Lab 6 : Files, Exceptions and Objects

Topics covered file input/output, exceptions, creating objects, objects vs. primitives

---

Exercise 1: Write a program that will generate 15 random characters (a to z) and print them to a file called `randomChars.txt`.

Exercise 2: Write a program that takes one command-line argument `<filename>`. The program will then open and read the file given by `<filename>.txt`, read each line in the file, and sort the lines in lexicographical order (remember: `Arrays.sort(...)`). The sorted lines will be output to `<filename>-sorted.txt`.

```
> java SortFile animals
```

`animals.txt`

```
lion
giraffe
snake
zebra
antelope
warthog
```

`animals-sorted.txt`

```
antelope
giraffe
lion
snake
warthog
zebra
```

Exercise 3: Write a program that takes in three arguments `filename`, `findWord` and `replaceWord`. Your program should replace all instances of `findWord` with `replaceWord`. Note: this will require you to read and write to the same file. Make sure that you don't try to write to the file while it is still open for reading – this won't work!

Exercise 4: In Exercise 7 of Lab 5, you wrote a program that takes two command-line arguments: the score and the maximum possible score. The program will calculate the percentage score of the input numbers. Here is a sample solution for that exercise.

```
1  public class Scores {
2     public static void main(String[] args) {
3        int inputScore = 0;
4        int intputTotal = 0;
5
6        // Read the two integers from arguments
7        inputScore = Integer.parseInt(args[0]);
8        intputTotal = Integer.parseInt(args[1]);
9
10       // Print the percentage
11       if (intputTotal > 0) {
12          double percentage = calculatePercentage(inputScore, intputTotal);
13          System.out.println("Percentage: " + percentage);
14       }else {
15          System.out.print("Total must be a positive number.");
16       }
17    }
18
19    public static double calculatePercentage(int score, int total) {
20       return ((double)score / total) * 100.0;
21    }
22 }
```

What kind of things can go wrong in this program? Revise this program to make sure the program can handle all the possible cases that you can think of. This means you'll need to work with **exceptions**.

Exercise 5: We have been using simple primitive data types for holding numerical data so far. Java also has "wrapper classes" for primitive data types, which are an object form of the primitive data. For example instead of the `double` type, we could use `Double`. Instead of `int`, we can use `Integer`.

Describe the importance of each of the components of this declaration and initialisation of a `Double` object.

```
1 Double weight = new Double(45.6);
```

`Double` _____

`weight` _____

`=` _____

`new` _____

`(45.6)` _____

Why might it be useful to use an object version of these numbers rather than the standard primitive type? Hint: think about the difference between the values after the following declarations: `Double d;` and `double d;`.

Point object

The `Point` class in Java represents a pair of coordinates `(x, y)`, which is useful when doing point-based arithmetic.

To use the `Point` class, you need to import the class:

```
1 import java.awt.Point;
```

To create a new point, you can call the **constructor** like this:

```
1 Point point = new Point(-2, 5); // represents the point (-2,5)
```

The `Point` API lists a few useful methods for working with points. Have a look at this API if you are unsure about using a `Point` object.

One method that you can call on a `Point` object is as follows:

`double distance(Point pt)`

- Returns the distance from this `Point` to a specified `Point`.

For example:

```
1 Point p1 = new Point(1, 2);
2 Point p2 = new Point(3, 4);
3 double distance = p1.distance(p2); // Gives the distance from p1 to p2
```

The reason this method is not listed on the `Point` API is because technically it belongs to the `Point2D` API due to *inheritance*. This is a topic that will be covered later in the semester.

Exercise 6: Write a program to read in four numbers as command line inputs in the order `x1`, `y1`, `x2`, `y2`, and then output the distance between the points `(x1, y1)` and `(x2, y2)`.

Exercise 7: What will the following code snippets output? Discuss the reason for these outputs with your classmates.

```
double weight = 60.0;
double oldWeight = weight;
weight = 57.0;
System.out.println("weight: " + weight + " oldweight:" + oldWeight);
```

```
Point point = new Point(1, 2);
Point oldPoint = point;
point.setLocation(3, 4);
System.out.println("point: " + point)
System.out.println("oldPoint:" + oldPoint);
```

```
Double weight = new Double(60.0);
Double oldWeight = weight;
weight = new Double(57.0);
System.out.println("weight: " + weight + " oldweight:" + oldWeight);
```

The `Date` class The `Date` class in Java is used to represent a date and time. In order to use the date class, you need to import it:

```
import java.util.Date;
```

When you create a new `Date` object, the date is automatically set to the time and date that it was created:

```
Date currentDate = new Date();
System.out.println(currentDate);
```

A standard in computer science is to represent the time as a single number. In general, this is known as Unix Time, and in Java we represent time as the number of milliseconds since 01/01/1970.

```
Date epochTime = new Date();
epochTime.setTime(0);
System.out.println(epochTime);
```

Exercise 8: Write a program that will take in a long integer, and print out what the date and time will be in that many milliseconds.

Extensions

Extension 1: Modify the program from Exercise 3 to make the find part case insensitive. For the more adventurous, try accepting a **regular expression** instead of a search word. This can be quite powerful!

Extension 2: Modify the program from Exercise 9 so that instead of a number of milliseconds, the program can take in multiple arguments to give units to the number. For example it might take in 40 hours 12 seconds, and will then tell you the time 40 hours and 12 seconds from now.

You may want to investigate the use of the `Calendar` class, which makes dealing with Java dates much easier.