PGCert IT: Programming for Industry

Introduction to Java

Download the source code for this lab, which is available on Canvas. For exercise one and two, you are required to create your own project. There is no source code for exercise three. You will find the code for the remaining exercises and the example code from the lecture under the `industry-lab01` folder. The `industry-lab01` folder is already set as an IntelliJ project, which allows you to open a Project from IntelliJ directly.

Before starting the exercises, don't forget to watch the video on Canvas that demonstrates how to use IntelliJ and how to open an IntelliJ project.

Exercise One: Create My First Java Program using IntelliJ

You will be using IntelliJ IDEA throughout this course. Before practising on what you have learned, let's start by doing the following:

- Launch IntelliJ IDEA and create a new Java project.
- Select the appropriate JDK from the Project SDK list.
- Don't create a project from the template.
- Select the appropriate project location and name the project **`lab01`.**
- Click Finish.

Once you have created the project, you can now create a new class.

- In the **Project** tool window, right click on the **src** folder, select **New** and then select **Java Class.**
- Name the class **`MyFirstProgram`** and click **OK**.
- You should now have the **MyFirstProgram** class in the **Project** tool window. Open the class and you should only have the following in the file:

```
public class MyFirstProgram {
}
```

- Add the following code to the class:

```
public class MyFirstProgram {
   public void start() {
       System.out.println("Hello World");
   }
```

```
    public static void main(String[] args) {
        MyFirstProgram p = new MyFirstProgram();
        p.start();
    }
}
```

You can now compile and run the class by clicking on the green arrow icon next to the class name.

Questions:

1. What does this class do?
2. Where are the classes generated and stored by IntelliJ?
3. Where do you see the messages after running the program in IntelliJ?
4. What are the different ways of running this class in IntelliJ?
5. What are some of the useful shortcut keys you found in IntelliJ?

Exercise Two: Add a New Class to an existing project

Using the **`lab01`** project you just created, create a new class called **`MySecondProgram`** in the same folder as `MyFirstProgram` class. Copy the following code to **`MySecondProgram`** class:

```
package ictgradschool.industry.introtojava.mysecondprogram;

public class MySecondProgram {
   public void start() {
       System.out.println("Hello World");
   }
}
```

Notice that you now have a compilation error associated with the package. Try to resolve the error by using IntelliJ.

Questions:

1. What happened to **`MySecondProgram`** class after you have fixed the error?
2. What are packages used for?

Modify **`MyFirstProgram`** class to the following:

```
public class MyFirstProgram {
   public static void main(String[] args) {
       MySecondProgram p = new MySecondProgram();
       p.start();
   }
}
```

You should have a compilation error associated with `MySecondProgram`. Try to resolve the error using IntelliJ. After you have fixed the error, you can run the program.

Questions:

1. What does the keyword **`import`** do in Java?
2. How would you resolve the compilation error without using the import keyword?

Exercise Three: Evaluation of Expressions

What output do you think would be produced by each of the following code fragments? (You can work on this exercise on paper)

1. `System.out.println((int)2.9 * 4.5);`
2. `System.out.println(Math.max(5,60) % Math.min(12,7));`
3. `System.out.println(0.2 * 3 / 2 + 3 / 2 * 3.2);`

Exercise Four: Programming Practice

Open the existing IntelliJ project under the folder: `industry-lab-01`.

The following questions all make use of skeleton code found in:
`ictgradschool.industry.introtojava.simplemaths.SimpleMaths.java`

Before completing each method in the `SimpleMaths` class, carefully examine the class and identify the flow of the program. You also need to think about these questions: What are the arguments and their types for each method? How is the computation being done in each method? How is the output being displayed on the console after each method call? What is the return type for each method?

1. Complete the method called `kilogramsToPounds()` making use of the following formula where: p is the weight in pounds, and k is the weight in kilograms.

$$p = k \times 2.20462$$

2. Complete the method called `convertCelsiusToFahrenheit()` making use of the following formula where: `f` is degrees Fahrenheit, and `c` is degrees Celsius. Hint: you'll need to use the divide (/) operator.

$$f = 32 + \frac{9}{5}c$$

3. Complete the method called `getCompoundInterestValue()` making use of the following formula where: `A` is the future value of the investment including interest, `P` is the principal investment, `r` is the annual interest rate (remember to convert to decimal by dividing by 100), and `t` is the investment time in years. Hint: make use of `Math.pow()` method.

$$A = P(1+r)^t$$

4. Complete the method called `getMyBMI()` making use of the following formula where: `BMI` is your body-mass index expressed as an integer, `W` is your weight in kilograms, and `H` is your height in metres.

$$BMI = \frac{W}{H^2}$$

5. Complete the method called `getSphereVolume()` making use of the following formula where: `V` is the volume in cubic centimetres, `r` is the radius in centimetres, and π should use Java's `Math.PI` constant variable. Hint: make use `Math.pow()`.

$$V = \frac{4}{3}\pi r^3$$

6. Complete the method called `convertTotalDaysIntoWeeksAndDays()` the purpose of which should be self-explanatory. Hint: you will need to make use of the divide (/) and modulus (%) operators.

7. Complete the method called `findSmallerInteger()` the purpose of which should be self-explanatory. Do not use if statements.

Exercise Five: Sorting Numbers

Complete the method `sortNumbersByAscending()` in `SortNumbers.java`. The code can also be found in `SortNumbers.java`. The method takes four integers as arguments and then displays them sorted into ascending order. Later in the course you will learn efficient ways of doing this using built-in methods, but for now use only methods you have learned in the lectures. **Do not use if statements.**

The skeleton code is found in:
`ictgradschool.industry.introtojava.sortnumbers.SortNumbers.java`

Optional Exercise Six: Calculate Elapsed Time

Complete the method `getTimeElapsed()` found in `ElapsedTime.java`. The method takes a start time and a finish time, each of which is broken down into hours and minutes and expressed as integers, and then displays the time difference between the two in minutes. You should assume a 12 hour period and that the start time is always earlier than the finish time; for example if the start time was 5 hours 30 minutes and the finish time was 5 hours 29 minutes, then 11 hours and 59 minutes must have passed. **Do not use if statements for this exercise.**

The skeleton code is in:
`ictgradschool.industry.introtojava.elapsedtime.ElapsedTime.java`

Optional Exercise: Compiling and Running My First Java Program

The code for this exercise is located in the optional folder and under the `ictgradschool.industry.introtojava.myfirstprogram` package. Based on the lecture, do the following:

- Compile `MyFirstProgram.java` using command line prompt
- Run `MyFirstProgram` using command line prompt
- Modify `MyFirstProgram` to display a different message
- Comment out the entire `main()` method, compile and run `MyFirstProgram` again

Questions:

1. What is the command for compiling `MyFirstProgram.java`?
2. What message do you see when you try the following command line?

   ```
   java MyFirstProgram
   ```

   What is the correct command for compiling and running `MyFirstProgram`?
   Hint: Look at how to compile and run a Java program **in package** from command line

3. What are the classes generated after compiling `MyFirstProgram.java`?
4. What messages do you see when you run `MyFirstProgram`?
5. After modifying `MyFirstProgram.java`, what do you have to do for the changes to take place, i.e. to display a different message?
6. What messages do you see when you run `MyFirstProgram` after commenting out the entire `main()` method?

Investigate the use of command-line arguments

Read the Command-Line Arguments article from The Java Tutorials. Investigate what command-line arguments in Java are and how to pass arguments from the command line.

Then, modify `MyFirstProgram` using the following code snippet so that the program now reads one command-line argument and displays it as part of the message on the console.

```java
public class MyFirstProgram {
      public void start(String name) {
            System.out.println("Hello " + name);
      }

      public static void main(String[] args) {
            MyFirstProgram p = new MyFirstProgram();
            p.start(args[0]);
      }
}
```

Questions:

1. What is a command-line argument?
2. How many arguments can you pass from the command line?
3. What is the command for displaying the following message on the console?
```
Hello Thomas
```
4. What is the command for displaying the following message on the console?
```
Hello Thomas and James
```