FIT9131 Programming Foundations

Week 4 Exercises

You should have downloaded the Assignment 1 specification by now, and started working on it. Make sure you understand exactly what is required.

A. Homework checklist

By this week you should have completed the following:

- Lab exercises from weeks 1, 2 and 3.
- Read chapters 1, 2 and 3 of the textbook.
- Read the document titled "***Java Coding Standards for FIT9131***".

B. Exercises for Week 4

1. Coding Standards

Last week for homework you were asked to read a document titled "https://lms.monash.edu/mod/resource/view.php?id=8902756". This document explains how you are expected to write your Java code in this unit. There are a few cases where these standards do not agree with what is shown in the textbook *Objects First with Java*. Where this is the case, ***you must follow the FIT9131 standards***.

Consider the following methods. Identify any *violations* of the FIT9131 coding standards. For each violation, state the standard that has been violated ***AND*** how the code should be changed so that it complies with our FIT9131 coding standards.

a)

```
public void DisplayPrice(boolean is_student, int AGE)
{
    int base_price=8;
    if(AGE>12)
    if(is_student)
       System.out.print(base_price*0.8);
    else
       System.out.print(base_price);
    else
       System.out.print(base_price*0.5);
System.out.print(" dollars");
}
```

b)

```
double interestDue (int balance, double interestRate) {
   return balance * interestRate;
}
```

c)

```
public void insertMoney(int amount)
{
    if(amount>0) {
        balance=balance+amount;
    }
    else {
        System.out.println("Use a positive amount: ");
        System.out.println(amount);
    }
}
```

2. Writing code for a Java Class from scratch

Some students are finding it difficult to understand the various components of a Java class. This is partly because BlueJ provides quite a bit (sometimes too much) of help in writing code. The following exercise lets you practice writing a small class without using the template code that BlueJ provides. Writing the code from scratch will also help you better understand the different components of a Java class.

You are to write the complete code for a class which matches the following **Class Diagram**:

Person
String : **name** int : **age**
Person() **Person**(String : newName, int : newAge) void **display**() int **getAge**() String **getName**() void **setAge**(int : newAge) void **setName**(String : newName)

Follow these steps (in the exact order as shown):

a) Start a new BlueJ project, name it **Week4**.
b) Create a new class, name it **Person.**
c) Open the code for the **Person** class in the BlueJ editor. **Immediately delete *ALL* the code in the editor**.

d) Now start adding in the code for the class, **one step at a time**. ***Compile the class after each step, to check that the code is syntactically correct***. ***Do not try to do all the steps at once!*** Keep the code simple - there is no need to write code for any error-checking.

1) create the **class header**, then add the two curly braces for the (empty) class body.
2) add the 2 **attributes** (name and age).
3) add the **default constructor**. In the body of this constructor, initialise the two attributes to some sensible values.
4) add the **other constructor** (with the 2 parameters). In the body of this constructor, initialise the two attributes using the two parameters.
5) add the **`getName`**`()` accessor method. In the body of this method, return the name attribute.
6) add the **`setName`**`()` mutator method. In the body of this method, set the name attribute to the parameter.
7) repeat steps (5) & (6) for the **`getAge`**`/`**`setAge`**`()` methods
8) add the **`display`**`()` method. In this method, display the two attributes in a sensible format.

Create some objects from the **`Person`** class, and test their methods to make sure they all work as expected.

3. Checking for coding standard violations

Check the code you wrote in Q.2 for the **`Person`** class. Can you find (and fix) any FIT9131 Coding Standard violations?

4. Creating an object in code

After a Class is created, you need to create an object from it in order to execute or perform any action. In our programming exercises so far, we have been using the ***BlueJ graphical interface*** to create objects and execute code. In this exercise we will practice writing the program code to create objects and invoke methods on those objects – we will do this using the ***BlueJ Code Pad***.

We will use the **`Person`** class from Question 2.

The syntax for creating a new object is as follow:

```
ClassName objectName = new Constructor();
```

These will always be the ***SAME*** as the constructor has the ***same*** name as the class.

```
e.g. Person aPerson = new Person();
     Person aPerson = new Person("Frank", 29);
```

(question continued on next page…)

a) Type the following into the *BlueJ Code Pad*:

```
new Person()
```

b) Double-click the red box which appears. Can you explain what happens?

____

c) On the *BlueJ Code Pad* again, now type the following:

```
Person aPerson = new Person ("Kim", 23);
```

d) Can you explain what has happened after you hit <**enter**> for the previous line of code?

____

e) On the *BlueJ Code Pad* again, now type the following:

```
aPerson.getName()
```

f) Can you explain what has happened after you hit <**enter**> for the previous line of code?

____

5. Clearing the BlueJ terminal window

The following will be useful for you to know. It is a link to some information on clearing the screen in BlueJ.

Go to:

http://www.bluej.org/faq.html

And then click on the topic:

How do I clear the terminal?

Please note, this method works in BlueJ but it will not work in every environment.

While you are on that webpage, check out any other useful tips which may be useful.

C. Homework

- Finish the Lab exercises for week 4.
- If you are having any problems with the Weekly exercises, you must attend the Helpdesk sessions.
- Read Chapter 4 of the textbook.

D. Pre-Lab tasks to be assessed in Week 5

Do the following exercises from the textbook. These will test and re-enforce your understanding of the concepts covered in the week 4 lecture using the **`clock-display`** example.

a. 3.10–3.14 (relational and logical operators).

b. 3.19 (string concatenation)

c. 3.21, 3.23, 3.26 (modulo operator)

d. 3.27, 3.29, 3.31-3.32 (constructors)

e. 3.56 3.57 (explaining and extending code)

Reminder: these exercises should be submitted via Moodle at least 48 hours (2 days) before your lab class. To submit your tasks, write your answers in a document and you can cut and paste code sections where appropriate.