for the calculations.

5. Given the class `Circle` previously described, modify it in order to:

   - Keep the count of how many `Circle` objects have been created by using the constructor without arguments
   - Keep the count of how many `Circle` objects have been created by using the constructor with arguments
   - Know if any `Circle` object created was of color “red”
   - Know the total area that is covered by all the `Circle` objects created

   You will need to incorporate class (`static`) attributes and manage them properly.

6. Write a datatype class `RealPoint3D` that represents a three-dimensional point in the space. Include:
   - The proper number of attributes
   - A constructor without parameters (that initialises the point to the origin)
   - Another constructor with parameters that represent each of the coordinates of the point
   - The corresponding consultors and modifiers
   - A method to transform a point to its symmetric with respect to the origin (0,0,0)

   Write a program class that creates a `RealPoint3D` object with coordinates (3.2, -3.5, 0.5), prints on the screen the coordinates of the `RealPoint3D` object, transforms it to its symmetric, and prints the new coordinates.

7. Write a datatype class `UsedCar` that represents a car for a used car dealer. The class must include:
   - Attributes for plate number (string), number of km (real number) and construction year (integer number)
   - A default constructor (without parameters) that gives to the plate the value "XXXX-XXX", to number of kilometers 100,000 and to year 2000
   - A constructor that receives all data (you can suppose all given data is valid)
   - The corresponding consultors and modifiers

8. Write a datatype class `Student` that represents a student. The class must include:
   - Attributes for the name (string), ID (string), and the grades for five subjects (real numbers)
   - A default constructor (without parameters) that assigns to name and ID the empty string (`""`) and to the grades the value -1
   - A constructor that receives name and ID and inits the grades to -1
   - The corresponding consultor and modifier methods; modifier methods on grades must check that the given grade is correct ($0 \leq \text{grade} \leq 10$)

9. Write a datatype class `Screen` that represents the features of a computer screen. The class must include
   - Attributes for the retailer (string), size (positive real number, in inches), horizontal and vertical refresh frequencies (positive integer, in Hz), and current resolution (horizontal and vertical, two positive integers)
   - A constructor that only receives the size and initialises the rest of attributes as `"???"` for retailer, 70×50 Hz refresh frequencies, and 1080×720 for resolution
   - A constructor that receives as many parameters as necessary to initialise all the attributes; in case a parameter has an incorrect value, the corresponding attribute would be initialised as in the other constructor
   - The corresponding consultor and modifier methods; modifiers must check data restrictions when necessary

10. Write a datatype class `Event` that represents an event in a calendar application. The class must implement:
    - Attributes to represent date (three integers), hour (two integers), place (string), and description (string)
    - A constructor that only receives date and hour, making place and description to be empty strings (`""`)
    - Another constructor that receives as many parameters as needed for the event; you can suppose that parameters are always valid
    - The corresponding consultor and modifier methods; you can suppose that parameters are always valid

11. Write a datatype class `TVChannel` that represents a TV channel in a TDT decoder. The class must include:
    - Attributes for the number of the channel (natural number), the frequency (positive real number), and the identifier (string), and a class (`static`) attribute that stores the maximum number of channel used until this moment
    - A default constructor (without parameters) that inits frequency to 0.0, identifier as "???" and number as maximum number used plus 1
    - A constructor that receives number, frequency and identifier; in case number or frequency are not valid, the values to be assigned to the attributes must be the same than in the other constructor
    - The corresponding consultors and modifiers for the object attributes, and the consultor for the class attribute

    Be careful with the management of the `static` attribute in all the methods that give value to the number of channel.

12. Write a datatype class `DigitalCamera` that represents a digital camera. The class must include:

    - Attributes that represent the retailer (string), model (string), resolution (a positive real, in Mpixels), and free storage (non-negative integer, in bytes)
    - A constructor that receives only retailer and model, making resolution equal to 5 Mpixels and free storage equal to 1Gb
    - Another constructor that receives all the parameters needed to initialise all the attributes (if the numeric parameters are not valid, attributes must be initialised to the same values that in the other constructor)
    - The corresponding consultors and modifiers, validating the new values in the modifiers
    - A method `public boolean newPicture(int size)` that receives the size (in bytes) of a newly taken picture and checks if it could be stored; if so, it updates the free storage and returns `true`; otherwise, it returns `false`

13. Given the following class `Circle`:

```
public class Circle {
    private double radius; private String color;
    private int centerX, centerY;
    public Circle() { radius = 50; color = "black"; centerX = 100; centerY = 100; }
    public Circle(double r, String c, int px, int py) {
      radius = r; color = c; centerX = px; centerY = py; }
    public double getRadius() { return radius; }
    public String getColor() { return color; }
    public int getCenterX() { return centerX; }
    public int getCenterY() { return centerY; }
    public void setRadius(double newRadius) { radius = newRadius; }
    public void setColor(String newColor) { color = newColor; }
    public void setCenter(int px, int py) { centerX=px; centerY=py; }
    public void toRight() { centerX += 10; }
    public void grow() { radius = radius * 1.3; }
    public void decrease() { radius = radius / 1.3; }
    public double area() { return 3.14 * radius * radius; }
    public double perimeter() { return 2 * 3.14 * radius; }
    public String toString() { String res = "Circle with radius "+ radius;
        res += ", color "+color+" and center ("+centerX+","+centerY+")";
        return res; }
}
```

    And the class `RealPoint2D`:

```
public class RealPoint2D {
  private double x, y;
  public RealPoint2D() { x=0.0; y=0.0; }
  public RealPoint2D(double x, double y) { this.x = x; this.y = y; }
  public double getX() { return x; }
  public double getY() { return y; }
  public double setX(double x) { this.x = x; }
  public double setY(double y) { this.y = y; }
  public String toString() { return "("+x+","+y+")"; }
}
```

    Redefine `Circle` to substitute the attributes `centerX` and `centerY` by an only attribute of this class `RealPoint2D`; this includes all the affected attributes and methods; take into account that `RealPoint2D` attributes are `double`, not `int`

14. Repeat the previous exercise for the following class `Square`:

```
public class Square {
  private double side;
  private String color;
  private int centerX;
  private int centerY;
  public Square(double s, String c, int cx, int cy)
    side = s; color = c; centerX = cx; centerY = cy;
  }
  public double getSide() { return side; }
  public String getColor() { return color; }
  public int getCenterX() { return centerX; }
  public int getCenterY() { return centerY; }
  public void setSide(double newSide) { side = newSide; }
  public void setColor(String newColor) { color = newColor; }
  public void setCenter(int cx, int cy) { centerX = cx; centerY = cy; }
  public void toRight() { cx += 10; }
  public void increase() { side = side * 2; }
  public void decrease() { side = side / 2; }
  public double area() { return side*side; }
  public double perimeter() { return 4*side; }
  public String toString() { String res = "Square with side " + side;
    res += " with color " + color + " and center ("+centerX+","+centerY")";
    return res;
  }
}
```

15. Add to the class `Square` of the previous exercise (the modified class) a constructor method `Square(RealPoint2D bottomLeft, RealPoint2D topRight)` that defines the square by the position of two of its corners

16. Write a `Rectangle` class based on the `Square` class defined in the solution of the previous exercise; must have `width` and `height` instead of `side`; include in that new class a constructor whose parameters are two `RealPoint2D` objects that define the bottom-left and top-right corners

17. Add an `equals` method to the classes `Circle`, `Square`, and `Rectangle` defined in the previous exercises

18. Write a `Triangle` class which represents a triangle; it must include:
    - Attributes that represent the vertexes (must be three `RealPoint2D` objects)
    - A constructor that receives the three vertexes
    - Consultors and modifiers for each attribute
    - A method to calculate the perimeter of the triangle
    - A `toString` method to write the data of the triangle (the vertexes)
    - An `equals` method that compares two triangles

19. Write a program class that allows to test the functionality of the `Triangle` class of the previous exercise; the program must create two triangles, calculate their perimeters, print their data, and compare them

20. Write a `DayTime` class that represents the hour and minutes in a day in 24 hour format; supose that attributes always get correct values. Write constructors, consultors and modifiers for each attribute.

21. The method `static long currentTimeMillis()`, from the predefined class `System`, returns the number of milliseconds from 00:00 hours of Jan, 1st, 1970, to now. Add a constructor to the class `DayTime` that uses this method to init the object to the current time.

22. Write a Java method in the `DayTime` class that returns the number of minutes that passed from midnight until that time (e.g., for 07:35 it will return 455, for 17:18 it will return 1038).

23. Write a Java program class that creates two `DayTime` objects and uses the method defined in the previous exercise to calculate the difference in minutes between the two `DayTime` objects. You can supose that the second time is greater that the first one.

24. Write a Java program class with a static method to show on the screen a given `DayTime` object. The `main` method must create the `DayTime` object and call the static method to write the `DayTime`.

25. Write a Java datatype class `YTVideo` to store data on YouTube videos. It must define:
    - Attributes for the URL (string), the uploader user (string), duration in seconds (integer number) and number of visits (integer number)
    - An only constructor that receives URL, uploader, and duration (you can suppose that duration is always valid); initial number of visits will be 0
    - The corresponding `get` and `set` methods
    - A method `public void visit()` that increments in one the number of visits

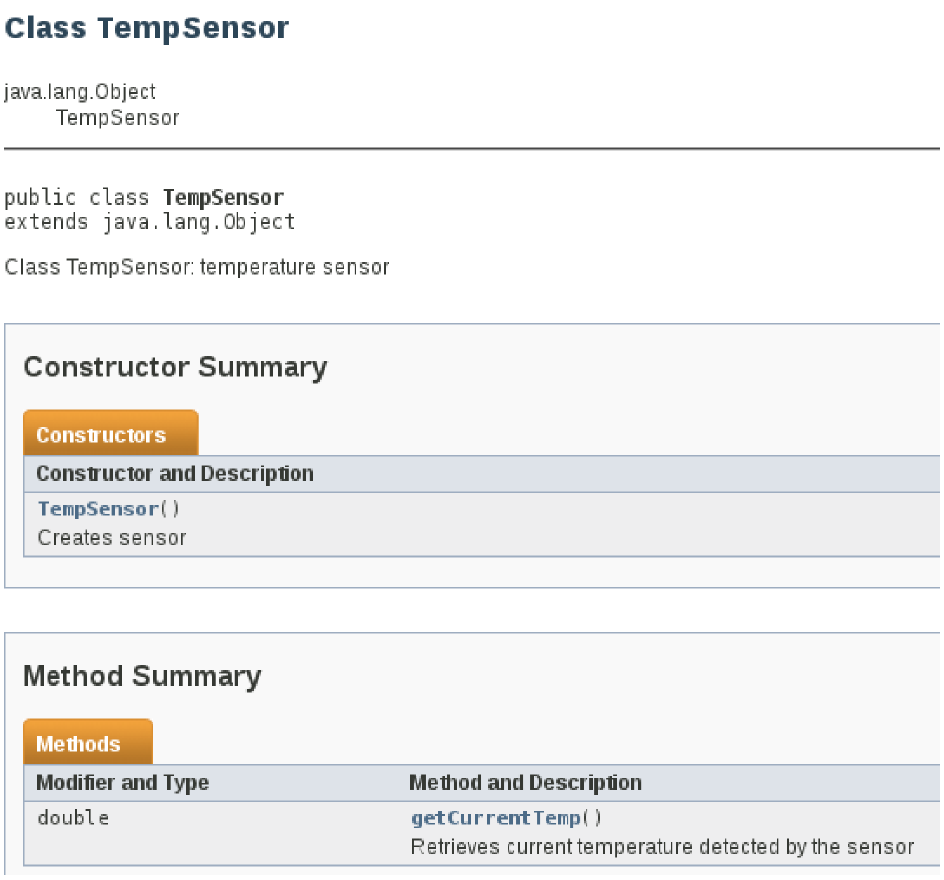
26. Write a Java datatype class `USBPen` to store data on USB memory sticks. It must include:
    - Attributes for manufacturer (String), encryption (boolean), total capacity (in Mb, positive integer number), and free capacity (in Mb, positive integer number)
    - A constructor that receives the manufacturer and the total capacity (supposed to be positive), and creates a USB memory (without encryption and without files) of that capacity
    - The corresponding consultors and modifiers (you can suppose that parameters are always correct)
    - A method `public void activateEncryption()`, that makes the encryption activated
    - A method `public boolean insertFile(int size)`, that inserts a file of `size` Mb (you can suppose that `size` is always positive) in the USB memory; it must return `true` if the insertion was possible and `false` otherwise; the free memory must be conveniently updated
    - `public boolean equals(Object o)`, that overrides the default method to say if two `USBPen` objects represent objects with the same values
    - `public String toString()`, that returns a `String` with format `‘‘Pen of MANUFACTURER un/encrypted with FREE Mb free of TOTAL Mb’’`

27. Write a Java datatype class `Car` that simulates the speed behaviour of a car. It must include:
    - Attributes for current speed, maximum speed, current gear, current RPM, and maximum RPM; all of them are real numbers greater than or equal to 0, except current gear that is an integer value between 0 and 5; speed is supposed to be in Km/h; current speed and RPM must not be higher than maximum speed and RPM
    - A default constructor (without parameters) that gives to maximum speed 180 and to maximum RPM 6500 (rest of attributes would be 0)
    - Another constructor that receives the values for maximum speed and RPM; in case the parameters are not valid values, the attributes must be initialised to the same values than in the other constructor
    - Consultor and modifier methods; modifiers must check that all attributes keep a correct value (think of all the possibilities!)

28. Complete the `Car` class by writing the following methods:
    - `public void accelerate(double a)`: increments in `a` (supposed to be positive) the current speed, and in `a`$\times$70 the current RPM; in case maximum speed or RPM is exceeded, the quantity to increment must be bounded by these limits
    - `public void brake(double a)`: decrements in `a` (supposed to be positive) the current speed, and in `a`$\times$70 the current RPM; in case any of those values gets lower than 0, both them and the gear must be changed to 0
    - `public void highGear()`: if current gear is lower than 5, increases gear in 1 and decreases current RPM to a 70%
    - `public void lowGear()`: if current gear is higher than 0, decreases gear in 1 and increases current RPM in a 30%
    - `public double consumption()`: returns litres of fuel consumed by 100 km., calculated as current RPM divided by gear; when gear is 0, it must return 0
    - `public double timeToPlace(double km)`: returns how many hours are needed to cover the distance `km` at current speed

29. Write a Java program class whose `main` method creates two `Car` objects (using default constructor), asks the user for the speed and gear for each car (giving the values to the corresponding objects), and calls another class (`static`) method in the class that returns the car with less consumption. Then, there must be printed out on the screen how many hours will need that car to cover 100 km.

30. We have available a class `TempSensor` where we have the following except of is documentation:



Class TempSensor

java.lang.Object
    TempSensor

```
public class TempSensor
extends java.lang.Object
```

Class TempSensor: temperature sensor

Constructor Summary

Constructors

Constructor and Description
`TempSensor()` Creates sensor

Method Summary

Methods

Modifier and Type	Method and Description
`double`	`getCurrentTemp()` Retrieves current temperature detected by the sensor

Write a Java datatype class `AirConditioning` that defines the behaviour of an air conditioning engine. The air conditioning is defined with:

- Four constant class attributes of type `char` that define the different modes, `AUTO`, `COOL`, `HEAT`, and `FAN`, with values `'a'`, `'c'`, `'h'`, and `'f'`, respectively
- A sensor temperature object of class `TempSensor`
- A state (boolean, `true` when is on and `false` when it is off)
- A reference temperature (real number)
- A fan speed (integer, rpm)
- The maximum fan speed (integer, rpm)
- The current mode (`char`), with values `AUTO` for auto, `COOL` for cooling, `HEAT` for heating, and `FAN` for only fan
- The internal mode (`char`), with values `COOL` for cooling and `HEAT` for heating.

Write the class with the corresponding attributes. Write the following methods:

- A constructor that receives the maximum fan speed (supposed to be positive) and initialises the rest of attributes as follows: state off, reference temperature is current temperature, current fan speed as a quarter of the maximum, auto mode, cool internal mode
- The corresponding consultors and modifier methods for state, reference temperature, and fan speed. Modifiers must take into account valid values (fan speed between 0 and maximum)
- Override the `equals` method, that must take into account current temperature and the rest of attributes

31. Complete the `AirConditioning` class with the following methods:
    - `public void switchOn()`: switches on the air conditioning
    - `public void switchOff()`: switches off the air conditioning
    - `public void changeMode(char m)`: changes the current mode to `m` if `m` represents a valid mode; if the current mode becomes `AUTO`, the internal mode must be changed accoding to current temperature and reference temperature
    - `public void updateMode()`: in case it is in auto mode, updates the current value of the internal mode to the correct one
    - `public void nextFanLevel()`: duplicates fan speed, except when maximum speed is reached, in which case fan speed will become an eighth of the maximum
    - `public double energy()`: returns the consumed energy in kW/h, calculated as the absolute difference between the current and reference temperatures multiplied by the ratio between maximum fan speed and current fan speed
32. Write a Java datatype class `Furniture` that represents the features for a famous Swedish DIY store furniture pieces. The class must contain:
    - Attributes for the reference (a string with format `"NNN.MMM.OO"`, where `N`, `M`, and `O` are digits), a description (string), cost (positive real), current number of units available (non-negative integer), and situation (two positive integers, one for aisle and another for section)
    - The constructor that receives the reference and the description, and makes default units equal to 10, cost equal to 4.99, and situation equal to (0,0); if reference is not in the proper format (see below), make it equal to `"000.000.00"`
    - A constructor that receives all parameters needed to fill in all the attributes (if any value given is not correct, make reference equal to `"000.000.00"` and numeric values equal to 0)
    - The corresponding consultors and modifiers, checking the correct values in the modifiers
    - A private method to check that the reference is in the correct format (i.e., length 10, three digits, point, three digits, point, two digits)
    - An overriden `equals` method, that must take into account only the reference
    - An overriden `toString` method to make it return a description of the object similar to `"DESCRIPTION (COST): currently UNITS units available in aisle AISLE, section SECTION (ref: REFERENCE)"`
33. Complete the `Furniture` class with the following methods:
    - `public boolean sell()`: decrements in one the available number of units (only when there are available units); if there were available units, it must return `true`, otherwise it must return `false`.
    - `public void replace(int u)`: increments the available number of units in `u`; if `u` is negative, makes no changes.
    - `public void move(int a, int s)`: moves furniture to aisle `a` and section `s`; if any of the two parameters is not valid, it is not moved.
    - `public double stockValue()`: returns the value of the stock of the furniture (number of units times cost).
    - `public void sale(int p)`: decrements in `p` percent the value of the furniture; if `p` is invalid (negative or greater than 100), makes no changes.
34. Implement a datatype class `Subject` that implements a university degree subject. It must include:
    - Constant `char` attributes for defining semester, `FIRST`, `SECOND`, and `ANNUAL`, with values `'A'`, `'B'`, and `'T'` respectively
    - Attributes for code (positive integer), name (string), short name (string), credits (positive real), and semester (char, with values `FIRST`, `SECOND` and `ANNUAL`)
    - A constructor that receives code, name, and credits; default short name is the empty string (`""`); default semester is `ANNUAL`; it must check that code and credits are positive, and in case they are not, code must be 99999 and credits must be 100
    - A constructor that receives the data necessary for initing all the attributes; it must check the same than the other constructor and that semester has as value `FIRST`, `SECOND`, or `ANNUAL` (if not, employ the same default value than in the other constructor)

- The `get` and `set` methods for each attribute; `set` methods must check that code and credits are positive and that semester has as value `FIRST`, `SECOND`, or `ANNUAL`; in other case, they will not modify the corresponding attributes
- An overriden `equals` method, that must only check the value of the code of the subjects
- An overriden `toString` method to make it return the string in format: "Subject NAME of CREDITS credits in SEMESTER", where SEMESTER must be `"semester A"`, `"semester B"`, or `"all year"` according to the semester attribute
- A method that returns the minimal number of hours of study for the subject (25 hours each credit)
- A method that returns if the subject is of a semester (given as parameter), i.e., `boolean` method

35. Write a Java datatype class `Segment` that defines a geometrical segment in a drawing space. It must include:
    - Attributes for the extreme points (pairs of real numbers), width (in pixels) and color (string)
    - A default constructor (without parameters) that creates an empty segment (width 0) in (0,0) of white color
    - Another constructor that receives the extreme points, making width of 1 and white color
    - Another constructor that receives all the possible values; width must be validated (positive, in other case it would be 1)
    - The corresponding consultors and modifiers; modifier for width must validate the new value
    - An overriden `equals` method
    - An overriden `toString` method that must return in format `‘‘COLOR-WIDTH-(ORGX,ORGY)-(DSTX,DSTY)’’`
36. Write for the `Segment` class the following methods:
    - `public double length()`: returns the length of the segment, calculated as the Euclidean distance between the extreme points
    - `public double angle()`: returns the angle (in radians) that forms the segment with respect to the horizontal axis; **tip**: use trigonometrical methods of `Math`
    - `public void widen(int p)`: increases width of the segment in `p` (supposed to be positive)
    - `public void narrow(int p)`: decreases width of the segment in `p` (supposed to be positive); in case the result is 0 or negative width, the previous width must be kept
    - `public void symmetric(boolean p)`: when `p` is `true`, it changes the second point to make the symmetric image with respect to the first point; when `p` is `false`, changes the first point and makes symmetric with respect to the second
37. Write a Java program class in whose `main` method, the extreme points for two segments are asked to the user and the corresponding `Segment` objects are created. Then the program must write on the screen if the segments form an acute, right, or obtuse angle (from the first to the second).
38. Make a trace of the following program and write what is its output. Draw yourself the stack evolution to make it clearer.

```
public class Example {
  public static void exchange(int j, int k) {
    int aux = j;
    j = k;
    k = aux;
    System.out.println("Inside: " + j + " " + k);
  } // of exchange
  public static void inc(int j, int k) {
    j++;
    k++;
    System.out.println("Inside: " + j + " " + k);
  } // of inc
  public static void main(String[] args) {
    int j = 1335;
    int k = 3672;
    System.out.println("Before: " + j + " " + k);
    exchange(j,k);
    inc(k,j);
    System.out.println("After: " + j + " " + k);
  } // of main
} // of Example
```

39. Suposing the following definition of the class `RealPoint2D`:

```
public class RealPoint2D {
  private double x, y;
  public RealPoint2D() { x=0.0; y=0.0; }
  public RealPoint2D(double x, double y) { this.x = x; this.y = y; }
  public double getX() { return x; }
  public double getY() { return y; }
  public double setX(double x) { this.x = x; }
  public double setY(double y) { this.y = y; }
  public String toString() { return "("+x+","+y+")"; }
}
```

Make a trace of the following program and write what is its output. Draw yourself the stack evolution to make it clearer.

```
public class Example2 {
  public static void main(String[] args) {
    RealPoint2D p1 = new RealPoint2D(1.0,-1.0);
    RealPoint2D p2 = new RealPoint2D();
    System.out.print("Before calling ");
    System.out.println("the points are: " + p1 +" and "+ p2);
    exchangePoints(p1,p2);
    System.out.print("After calling ");
    System.out.println("the points are: " + p1 +" and  "+ p2);
  } // of main
  private static void exchangePoints(RealPoint2D a, RealPoint2D b) {
    RealPoint2D aux = a;
    a = b;
    b = aux;
  }
} // of Example2
```

40. Rewrite the code of `exchangePoints` in the code of the previous exercise to make an effective exchange of the values of the `RealPoint2D` objects that receives as parameters the method