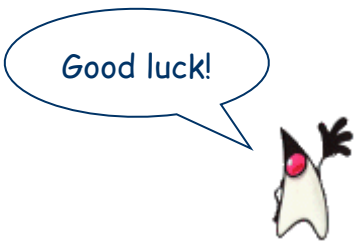
Mittuniversitetet
MID SWEDEN UNIVERSITY

Assignment 4

Java for C ++ programmers, 7,5 hp

Objective:	To be able to use classes in the Java Collection Framework.
To read:	Lecture 5
Tasks:	1
Submission:	Inlämningslåda 4 at Moodle



Task 1

In this task, you will start from your Assignment 3.
Create a new Java project in Eclipse. Your classes should belong to the package: `dt062g.studentid.assignment4` where studentid is your username in the student portal / Moodle.
The program should be completely text-based (use System.in and System.out for input and output from / to the user).
NOTE! Inputs and outputs may only be made in the class containing the main method (unless otherwise stated in the text).

Each class and interface (interface) that you create must be documented with proper comments. Ex.

```
/**
* A short description (in Swedish or English) of the class.
*
* @author  Your Name (your student id)
* @version 1.0
* @since   yyyy-mm-dd (last edited)
*/
```

Shape

In this class, you should change the instance variable `points` so that instead of being of type `Points[]` (an array) instead, `ArrayList<Point>` is the type. Update constructors and methods that use points to work with the new type. Pay particular attention to the `addPoint` methods so that they work according to previous requirements.

Drawing

Now it is time to create a class that can handle a collection of `Shape`-objects. This class will be called `Drawing` and is intended to be used in an upcoming assignment to draw a diagram consisting of many different figures. The class should:

- Implement the interface `Drawable`.
- Have the following instance variables:
  - `name` of type `String`
  - `author` of type String
  - `shapes` of class that implements the interface `List<E>`
    Select the appropriate access modifiers for these.
- Have the following public constructors:
  - `Drawing()`
  - `Drawing(String name, String author)`
    The constructor that takes two arguments will assign the values to instance variables given as arguments. Regardless of which constructor is used, list `shapes` will be created.
- Have the following public methods:
  - Set and get methods for the instances variables `name` and `author`.

  - addShape that takes a Shape-object as an argument and adds it to the list of figures. If the argument sent to the method refers to `null`, the figure will not be added to the list.
  - `getSize` which returns number of figures that the drawing contains (the size of the list).
  - `getTotalCircumference` which returns the sum of the circumference of all figures. If the circumference of a figure cannot be calculated, this can be deleted from the sum.
  - `getTotalArea` which returns the sum of the area of all figures. If the area of a figure can not be calculated, this can be deleted from the sum.
- Implement all abstract methods from `Drawable`.
  - `draw` will draw the entire drawing to standard out (`System.out`). Begin with a printout of the drawing's name and author followed by that is returned by each figure's `toString`.
  - `draw(Graphics g)` you can leave completely empty. This method is used in a later assignment (about 2D graphics).
- Override the method `toString` and return a `String` of data about the drawing. The data to be returned is the name, author, size, total circumference and area of the drawing (same rules as for the methods `getTotalArea` and `getTotalCircumference`). For the data that is not (for circumference and area) use 0 (zero). It can be for example a drawing containing two figures, but for both of these figures no endpoint is set. Then 0 (zero) should be used for its circumference and area. Ex:

  Drawing[name=Java; author=J. Gosling; size=2; circumference=0; area=0]
  Drawing[name=Mona Lisa; author=L. da Vincis; size=5; circumference=846.5999999999999; area=19340.5]
  Drawing[name=Emptyness; author=V. Oid; size=0; circumference=0; area=0]

Assignment4

This class is attached to this task description. The class demonstrate the use of the other classes in this task. The class should be used without any modifications to the code. What you can change is package declaration, author, version, and date information. You must also change in the try-catch that exists so that the correct name of your exception class is used.

On the next page, you will see a printout of running the test program.

```
Creating a new empty drawing...
Drawing[name=; author=; size=0; circumference=0.0; area=0.0]

Setting name and author...
Drawing[name=Mona Lisa; author=L. da Vincis; size=0; circumference=0.0;
area=0.0]

Adding 5 shapes with no end points...
Drawing[name=Mona Lisa; author=L. da Vincis; size=5; circumference=0.0;
area=0.0]

Size is: 5
Adding a null shape...
Size is: 5

Adding end point to all shapes...
Total circumference is: 846.5999999999999
Total area is: 19340.5

Draw the drawing...
A drawing by L. da Vincis called Mona Lisa
Circle[start=100.0, 100.0; end=175.0, 100.0; radius=75.0; color=ffe0bd]
Circle[start=75.0, 75.0; end=85.0, 75.0; radius=10.0; color=0000ff]
Circle[start=125.0, 75.0; end=135.0, 75.0; radius=10.0; color=0000ff]
Rectangle[start=95.0, 100.0; end=105.0, 115.0; width=10.0; height=15.0;
color=000000]
Rectangle[start=55.0, 130.0; end=145.0, 140.0; width=90.0; height=10.0;
color=ff0000]

Area of nose=150.0
Changing end point of nose to 100.0, 110.0
Area of nose=50.0
```