Imperial College
London

EE2-12 Software Engineering 2: Object-Oriented Programming

Week 2 - Classes and Objects II – Lab*

Sahbi Ben Ismail (s.ben-ismail@imperial.ac.uk)

Points

Update the class `point` with:

- A default constructor with no parameters initialising the coordinates to 0 and an overloaded constructor with two parameters for the initial value of the coordinates.
- Where appropriate, the call by `const` references for member functions.
- Where appropriate, `const` member functions.

Organise your code using headers and separate source files.

Use a basic `Makefile`, similar to the one presented in the lecture, to build your application (compiling, linking, executing, etc ...)

Triangles

Define a class `Triangle`, which attributes are the three points delimiting it.

Define a constructor which takes three points as arguments. Define a member function `perimeter` which returns the perimeter of the triangle object on which it is called. Define a member function `translate` which takes as argument a point (representing a vector) and changes the state of the triangle translating it by the vector.

*Lab content originally written by Max Cattafi.

Write a main to test the class, check for instance that the perimeter of a triangle is the same before and after a translation.

Organize the source for classes `point` and `triangle` using headers and separate source files.

Update the basic `Makefile` with the appropriate rules to take both classes `point` and `triangle` into consideration.

Remove the default constructor from `point` and make sure everything still works in `triangle`.

Operators

- Add an operator `<` to `point` (a point is less than another if it is closer to the origin). Pass the arguments by const reference and use a getter method for the distance from the origin.
  - Write a function which takes as argument a vector of points and returns the index of the one which is closest to the origin. Use the `<` operator.
  - Write a `main` to test this function, the input points should be read from a text file.
- Add an operator `==` to `point`, do not use getters.
  - Devise and write a (meaningful) function which makes use of the `==` operator defined on `point`.
  - Write a `main` to test this function (e.g. detecting duplicates in a vector of points).
- (Optional) Add the same operators to the class `triangle`, where the `<` comparison is based on perimeter instead of distance