Exception Basics

1. ***Basic Concept of Exceptions:*** - Exception handling allows developers to detect errors easily **`without`** writing special code to **`test return values`**. Traditional programming requires us to check return calls on every method we call and take some action when an error occurs. In a large system, this can generate some difficult to follow and hard to test programs. To see how the Exception error handling approach works, we need some definitons:
2. ***Exception*** - A class that is created at the point an error occurs. The Exception **`class contains detailed error`** information about what happens.
3. ***throwing an Exception*** - When an Exception occurs, the program that discovers the problem executes a statement that looks like:
   ```
   Exception  e= new Exception("The xyz widget received an error 123 when accessing the abc method");
   throw e;
   ```
4. ***Exception Processing*** - The code which is responsible for doing something about the error is called an exception handler, and it catches the exception. Exception handling works by trnasferring the execution of a program to an exception handler when an exception occurs.
   ```
   try
   {
     Your main line code
    }
    catch (Exception e)
    {
       Error Handling code
    }
   ```
5. Java exception handling improves **`code organization`** by separating a method's error handling from the body of the method. Exception handling allows similar errors to be handled by an single handler which can be a previously called method. About the only downside is that there is **`small performance hit`** when exception handling is used. However, your code is much easier to debug and **`it's much easier to track errors`**.
6. Create some simple examples:
   - Cause an exception(how about div/0) and see what happens if you don't catch it.
   - Cause the exception again and now catch it.
   - Cause an exception in another method and catch it at the main level and also try out catching at the method level.
   - Demo calling the printStackTrace method in the Throwable class.
   - Demo constructing an exception in another method and throwing it

   The above examples were worked in the Video and the following is the resultant code.

   ```
   import java.util.InputMismatchException;
   import java.util.Scanner;


   public class Test {

       Scanner keyboard = new Scanner(System.in);

       int divideByZero(int num, int div)
       {
           try
           {
               return num/div;
           }
           catch (ArithmeticException e)
           {
               System.out.println("We caught our divide by zero error "+ e);
               return -1;
           }
       }

       int divideByZero2(int num, int div)
       {
               return num/div;
       }

       int readInteger()
   ```

```
        {
            boolean goodInput = false;
            int retValue=-1;

            while(!goodInput)
            {
                try
                {
                    retValue = keyboard.nextInt();
                    goodInput = true;
                }
                catch (InputMismatchException e)
                {
                    System.out.println("Your integer was bad, try again");
                    keyboard.next();
                }
            }
            return retValue;
        }



        public static void main(String[] args) {
            Test t = new Test();


            try
            {
                System.out.println("Enter in divisor ");
                int divisor = t.readInteger();

                int x = t.divideByZero(10, 1);
                x = t.divideByZero(20, divisor);
                x = t.divideByZero2(10, 1);
                x = t.divideByZero2(20, divisor);  // We expect to get an exception here


            }
            catch (ArithmeticException e)
            {
                System.out.println("Caught divide by zero in main: " + e);
            }

            System.out.println("Normal exit from my routine");


        }

    }
```

7. More Specific Exception Processing

```
import java.util.*;

class ReadInts
{
    Scanner scan = new Scanner(System.in);

    int[] readAtYourRisk(int num)
    {
        System.out.println("readAtYourRisk ");
        int[] retVals = new int[num];
        for (int i=0; i < num; i++)
        {
            // An InputMismatchException is thrown if you enter
            //    in a non-integer
            retVals[i] = scan.nextInt();
        }
        return retVals;
    }

    int[] readCarefully(int num)
```

```
        {
            System.out.println("readCarefully ");
            int[] retVals = new int[num];
            for (int i=0; i < num; i++)
            {
                boolean unread = true;
                while(unread)
                {
                    try
                    {
                        // An InputMismatchException is thrown if you enter
                        //    in a non-integer
                        retVals[i] = scan.nextInt();
                        unread = false;
                    }
                    catch (InputMismatchException e)
                    {
                        System.out.println("That Last number was bad ... try again");
                        scan.next(); // Throw away bad int
                    }
                }
            }
            return retVals;
        }

        void dump(int[] vals)
        {
            System.out.println("dump ****************************");
            for (int i=0; i < vals.length; i++)
                System.out.println(vals[i]+ " ");
            System.out.println();
        }

        public static void main(String[] args)
        {
            ReadInts rd = new ReadInts();
            int[] vals = rd.readAtYourRisk(5);
            rd.dump(vals);

            vals = rd.readCarefully(5);
            rd.dump(vals);

            boolean done = false;
            System.out.println("readAtYourRisk with main try ... catch protection");
            while (!done)
            {
                try
                {
                    vals = rd.readAtYourRisk(5);
                    rd.dump(vals);
                    done = true;
                }
                catch (InputMismatchException e)
                {
                    System.out.println("Your last attempt failed ... Start over");
                    rd.scan.next(); // Throw away bad int
                }

            }

        }
    }
```

Last Updated: July 27, 2014 11:52 PM