Questions and Exercises

These questions and exercises is an opportunity to see what you've learnt from the lecture as well as practice the new things we've been talking about. In other words, these questions and exercises are completely optional but it's recomended to do them. In the end of the document you will find the answers to the questions as well as possible solutions to the exercises, note that one can solve an exercise in different ways. There will also be some suggestions about what one could code if one want to continue with some more advanced things. These suggestions will not come with a possible solution and might include things that haven't been covered in the lecture.

Question 1
What scopes are valid for an enum constructor?

Question 2
What is the finally block used for?

Question 3
How do you make a catch block that accepts multiple error types?

Question 4
What is an EnumSet?

Question 5
What does the *throws* in the following method do?

```
private void myMethod throws FileNotFoundException {

}
```

Exercise 1
Write a enum with string modifaction filters (UPPERCASE, REVERSE and DOUBLE) and a method accepting an EnumSet with these filters and a string to apply the filters to. The filtered string should be returned. For instance if the UPPERCASE and DOUBLE filters are used on the string

Hello

it should return

HELLOHELLO

Exercise 2
Write a progam that asks the user for a file path. Open the file and sum all the integers in the file. If an ivalid files was given(it doesn't exist or contain other things than integers) the user should get the option to enter another file path.

Further explorations 1
Time to do the assignment, check it out on the course page.

Further explorations 2
Use the Dragon Game base shown in the lecture (the source can be found on the lecture page) and add more things to it. Have fun.

Answers and solutions

Answer to Question 1
For a constructor of a enum you're not allowed to use public or protected scopes. Therefore the only valid ones are private and the default one. Not however that other methods can have public or protected scope.

Answer to Question 2
In a try statement the finally block will run its code in the end. This means that it will be run wether the try block successfully ran its code or if any batch block had to run its code.

The good thing about it compared to just adding your code after the try catch is that even if you return a value from the method you're in in for instance a catch block(which will simply prevent it from running any code after the try catch) it will still run the finally block just before it returns the value.

Answer to Question 3
You can pipe different error types together while allows the catch block to accept multiple types, it can be done with something like the following

```
try {

}catch (FileNotFoundException|IllegalStateException e) {

}
```

Answer to Question 4
An EnumSet is a sort of generic list specificly designed for enums. It can be used to store multiple

enum values of the same type at the same time. Methods like EnumSet.allOf(...), EnumSet.complementOf(...) and EnumSet.of(...) makes it very easy to use.

Answer to Question 5

Adding throws and an exception type after a method allows us to freely throw that exception inside the method. In other words we don't have to bother catching it with a try catch statement. However when the method is called we will have to be able to catch the error somehow.

Possible solution to Exercise 1

```
public enum StringFilters {
   UPPERCASE,
   REVERSE,
   DOUBLE;
}

import java.util.EnumSet;
public class Exercise1 {

   public static void main(String[] args) {
                              System.out.println(applyFilters(EnumSet.of(StringFilters.UPPERCASE,
StringFilters.DOUBLE), "Hello"));
      System.out.println(applyFilters(EnumSet.allOf(StringFilters.class), "Hello World"));
      System.out.println(applyFilters(EnumSet.noneOf(StringFilters.class), "This won't change"));
   }


   private static String applyFilters(EnumSet<StringFilters> filters, String raw) {
      if (filters.contains(StringFilters.UPPERCASE)) {
         raw = raw.toUpperCase();
      }

      if (filters.contains(StringFilters.REVERSE)) {
         String temp = raw;
         raw = "";
         for (int i = 0; i < temp.length(); i++) {
            raw = temp.charAt(i) + raw;
         }
      }

      if (filters.contains(StringFilters.DOUBLE)) {
         raw += raw;
      }

      return raw;
   }

}
```

Possible solution to Exercise 2

```
import java.util.Scanner;
import java.io.File;
import java.io.FileNotFoundException;
import java.util.InputMismatchException;

public class Exercise2 {
   public static void main(String[] args) {
      Scanner myScanner = new Scanner(System.in);

      while (true) {
         Scanner fileScanner = null;
         try {
            System.out.println("Please enter the path to read");
            String path = myScanner.nextLine();

            fileScanner = new Scanner(new File(path));

            int sum = 0;
            while (fileScanner.hasNext()) {
               sum += fileScanner.nextInt();
            }
            System.out.println("The sum is " + sum);

            break;
         }catch (FileNotFoundException|InputMismatchException e) {
            System.out.println("Something went wrong");
            continue;
         }finally {
            if (fileScanner != null) {
               fileScanner.close();
            }
         }
      }
   }
}
```