Lab 1: Model a domain

Task 1: *Graphics.*

- Write a *Hello World* program in the graphical mode.
- Define the package *DrawingTool_00*
- Define a class *TestDrawingTool_00*
- Define a class *DrawingArea*
- Open *TestDrawingTool_00.java* and *DrawingArea.java* provided by Emil.
- Don't drag the java files into Eclipse!
- Copy their contents into the source code of the files generated by Eclipse.
- Change the graphical objects and observe what happens. Use Run/Run.
- Find out how to draw a polygon ().

Task 2: *Provide a domain for which you will program beautiful graphics.*

- Describe your application domain:
  - What is the title of your domain?
  - Which kinds of objects are involved? (Think of parts and wholes as well as details, decorations, and variations.)
  - How do they relate?
  - Provide a UML class diagram of your domain (either drawn by a tool or by hand; just provide a picture of it).
- The Implementation:
  - Define the new package *DrawingTool*.
  - Provide a main program like in Task 1 (use the code in *TestDrawingTool_00.java* (without _00) and *DrawingArea.java*).
  - For each of your identified objects, provide a class (by New/Class separate files are generated for each class with the names of the classes).
  - Each class should provide only the relevant properties, a constructor, and a draw()-method.

- In the method *paintComponent* of the class *DrawingArea* you construct one object instance of exactly one of your classes (e.g. called Scene). This is where you enter your own graphics application.
- In your Scene class you provide a method called draw() with no arguments:
  - Here, for each of your classes you provide object instances.
  - After that, you call their draw()-methods. That is, each class has also its own draw()-method.
  - Does the result meet your expectations?
  - Where should components of objects be combined graphically?

Software Quality

a) Identifiers are in English.

b) Identifiers are meaningful.

c) Variable identifiers begin with a small letter. Multiple words composed as CamelCase.

d) Identifiers for classes and interfaces begin with a capital letter. Multiple words composed as CamelCase.

e) Identifiers for constants consist only of uppercase letters. Multiple words composed by underline.

f) Left curly braces not in a new line. New line after left curly braces.

g) New line after right curly braces.
Exception: keyword else is in the same line.

h) Logical sections within a method have a comment as a heading.

i) Each block level is horizontally tap-indented by one level.

j) There is a blank line between methods.

k) There is a blank line between classes.

l) Classes and interfaces are separated by a blank line of import and package statements.

m) No more than one blank line in a row.

n) Order within a class or an interface:

1. properties (constants and variables)
2. constructors
3. getter and setter for properties
4. other methods