CECS 174 – Lecture 28

Exception Handling

Exceptions – are java's way of telling you that something bad happened. Usually something so bad, the program crashes. Java gives us the ability to handle these exceptions if we plan for them ahead of time.

Examples – Here are a few examples of run time errors you may have seen:

1. InputMismatchException – occurs when the user inputs a value that is not of the expected data type. `int x = in.nextInt() //user inputs an 'a'`
2. ArithmeticException – occurs when you divide by 0.
3. ArrayIndexOutOfBoundsException – occurs when you attempt to access an element of an array that does not exist.
4. StringIndexOutOfBoundsException – occurs when you attempt to access an index of a string that does not exist.

Whenever one of these situations arises, an exception is thrown. If it is caught by your code, then your program can continue on. If it isn't caught, then your program will crash.

Try/Catch Block – the try section allows us to attempt code that may pose a risk of crashing the program. If an exception isn't thrown, it just continues on with the rest of the code. If an exception is thrown, then we need to catch it with a catch block by identifying the type of exception it will expect and then executing the solution.

Example – InputMismatchException

```
import java.util.*;//needed for scanner and input exception
public class InputTest {
     public static void main(String [] args) {
          boolean testInput = true;
          while ( testInput ) {
               Scanner in = new Scanner(System.in);
               System.out.println("Enter a Number");
               try{
                    int x = in.nextInt(); //exception?
                    testInput = false;
               }catch ( InputMismatchException im ) {
                    String junk = in.next(); //clear buffer
                    System.out.println("Invalid Input");
               }
          }
     }
}
```

Multiple catch blocks may be used after a try block if you expect more than one possible exception to be thrown.

Example – InputMismatchException

```
import java.util.*;//needed for scanner and input exception
import java.lang.*;//needed for array exception
public class InputTest {
     public static void main(String [] args) {
          Scanner in = new Scanner(System.in);
          int [] array = {0, 1, 2, 3, 4};
          boolean testInput = true;
          while ( testInput ) {
               System.out.print("Enter a Number: ");
               try{
                    int x = in.nextInt(); //exceptions?
                    System.out.println(array[x]);
                    testInput = false;
               }catch ( InputMismatchException im ) {
                    String junk = in.next(); //clear buffer
                    System.out.println("Invalid Input");
               }catch (ArrayIndexOutOfBoundsException ob){
                    System.out.println("Invalid Location");
               }
          }
     }
}
```

```
/* Output
Enter a Number: a
Invalid Input
Enter a Number: 7
Invalid Location
Enter a Number: 3
3
*/
```

You can use as many catch blocks as you feel is necessary to handle your unwanted exceptions. Try/Catch blocks may even be nested within other Try/Catch blocks.

Note: If you do not know what type of exception to put in your catch, you can write a test program to purposely cause the error to occur, the exact name of the error and the library it's located in will be displayed when the error happens.

Throws – you can write a method that will throw an exception so that the method that calls your method is then forced to use a try/catch on the possible error.

```
     public void methodEx() throws ExampleException {
          if(error){
               throw new ExampleException();
```