EECS1720
Worksheet 1 – Exceptions

1) Which types of expressions may raise exceptions?

   (a) class instance creation expressions e.g.,
   ```
   new MyClass();
   ```
   (b) method invocation expressions e.g.,
   ```
   myObject.myMethod(arg);
   ```
   (c) expressions with integer division or integer remainder with a zero divisor e.g.,
   ```
   78 % 0
   ```
   (d) all of the above

2) Suppose an expression potentially raises an exception and you wish to add exception handling. What are your options?

   (a) use a `try-catch` block
   (b) use a `throws` expression in the method header
   (c) neither A nor B; just hope for the best
   (d) either A or B; either is an exception handler.

3) Suppose an exception is raised at run-time and my code is set up to handle the exception in a catch block.

```
catch (RuntimeException  theCaughtException) {
   // here is the code to handle the exception
   // the variable theCaughtException will hold
   // the reference to the exception object
}
```

   What things can I do with this object reference?
   a) there are no possible actions because it is a variable
   b) there are some actions, but it a mystery and there is no way to tell
   c) these are some possible actions, to understand look in the API
   d) there are infinite possible actions, your imagination is the limit

4) Given two example applications.. In one example, the compiler does not give a compile time error. For the other, the compiler does give a compile-time error. Why is this?

   (a) The compiler is unpredictable;
   (b) It might work differently another time and we should try again later
   (c) The compiler is enforcing a rule that is conditional; it depends on whether the expression is a method invocation or an instantiation
   (d) The compiler is enforcing a rule that is conditional; it depends on the type of the exception being thrown
   (e) None of the above

5) Consider the following code segment (you may need to refer to the Java API to answer)

```
output.println ("Enter a fraction (x/y) and I will give you the
quotient");
String str = input.nextLine();
int slash = str.indexOf("/") ;
String left = str.substring(O, slash);
String right = str.substring(slash + 1);
int numer = Integer.parseint(left);
int denom = Integer.parseint(right);
int quotient = numer/denom;
output.println("Quotient = " + quotient);
```

   How many different exceptions could potentially be raised?

   a) 0
   a) 1
   b) 2
   c) 3

6) A compiled program consists of a series of bytecode instructions. One of these instructions may be invalid, even though the compiler did not issue an error.

   (a) TRUE
   (b) FALSE

7) If an app that is invoked using in one particular running environment (e.g. on one computer) **does not crash**, then that same app, when invoked in some other running environment (i.e. another different computer), will also **not crash**.

   (a) TRUE
   (b) FALSE

8) Thrown exceptions are like run-time errors: they are bad and a sign that something went wrong.

   (a) TRUE
   (b) FALSE

9) What is a crash?

   a) When the compiler issues an error due to syntactic problems
   b) When the JVM terminates at run-time because an exception was never handled
   c) When the app is incorrect due to logical errors
   d) When the flow of control in an app reaches the end of the main method

10) Why doesn't the compiler check for run-time errors and prevent them from happening?

   a) The compiler *does* check for run-time errors. This is a trick question!
   b) The compiler does not check for run-time errors because this checking is not specified in the Java Language Specification.
   c) The compiler does not check for run-time errors because this is not possible, from both a practical and theoretical perspective.
   d) The compiler does not check for run-time errors because it is desired for programmers to develop stronger design skills.

11) What is the typical strategy for building exception handling into your code?

12) What is the difference between an Error and an Exception?

13) Explain how the JVM deals with exceptions at runtime