FIT9131 Programming Foundations

Week 3 Exercises

The specification for assignment 1 will be available on Moodle during Week 3. Please download and start working on it as soon as possible.

A. Homework checklist

- By this week you should have completed the following:
- Lab exercises from weeks 1 and 2.
- Read Chapters 1 & 2 of the textbook, *Objects First with Java*, Barnes & Kölling.

Don't forget that from now on, you are expected to have purchased the textbook, and no more scanned chapters will be posted on the FIT9131 website. You will need the textbook to work on the lab exercises, as well as studying the examples from the lectures.

Online Help Desk sessions for FIT9131 start running this week. It is very important that if you are having any problems with understanding any of the work (lectures/labs) in Week 1 and Week 2, you must visit the Help Desk to seek help now. Check the FIT9131 Moodle site for more details (and the Help Desk timetable).

B. Exercises for Week 3

1. Expressions and operators

What will be ***returned*** from the evaluation of the following ***Java expressions***? Work out your answers on a piece of paper (ie. do not type them into BlueJ).

34 + 3.5 _____

13 % 7 _____

70 / 2 != 35 _____

3 + 4 * 2 _____

11 / 4 _____

7 % 13 _____

4.75 * 2 _____

11 / 4.0 _____

"Java".length() _____

7 % 0 _____

Check your answers to the exercises above using the **Code Pad** in BlueJ. The **Code Pad** can be used to evaluate simple expressions. **Code Pad** is normally hidden. To bring it into view, select the **Show Code Pad** option in the **View** menu item or use the shortcut: **Ctrl-e**.

Are your answers correct? If not, make sure you understand why. If you are unsure how to use the code pad, review the BlueJ Tutorial available on Moodle under Week 1.

2. Some more advanced operators

Study the following pieces of Java code carefully. Work out (on paper) what will be displayed on the screen if each piece of code is executed. Write down your answers.

a.
```
int total = 10;
total -= 2;
System.out.print(total);
```

b.
```
int total = 0;
total += 1;
System.out.print(total);
```

c.
```
int total = 0;
total++;
System.out.print(total);
```

d.
```
int total = 0;
++total;
System.out.print(total);
```

e.
```
int total = 0;
System.out.print(total--);
```

f.
```
int total = 0;
System.out.print(--total);
```

3. Writing Methods

When programming in Java, methods are crucial as they define what an object can do. The basic syntax for a ***method header*** is as follows:

(Visibility modifier) (return type) (method name) ([optional parameters])

Visibility Modifier – *private* or *public* (or *protected* – we will learn about this later)

Return Type – *void*, *int*, *boolean*, *char*, *float*, *double*, *String*, or any other Class Object types

Method Names – note that a method ***CANNOT*** have the same signature as another method in the same class.

Parameters – can potentially define any number of parameters as required.

Some examples of method headers (and explanations):

```
public void sampleMethod()
```

Publicly accessible method which returns nothing and accepts no inputs.

```
public void sampleMethod(int inputA, int inputB)
```

Publicly accessible method which returns nothing but accepts two inputs

```
private String sampleMethod(int inputA, int inputB, String inputC)
```

Private method which returns a String as an output and accepts three inputs

```
private Student sampleMethod()
```

Private method which returns a **`Student`** object as an output and accepts no inputs

We will now practice writing methods to test the answers to the code reading exercises in **Q.2**.

First, create a project in BlueJ named "**`Week3`**", and then create a class in it named "**`Question2`**", and then create methods (one for each of the six code segments) to test the code – i.e. **put each piece of code inside a separate method**. Then invoke each method to see if the output matches your own answer above. **Do NOT COPY and PASTE the code.**

4. Selection

Work out (on paper) what will be displayed on the screen if each piece of code below is executed. Hint: is there anything missing from the code?

a.
```
char seatLocation = 'B';
int price = 0;
switch (seatLocation)
{
     case 'A':
          System.out.println("Front row");
          price = 100;
          break;
    case 'B':
          System.out.println("Stalls");
          price = 80;
          break;
    case 'C':
          System.out.println("Balcony");
          price = 55;
    default:
         System.out.println("Unknown seat location");
         price = 0;
         break;
}
System.out.println(price);
```

b.
```
char seatLocation = 'C';
int price = 0;
switch (seatLocation)
{

    case 'A':
          System.out.println("Front row");
          price = 100;
          break;
    case 'B':
         System.out.println("Stalls");
          price = 80;
          break;
    case 'C':
         System.out.println("Balcony");
         price = 55;
    default:
        System.out.println("Unknown seat location");
        price = 0;
        break;
}
System.out.println(price);
```

c.
```
int amount = 0;
int balance = 50;
if (amount >= 0)
{
    balance = balance + amount;
    System.out.println(balance);
}
else
    System.out.println("Invalid amount:  " + amount);
```

(What will be displayed if the curly braces are not included?)

In the same way as in question 3, check your answers by creating a class named "**`Question4`**" in the same "**`Week3`**" project. Create methods (one method for each piece of code above) then invoke each method to see if the output matches your own answer.

5. Understanding Variables (Fields, Parameters, Local Variables)

Consider the code for the **`TicketMachine`** class in the ***better-ticket-machine*** project, then answer the following questions:

a. What sort of variable is each of **`cost`**, **`balance`** and **`amountToRefund`**?

b. What is the *scope* of the variable **`cost`**? _____

c. What is the *scope* of the variable **`balance`**? _____

d. What is the *scope* of the variable **`amountToRefund`**? _____

e. What is the *lifetime* of the variable **`cost`**? _____

f. What is the *lifetime* of the variable **`balance`**? _____

g. What is the *lifetime* of the variable **`amountToRefund`**? _____

6. Accepting input from the keyboard

The following exercise will help you understand how to obtain input from the user via the keyboard. Input from the keyboard in Java is obtained using an object of the **`Scanner`** class. The **`Scanner`** class has several methods that you can use to read in data, depending on the type of that data.

Using the same "**`Week3`**" project that you created in Q.3 above, add a new class called **`ScannerExample`** and type in the following code to define the class.

```
import java.util.Scanner;

public class ScannerExample     // constructors are omitted for brevity
{
        public void testScanner()
        {
                int age;
                char status;
                String name;
                Scanner console = new Scanner(System.in);

                System.out.println("Welcome to the Scanner example!");
                System.out.print("Press enter to continue ... ");
                console.nextLine();

                System.out.print("Enter your name: ");
                name = console.nextLine();
                System.out.println("Hello, " + name);

                System.out.print("Postgrad or Undergrad - P/U: ");
                status = console.nextLine().charAt(0);
                System.out.println("Your status is " + status);

                System.out.print("Enter your age: ");
                age = console.nextInt();
                System.out.println("Your age is " + age);
        }
}
```

Compile and create an object of the **`ScannerExample`** class. Then call the **`testScanner()`** method, type in your responses to the prompts on the screen and press the <**enter**> key after each input. Try and work out what each line of code in the method defined above does.

Try this exercise several times entering valid data and then try with invalid data to see what happens (e.g. type in letters when asked for the age or type in several characters when asked for status).

C. Homework

1. Finish the lab exercises for week 3.

2. Read the document titled *Java Coding Standards for FIT9131*. This document can be found under the Week 2 Topic area on Moodle. Study the document carefully; we will be using it in one of the lab exercises next week.

3. Read chapter 3 of the textbook, *Objects First with Java*, Barnes & Kölling.

4. Consolidation of concepts:

   Do the following exercises from the textbook. These will test and re-enforce your understanding of the concepts covered in the week 3 lecture. The code is using the ***better-ticket-machine*** project from the ***Chapter02*** folder of the ***Sample Projects***.

   2.46 (exploring behaviour of a class)

   2.47, 2.52 (conditions and selection)

   2.58, 2.59, 2.62 (local variables)

D. Pre-lab tasks to be assessed in Week 4

1. Complete these exercises from the textbook (which use the ***book-exercise*** project from ***Chapter02*** folder): 2.83 to 2.85 and 2.87 to 2.90.
2. On the following page is the code for a simple (and slightly incomplete) class called **`Car`**. Examine this code and find a single example of ***each*** of the following:
   a. a default constructor (sometimes called a “non-parameterised constructor”).
   b. a parameterised constructor.
   c. a field and its corresponding accessor (get) and mutator (set) method.
   d. a comment.
   e. a method header.

Reminder: these exercises should be submitted via Moodle at least 48 hours (2 days) before your class.

To submit your tasks, write your answers in a Word document (you can cut and paste code sections where appropriate) and then submit it via the link on Moodle.

```java
/**
 * Class which creates stores details of a Car
 *
 * @author Mark Creado
 * @version 1.0.0
 */
public class Car
{
    private String model;
    private int year;

    /**
     * Constructor for objects of class Car
     */
    public Car()
    {
        model = "Ferrari";
        year = 2019;
    }

    /**
     * Non-Default Constructor for objects of class Car
     */
    public Car(String newModel, int newYear)
    {
        model = newModel;
        year = newYear;
    }

    /**
     * Method to display the state of the object
     */
    public String displayCar()
    {
        String carState = model + " " + year;
        return carState;
    }

    /**
     * Accessor Method to obtain the model of the car object
     *
     * @return The model of the car
     */
    public String getModel()
    {
        return model;
    }

    /**
     * Accessor Method to obtain the year of the car object
     *
     * @return The year of the car
     */
    public int getYear()
    {
        return year;
    }

    /**
     * Mutator Method to set the model of the car object
     *
     * @param newModel The model of the car
     */
    public void setModel(String newModel)
    {
        model = newModel;
    }

    /**
     * Mutator Method to setthe year of the car object
     *
     * @param newYear The year of the car
     */
    public void setYear(int newYear)
    {
        year = newYear;
    }
}
```