COMPSCI 105: Principles of Computer Science
Summer 2007

Tutorial Two: Exceptions

Helen Gu and Slobodan Vukanovic
{ygu029, svuk002}@ec.auckland.ac.nz

This tutorial is not being assessed. It provides you with an opportunity to become familiar with concepts introduced in lectures.

Question One

An absent--minded programmer wrote the following piece of code:

```
void addOneAndPrint () {
      Object[] numbers = {
         new Integer (1), new Integer (2), "3", new Integer (4),
         new Integer (5), "6", "7", new Integer (8), "9", "10"
      };

      for (int i = 0; i < numbers.length; i++) {
         int value = ((Integer)numbers[i]).intValue ();
         System.out.print (value + 1 + " ");
      }

      System.out.println ();
}
```

They wanted to iterate through each element of array `numbers`, add one to it, and print the resulting array on a single line. Their program compiles, but generates a runtime exception. Your task is to, without modifying the array `numbers` and the overall structure, make the code work. Write your answer in the box below.

```
void addOneAndPrint () {
      Object[] numbers = {
         new Integer (1), new Integer (2), "3", new Integer (4),
         new Integer (5), "6", "7", new Integer (8), "9", "10"
      };

      for (int i = 0; i < numbers.length; i++) {

          int value;

          try {
              value = ((Integer)numbers[i]).intValue();
          }
          catch (Exception e) { /* A ClassCastException really */
             value = Integer.parseInt ((String)numbers[i]);
          }
          System.out.print (value + 1 + " ");
      }

      System.out.println ();
}
```

Question Two

There is something wrong with the following code. Find the problem, and then solve it by rewriting the code.

```
String course = "CS105";
int number = 0;
try {
   number = Integer.parseInt (course.substring (1, 6));
}
catch (Exception e) {
   System.out.println ("An exception occurred.");
}
catch (NumberFormatException e) {
   System.out.println ("Wrong format for a number.");
}
catch (StringIndexOutOfBoundsException e) {
   System.out.println ("Invalid String Index");
}
finally {
   number = Integer.parseInt (course.substring (2, 5));
   System.out.println (number);
}
```

The problem is:

Since catches are checked sequentially, the program won't compile as both NumberFormatException and StringIndexOutOfBoundsException are covered by the more general Exception.

Solution is:

Reorder the catches:

```
String course = "CS105";
int number = 0;
try {
   number = Integer.parseInt (course.substring (1, 6));
}
catch (NumberFormatException e) {
   System.out.println ("Wrong format for a number.");
}
catch (StringIndexOutOfBoundsException e) {
   System.out.println ("Invalid String Index");
}
catch (Exception e) {
   System.out.println ("An exception occurred.");
}
finally {
   number = Integer.parseInt (course.substring (2, 5));
}
```

Question Three

What is the output of the following code?

```
int numberOfExceptions = 0;
int result;
int[][] values = { {0, 1, 1},
                   {1, 0, 0} };

for (int i = 0; i < values.length; i++) {
   for (int j = 0; j < values[0].length; j++) {
      try {
         result = values[i][j]/values[i+1][j+1];
      }
      catch (ArithmeticException e) {
         System.out.println ("Division by Zero.");
         numberOfExceptions++;
      }
      catch (ArrayIndexOutOfBoundsException e) {
         System.out.println ("Invalid index.");
         numberOfExceptions++;
         break;
      }
   }
}

System.out.println ("There are " + numberOfExceptions
                + " exceptions raised in this program");
```

Output is:

```
Division by Zero.
Division by Zero.
Invalid index.
Invalid index.
There are 4 exceptions raised in this program
```

Question Four

To convert a lower--case character to upper--case (caps), we could do the following:

```
upperCaseCharacter = lowerCaseCharacter - 'a' + 'A'
```

This approach will not work if `lowerCaseCharacter` is not a lower--case character. To deal with this situation, we can force a check:

```
if lowerCaseCharacter is really a lower case character:
  upperCaseCharacter = lowerCaseCharacter - 'a' + 'A'
else:
  do not convert
```

You are to download two Java files from the course webpage (look under ``Tutorials''). `CapsFormatException.java` defines a new type of exception -- you don't have to modify this file. File `ToCaps.java` is responsible for converting the lower--case characters of some strings to upper case. This is done in method `convertToCaps ()`: the method takes in a string, and returns the same string but with its lower--case characters converted to caps. All other characters in the original string are left intact. Your approach is required to cause a `CapsFormatException` if the currently--processed character is not lower--case. You should also deal with handling this exception.

An example input string is ``Hello, World!'', and its corresponding output is ``HELLO, WORLD!''. Some test cases are provided -- just run the `main ()` method of `ToCaps.java` once you complete `convertToCaps ()`.

You may find the pseudocode below useful:

```
let s be the string to be returned

for each character c in the input string:
  Attempt to convert c to upper case
  On a failed attempt (i.e. if c is not lower--case),
   raise CapsFormatException
  if a CapsFormatException occurs:
     notify the user that the exception has occurred, and do
      not modify c
  in any case, whether an exception occurs or not, append c to s

return s
```