Tutorial 6 Exceptions

1. Having seen an example of how to create an arithmetic exception in the lecture slides i.e. System.out.println(number1/number2); when variable number2 is 0, create a new class named Test with a main method. Write code in the main method that will cause the Java default exception handler to output the following stack trace. (Your output doesn't have to have the exact line number and exception message 10).

```
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 10
at Test1.main(Test1.java:7)
```

```java
int [] a = new int[10];
a[10] = 100;
```

2. Write a try-block and catch-block to handle the exception. Once the exception is caught print the string of the exception and exception message on separate lines.

```java
try {
        int [] a = new int[10];
        a[10] = 100;
} catch (ArrayIndexOutOfBoundsException e) {
        System.out.println(e.toString());
        System.out.println(e.getMessage());
}
```

3. If we want to catch all exceptions some block of code will throw. What should we put in the () at the start of the catch blocks?

```java
try {
        .
        .//block of code goes here
        .
} catch (Exception e) {
        .
        .
        .
}
```

4. Why does this work?

Because Exception is the base class for all exceptions.

5. In most situations, we want our exception handler to be as specific as possible i.e. being able to identify the exact type of exception being thrown rather than identifying exceptions by the most generic Exception class. Why is that?

"The reason is that the first thing a handler must do is determine what type of exception occurred before it can decide on the best recovery strategy. In effect, by not catching specific errors, the handler must accommodate any possibility. Exception handlers that are too general can make code more error-prone by catching and handling exceptions that weren't anticipated by the programmer and for which the handler was not intended."

https://docs.oracle.com/javase/tutorial/essential/exceptions/advantages.html

6. We can also arbitrarily throw exceptions in our code. Inside the main method, write code to arbitrarily throw a NoSuchMethodException inside a try-block followed by a catch-block that catches the most generic exception.

```
try {
	throw new NoSuchMethodException();
} catch (Exception e) {
	System.out.println(e.toString());
}
```

7. It is also possible to write a customised exception. Write a customised exception named [your name]sSpecialException. In your customised exception, overwrite the getMessage() and getLocalizedMessage with your own messages. Also implement a method getOrigin() that returns a string "This exception was made in a Compsci 230 lab".

```
public class WilliamsSpecialException extends Exception {

	public String getLocalizedMessage() {
		return "This is the localised exception message";
	}

	public String getMessage() {
		return "This is the exception message";
	}

	public String getOrigin() {
		return "This exception was made in a Compsci 230 lab";
	}
}
```

8. Throw an instance of your customized exception inside a try-block. Catch it using a specific catch block and print out the result of the getMessage(), getLocalizedMessage() and getOrigin() methods.

```
try {
        WilliamsSpecialException w = new WilliamsSpecialException();
       throw w;
} catch (WilliamsSpecialException e) {
        System.out.println(e.getMessage());
        System.out.println(e.getLocalizedMessage());
        System.out.println(e.getOrigin());
}
```