Mobile Computing

Laboratory 2

1 Strategy

One of the best ways to improve your programming and to have more fun when programming is to program with someone else. This week we're going to experiment with pair coding. How pair coding works is that every pair of students has only one computer. The students must then take turns coding while the other watches and edits. This may not sound super productive, but trust me, it's fantastic. Now given that a lot of you are working from home, there are a number of ways of accomplishing a similar outcome. The main thing is to make sure that you collaborate and communicate within your pair. You'll be exploring some ways to accomplish this in today's class discussion.

Today's lab has a TON of questions. Like.. just too many questions. I'm not even sorry! What I want you to do is divide into groups of two and alternate typing on questions – if person 1 is typing for Lab 2a, person 2 should be typing for Lab2b. The person who is not typing should be watching and discussing. This improves the coding and communication of both students. Then, both of you should submit the same program to the moodle marker. Note that if you don't submit, you won't get marks even if your partner did submit. Now, because of the pandemic a lot of you are working from home. So get to the class discussion for some thoughts on how to do this.

Make sure to put BOTH student numbers in a comment at the top of the file you submit so I can allow these pairs in the plagiarism checker.

2 Creating a project

In this section, we will outline how to create a project in the Eclipse IDE (Integrated Development Environment)

- Launch Eclipse
- Eclipse will then prompt you for a location for your workspace. You can accept the default location, which is in your user folder. Remember this location as you will need it later.
- In Eclipse, create a new Java Application, by selecting File - New - Java Project
- In the resulting dialog box, enter your project name. Consider using something like Lab2a as it will make it easier to find your code later.
- Eclipse will now display your project in the left hand pane (Unless you have a splash screen open, in which case you can safely close that).
- Expand your project by clicking on the + icon, exposing the src folder.
- Make a new class by right clicking on the src folder and selecting New - Class.
- In the resulting dialog box, you must fill in the class name. In this case, the class will be called Program. When you're done creating the class, it will appear in the editor window so that you can make changes to it.

3 Lab2a - Using Methods

1. In this lab, you must create a static method called `multiply`. This method must accept two integers as input and return the product of the two integers.
2. The method header is shown below to help you.

```
public static int multiply(int x, int y){
```

3. In your main method, you should use your multiply method to calculate and print out the product of two numbers entered by the user.

3.1 Example Input

```
3
7
```

3.2 Example Output

```
21
```

4. Submit a zip file containing the program to the marking system. You can find the source files in the folder that you selected as your workspace when eclipse started. Go to this folder in your file explorer, then select the folder corresponding to your project (Lab2a), then select the src folder. You can then compress the java files into a zip file from there.

4 Lab2b - Methods and Arrays

1. You must make a static method called `scale`. This method must accept an array `arr` containing integers, as well as an integer `s`. It must return a new array which contains the items from `arr` multiplied by `s`.
2. The method header is shown below to help you.

```
public static int[] scale(int[] arr, int s){
```

3. You should make a main method that looks as follows:

```
public static void main(String args[]){
   int[] toSend = {4, 9, 2, 7};
   int[] result = scale(toSend, 4);
   System.out.println(result[1]);
}
```

4.1 Example Output

```
36
```

5 Lab2c - More complex methods

1. In this lab, you must create a static method called `getX`. This method must accept a String as input and return an integer. The String will be of the form `x,y` (eg `4,3`), and the method must return the x coordinate which in the example would be 4.
2. You must also create a static method called `getY`. This method must accept a String as input and return an integer. The String will be of the form `x,y` (eg `4,3`), and the method must return the y coordinate which in the example would be 3.
3. You can assume the x and y coordinates will be single digit integers.
4. Lastly, you must create a static method called `getXDistance`. This method must accept two Strings in the coordinate form described above. It must use the `getX` method to get the x coordinate from each String, and must then return an integer representing the distance between the two x values. This distance must be a positive number. For example, if your `getXDistance` function receives two Strings `2,3` and `9,2`, it should return `7`.

6 Lab 2d - Working with Arrays

This Lab will require the use of an array.

1. This program must first read 10 numbers into an array, using a for loop.
2. It must then read in another number representing the index of one of the numbers, and must print out that number.

6.1 Example Input

```
7
9
2
5
3
8
1
6
9
4
3
```

6.2 Example Output

```
5
```

Note that the last number entered (3) indicates that we need to print out the number in position 3 in our array. For the numbers presented as input, this would be 5.

7 Lab 2e - Array Sizes

1. This program must first read a number representing the size of the desired array.
2. It must then create an array of the required size.
3. It must then read the correct number of numbers into the array in order to fill up the array, using a for loop.
4. It must then read in another number representing the index of one of the numbers, and must print out that number.

7.1 Example Input

```
4
2
9
4
3
1
```

7.2 Example Output

```
9
```

8 Lab 2f - Loops

1. This program must first read a number representing the size of the desired array.
2. It must then create an array of the required size.
3. It must then read the correct number of numbers into the array in order to fill up the array, using a for loop.
4. It must then print out the numbers in reverse order.

8.1 Example Input

```
4
2
9
4
3
```

8.2 Example Output

```
3
4
9
2
```

9 Lab 2g - Arrays of objects

1. This program must print out the Strings a user has entered in reverse order. Take a look at the example output if you are unsure what this means.
2. The program must first read in 10 Strings.
3. It must then print out the Strings in reverse order.

9.1 Example Input

```
One
Two
Three
Four
Red
Blue
Green
Yellow
Alpha
Beta
```

9.2 Example Output

```
Beta
Alpha
Yellow
Green
Blue
Red
Four
Three
Two
One
```

10 Setting class variables

Constructors are methods that are called when your class is created. The constructor is a method that has the same name as the class it is in. An example is shown below.

```
public class Student{
   int studentNumber;
   String name;

   public Student(String n, int no){
       name = n;
       studentNumber = no;
   }
```

```
}
```

Now, to create a Student object, we would be forced to supply the constructor with a String and an int, as follows:

```
Student s = new Student("Pravesh",314384);
```

This is because the constructor forces us to supply the name and number when we create a Student object.

Not that in the constructor, `n` and `no` are local variables, and thus cannot be accessed outside the constructor. However, we would like these to be properties of the class, so that we can see them later. This is why we copy them into the variables `name` and `studentNumber` using the = operator.

Another way to read and write the local variables is to use get and set methods. An example of this in the Student class is shown below:

```
public void setStudentNumber(int no){
   studentNumber = no;
}

public int getStudentNumber(){
   return studentNumber;
}

public void setName(String myName){
   name = myName;
}

public String getName(){
   return name;
}
```

Look at the method definitions and pay particular attention to the types that are returned and accepted. Make sure you understand how these work. For example, the `setStudentNumber` method doesn't return anything, hence the `void` return type. It does however require the user to supply the number to set the studentNumber to, hence the `int no` in the brackets.

11 Lab2h - Working with Classes

Now we will begin using more than one class in our programs. For this lab, our two classes are Dog and Kennel.

11.1 Setup

1. Create a project for Lab2h

11.2 Dog

1. Create a new class in your lab3e project, called "Dog"
2. This class must have the following attributes:
   - name
   - age
3. Each attribute must have a get and a set method
4. The dog must also have a bark() method which must print the dog's name followed by a :, followed by the word "woof".
   eg. If a dog's name is snoopy, the output from its bark method will be:

   ```
   snoopy:woof
   ```

5. Finally, the Dog class must have a constructor that accepts a dog's name and age as parameters so that a Dog object can be created using a command similar to:
   `Dog myDog = new Dog("Snoopy", 15)`
6. Submit your code to the marking system

11.3 Program

1. The Program class must contain an array of dogs, storing 5 Dog objects.
2. The main method of the Program class must be used to create the dogs. Note that you have to create each Dog individually using the **new** operator. The names and ages of the dogs must be read from the keyboard. An example input is shown below:

   ```
   Snoopy
   10
   Romeo
   8
   Rex
   3
   Biff
   7
   Spot
   13
   ```

3. Note that it will be easier to read in each line as a `String` and convert the lines that are actually integers afterwards, using the following syntax:

   ```
   String test = in.nextLine();
   int age = Integer.parseInt(test); //Converts the String test into an integer
   ```

4. In the main method of your Program class, use a loop to call the `bark()` method of each Dog in the array.

5. The output that should occur for the example input is shown below:

```
Snoopy:woof
Romeo:woof
Rex:woof
Biff:woof
Spot:woof
```

6. Submit your code to the marking system.