Lab 4: State Machine

Task 1: *Frame for state machine*

- Create a new sub-package of your package, e.g. called *graphicState*.
- The design pattern of a *State* is to be implemented within that sub-package.
  - It consists of the abstract class *State* and of as many classes as there are states you wish to distinguish.
  - Each such state class extends *State* in accordance to the design pattern.
  - Each state class implements its own way how to work within each state.

Task 2: Add state machine to your project, to control different (graphical) states

- The state machine is to be integrated within your project.
- For this purpose, the *Scene* class should have a new property like:
  private static State graphicState;
- This property needs to be initialised with one of the states in the constructor of *Scene*.
- The GUI buttons change the state by calling methods of the class *Scene*.

Task 3: *Provide*

- Update your UML diagram.
- Especially:
  - Consider the state machine in the new package.
  - Class *Scene* connects to the state machine via graphicState.

Task 4: *C*

- Provided that you would have to solve Lab 1 to Lab 4 within a C program.
- What do you think, which kinds of problems might occur?
- Be prepared to tell your professor in the last lab.

Software Quality

a) Identifiers are in English.

b) Identifiers are meaningful, but not too long.

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, but only if required
   4. other methods