Faculty of Science and Engineering

COMP125 Fundamentals of Computer Science
Workshop Week 5

Learning outcomes

Following are this week's learning outcomes,

a. Design classes

b. Implement classes (write class definition), including,

   (a) instance variables
   (b) getters and setters
   (c) constructors, that call setters
   (d) `toString()`, `compareTo()`, `equals()` methods

c. Understanding the `this` keyword

d. Unit testing methods in classes

You need to finish an online quiz by 23:45 (11:45pm) Sunday 30th August, 2015

Questions

1. (Pen and Paper exercise) Draw class diagrams for classes that encapsulate the following real life entities. Add up to three data members for each class. Select the three most important attributes if you think a class has more than three attributes. Describe your design in terms of a UML class diagram as shown in week 4 lecture.

   a. Cylinder
   b. Book

2. (a) Consider the following class definition,

```
public class Time {
        public int hour, minute, second;
}
```

Explain why it's a bad idea for the data members to be public, by writing a client that is malicious and assigns invalid values to the data members of `Time` object.

(b) Solve the problem of `public` data members getting assigned invalid values by first changing visibility of the data members of class `Time` to `private` and then adding getters and setters. The setter for `hour` should constrain the passed value in the range [0, 23]. That is, if the passed value is less than 0, `hour` should become 0, otherwise if the passed value is more than 23, `hour` should become 23, otherwise `hour` should become the passed value. Similarly, the setters for `minute` and `second` should constrain the passed value in the range [0, 59].

(c) Declare and instantiate an object `myTime` of class `Time` written in part (b). This code is to be located outside class `Time`. Assign values to the data members such that it represents the time 19:30:45 (half past seven in the evening and another 45 seconds).

(d) Declare and instantiate an object `yourTime` of class `Time` written in part (b). This code is to be located outside class `Time`. Assign 95 to `hour`, -78 to minute, and 55 to second. Display all data members on the console. What time would `yourTime` represent?

(e) List the mistakes (syntactical and logical) in the following constructor for class `Time` -

```
public void time(int h) {
        hour = h;
        minute = 0;
        second = 0;
}
```

(f) Add two constructors to class `Time` with the following requirements:

- A constructor that is passed three parameters, one for each data member.
- A constructor that is passed two parameters, for hour and minute, and sets seconds to 0.

(g) Assuming the two constructors have been added to class `Time` according to previous part. Will the following program run successfully, or result in a compilation error? Explain your answer. Also, if there is a compilation error, what should be done to fix it?

```
Time ourTime = new Time();
```

3. Write a class definition for a `Cube`. A cube is a 3-dimensional object, enclosed by 6 equal squares. A cube is characterized by the length of each side (all sides have equal length).

a. Correct class header.

b. Data members with appropriate visibility and data types.

c. Getters

d. Setters

e. Constructors

   (a) With no parameters. Length of each side is set to 0.

   (b) With one parameter for length of each side.

f. A method `volume()` that returns the volume of the cube given by $side * side * side$.

g. The method `toString()` that returns the `String` representation of object of class `Cube`. For example, for a cube of sides 2.5, the method should return "`cube of size 2.5`".

4. **The `this` keyword**
   The class `Box` contains a setter but the assignment operation is ambiguous. It's not clear whether the parameter is being assigned to the instance variable or vice versa. Explain the problem and fix it by *qualifying* the appropriate side (left or right) to correctly indicate which `capacity` is the instance variable and which is the parameter.

5. **The `compareTo` method**
   Complete the `compareTo` method in class `Box` such that it returns,

   - 1, if the calling object has a higher capacity than the parameter object
   - -1, if the calling object has a lower capacity than the parameter object
   - 0, if the calling object has the same capacity as the parameter object

6. **JUnit test**

   a. Run the JUnit test `testVolume` in JUnit test class `SphereTest`. Correct the method `volume` in class `Sphere` so that the test passes. The description for the method's requirements is provided as method comment.

   b. Complete the JUnit test `testSurfaceArea` in JUnit test class `SphereTest` so that it ensures the correctness of method `surfaceArea` from class `Sphere`. You can get the values for surface areas of spheres with different radii by Googling for *"surface area of sphere with radius ...."*.

7. **Weekly Task Week 5 (to be completed on iLearn)**

   Answer the quiz Week 5 Task available on iLearn by 23:45 (11:45pm) Sunday 30th August. The quiz will ask questions about last week's lecture and about this week's workshop.

   You have two attempts to do the quiz. There is no penalty if you re-attempt but note that the questions or the answer candidates may be slightly different.

   Don't leave the quiz submission to the last minute and don't underestimate the time that you may need to solve the quiz, especially if you haven't done the exercises of this workshop.

Supplementary exercises (take-home exercises)

1. **Email address**

   Complete the definition of class `EmailAddress` based on the requirements provided as comments in the class, and run the the client `EmailAddressClient`.

2. **Bank account**

   Define a class representing a bank account with methods to check the balance, to deposit and withdraw funds and to test whether the account is overdrawn.

   In project `comp125Banking`, there is an outline implementation of the class along with a set of tests for the different methods. Use this as the basis for your implementation. Read also the in-code comments for the details of what each method is expected to do.

   The first thing you need to do is decide how the `BankAccount` class is going to store its data – the bank balance. You need to define an instance variable of the appropriate type in the class. (Label your instance variable `private`.)

   Then you need to write the body of each of the methods. At each point, run the tests to see if what you've written works. Try to work on the tests from top to bottom, getting one working after another. You can use the debugger to walk through your program if it's not doing what you expect – remember to place a breakpoint in your class to stop execution when running under the debugger.

   When you've passed all the tests, complete the `public static void main(String[] args)` method in `BankAccountClient` to make use of the bank account class. Make a bank account, add 42.5*totheaccount*,*thenwithdraw*80.6 from it, display if it's overdrawn (`true`) or not (`false`) and display the balance. You can include such a `main` method inside your `BankAccount` class.

   Please note that `BankAccountTest` uses an `assertEquals` assertion that has three parameters. The syntax is;

   ```
   1        assertEquals(expectedDouble, returnedDouble, tolerance);
   ```

   The assertion is successful (passes) if the `expectedDouble` and `returnedDouble` differ by at most `tolerance`. That is, $Math.abs(expectedDouble - returnedDouble) \leq tolerance$.

3. **The `equals` method**
   Complete the `equals` method in class `Box` such that it returns,

   - `true`, if the calling object has the same capacity as the parameter object
   - `false`, if the calling object has a different capacity from the parameter object