Inheritance

In the following, we will create classes to manipulate geometry figures (for instance, we could use them to draw a user interface, though we will not do it today). First, we will create a class to model a point. To do so, we will simply represent a point using its coordinate.

We also want to create a class for a labelled point: it is a point, but it will have a label. To do so, we will use the concept of inheritance.

Finally, we will also use our classes for modelling simple geometric figures (rectangle, circle, ...). For example to model a circle, all we need is a point that will model the center of the circle and a **double** that will model the radius of the circle. To model a rectangle, we will consider only "straight" rectangles and we can identify them using the top left corner and bottom right corner (which are ... `Points`!). Again, we will use inheritance.

For each question, write the corresponding code and test it in the `main` method.

1. Implement a `Point` class. You will use a constructor with two arguments of type `double` that model the cartesian coordinate of a point.
2. Implement a subclass `LabelledPoint` that contains a label of type `String`.
3. Now that we have two classes, we use this opportunity to test the visibility of instance variables: **public**, **private** and **protected**. First check that when you declare the variables as **public**, you can access them. Then, check what happens when you update your code by declaring them as **private**. Finally, check again with **protected**. Think of a nice examples that test all possibilities.
4. Implement the following methods for `Point` and `LabelledPoint`:
   - `toString()`,
   - `equals(Object o)`
   - `hashCode()`.

   Short lecture needed before moving on to the next question.
5. Implement an *abstract* class `Figure`. This class contains the following *abstract* methods
   - **void** `translate(`**double** `dx,` **double** `dy)` that translates a figure using the vector `(dx,dy)`
   - `Point getBarycenter()` that returns the barycenter of the figure (the barycenter of a circle is its center, the barycenter of a rectangle is the intersection of its diagonals).
6. Implement two subclasses of the `Figure` class: `Circle` and `Rectangle`.
7. Write the `clone()` method for the classes `Circle` and `Rectangle`. Test and make sure the clones are behaving as you want.