OOP and Data Structures — CS2110 Summer 2019

Discussion 3 — exceptions

Due 7/3 at 5PM

Before working on these problems, browse the following sections of the Java tutorial:

- What is an exception?
- Catching and handling exceptions
- The try block
- The catch blocks
- How to throw exceptions

As you work, feel free to search up other resources. The web is your oyster.

1. In the file `written.txt`, write the basic exception-class hierarchy, with `Throwable` at the top. Include at least two subclasses of each of `Error`, `Exception`, and `RuntimeException`.

   When writing a class hierarchy, we usually use indentation to denote subtyping. For example

   ```
   Animal
     Dog
       Collie
     Cat
   ```

   indicates that classes Dog and Cat are subclasses of class Animal and class Collie is a subclass of Dog.

2. Consider the following class:

   ```java
   public class A {
     public static double p(int x) {
       int y = x;
       try {
           System.out.println("six");
           y = 5/x;
           System.out.println("five");
           return 5/(x + 2);
       } catch (RuntimeException e) {
           System.out.println("four");
           y = 5/(x+1);
           System.out.println("three");
       }
       System.out.println("two");
       y = 4/x;
       System.out.println("one");
       return 1/x;
     }
   }
   ```

   Answer the following in `written.txt`:

   (a) Write the output of `p(0)`. Do it by hand first, then check your work in Eclipse.

   (b) Write the output of `p(-2)`. Do it by hand first, then check your work in Eclipse.

3. In the attached file `Discuss3.java` (on CMS), complete the method `min`, which should throw an exception but does not. See the method specification for details.

4. In `Discuss3.java`, update method `main`, so that any of the method calls throws an exception, the exception is caught, an error message is printed, and the execution of `main` continues.

   Do **not** add a `throws` clause to main; main should not throw any exceptions.

5. In `Discuss3.java`, update method `printProducet` so that it satisfies its specification. It currently does not replace invalid inputs with 1. While doing this, update `main` to call `printProduct`; you will need to understand and fix an error (don't change any method specifications while doing this).

6. Java method headers can have "throws" clauses:

   ```java
   public static void f () throws E1, E2 {
   }
   ```

   This declaration means that f will **not** throw any exception **other than** one that is-a E1, E2, RuntimeException, or Error[1].

   Java enforces these restrictions by requiring you to catch any other exceptions that might be thrown.

   Suppose an interface declares a method `f` that can throw exception `E`:

   ```java
   interface I {
     /**
      * Does the thing.
      * @throws E if things go wrong.
      */
     void f() throws E;
   }
   ```

   Suppose a class `C` wants to implement the interface. Which of the following are valid declarations of `f` (assume `E`, `E1` and `E2` are not `Errors` or `RuntimeExceptions`)?

   (a) `void f() {}` (with no `throws`)

   (b) `void f() throws E1 {}` where `E1` is a subtype of `E`

   (c) `void f() throws E2 {}` where `E2` is a supertype of `E`

   (d) `void f() throws E1, E2 {}` where `E1` is a subtype of `E` and `E2` is an unrelated exception type.

   To answer these questions, think about whether something that satisfies the specification in `C.f` will automatically satisfy the specification in `I.f`. If so, then `C.f`'s specification is more specific, and is thus a valid implementation of `I.f`.

   Another way to think about this question is to ask what you can do with an object of type `I`, and then ask whether (with the given specification) you can do that with an object of type `C`.

   After you have an idea, check your answers in Eclipse.

[1]every java method can implicitly throw any subtype of RuntimeException or Error. These are called "Unchecked exceptions"; all other `Throwables` are called "checked exceptions".