Introduction to java

COMP1030 Lecture #8

Housekeeping

- Our goal - 100% pass rate in this course.
- Review the lecture slides ahead of time.
- Review the lecture slides after class with a study group.
- Repeat the lab at home 1-2 times.
- Take notes for each lecture.

Housekeeping

- Assignment #1 Answer key is posted under the assignments link
- Midterm-Exam question paper and answer key are posted under course information link
- Labs will no longer be checked as homework

Review

- Array definition and representation
- Declaring arrays
- Using arrays

Instantiating Multidimensional Arrays

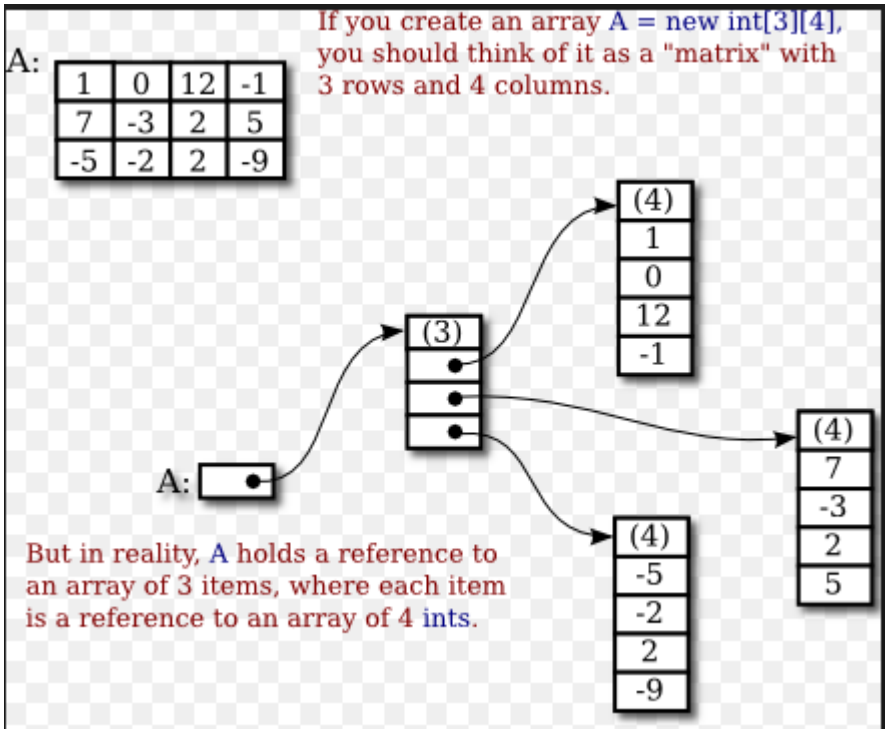
int[][] myArray = new int[3][4];

```
int[][]  A  =  {  {  1,  0, 12, -1 },
                  {  7, -3,  2,  5 },
                  { -5, -2,  2, -9 }
               };
```

```
A  =  new int[][] {  {  1,  0, 12, -1 },
                     {  7, -3,  2,  5 },
                     { -5, -2,  2, -9 }
                  };
```

From: http://math.hws.edu/javanotes/c7/s5.html

Multidimensional Arrays



From: http://math.hws.edu/javanotes/c7/s5.html

Multidimensional Arrays

```
1  import java.util.Random;
2  public class Test
3  {
4      public static void main (String [] args)
5      {
6          int[][] myArray = new int[3][4];
7          Random randomGenerator = new Random();
8          for (int i = 0;i < 13; i++)
9          {
10             int randomValue = randomGenerator.nextInt(1000);
11             myArray[0][0] = randomValue;
12             randomValue = randomGenerator.nextInt(1000);
13             myArray[0][1] = randomValue;
14             randomValue = randomGenerator.nextInt(1000);
15             myArray[0][2] = randomValue;
16             randomValue = randomGenerator.nextInt(1000);
17             myArray[0][3] = randomValue;
18             randomValue = randomGenerator.nextInt(1000);
19             myArray[1][0] = randomValue;
20             randomValue = randomGenerator.nextInt(1000);
21             myArray[1][1] = randomValue;
22             randomValue = randomGenerator.nextInt(1000);
23             myArray[1][2] = randomValue;
24             randomValue = randomGenerator.nextInt(1000);
25             myArray[1][3] = randomValue;
26             randomValue = randomGenerator.nextInt(1000);
27             myArray[2][0] = randomValue;
28             randomValue = randomGenerator.nextInt(1000);
29             myArray[2][1] = randomValue;
30             randomValue = randomGenerator.nextInt(1000);
31             myArray[2][2] = randomValue;
32             randomValue = randomGenerator.nextInt(1000);
33             myArray[2][3] = randomValue;
34         }
35
```

Multidimensional Arrays

```
for (int i = 0;i < 13; i++)
{
    System.out.println(myArray[0][0]);
    System.out.println(myArray[0][1]);
    System.out.println(myArray[0][2]);
    System.out.println(myArray[0][3]);
    System.out.println(myArray[1][0]);
    System.out.println(myArray[1][1]);
    System.out.println(myArray[1][2]);
    System.out.println(myArray[1][3]);
    System.out.println(myArray[2][0]);
    System.out.println(myArray[2][1]);
    System.out.println(myArray[2][2]);
    System.out.println(myArray[2][3]);
}
```

```
$javac Test.java
$java -Xmx128M -Xms16M Test
38
296
14
455
49
123
858
783
102
354
769
651
```

Exception Handling

- The term *exception* is shorthand for the phrase "exceptional event."
- An *exception* is an event, which occurs during the execution of a program, that disrupts the normal flow of the program's instructions.
- When an error occurs within a method, the method creates an object and hands it off to the runtime system. The object, called an *exception object*, contains information about the error, including its type and the state of the program when the error occurred. Creating an exception object and handing it to the runtime system is called *throwing an exception*.

source: Oracle Corporation

Exception Handling

- To avoid having the normal flow of the program disrupted, the programmer can place their code within a “try”, “catch” block.
- This will prevent the program from being interrupted, and instead will allow the program to recover from the error because the exception was “handled” by the programmer/code.

Exception Handling

- A stack trace, for example, is seen when the exception is not handled, and shows the name of the exception, the problem that occurred and the contents of the method-call-stack.

Exception Handling


```
1  // Fig. 11.1: DivideByZeroNoExceptionHandling.java
2  // Integer division without exception handling.
3  import java.util.Scanner;
4
5  public class DivideByZeroNoExceptionHandling
6  {
7     // demonstrates throwing an exception when a divide-by-zero occurs
8     public static int quotient( int numerator, int denominator )
9     {
10       return numerator / denominator; // possible division by zero
11    } // end method quotient
12
13    public static void main( String[] args )
14    {
15       Scanner scanner = new Scanner( System.in ); // scanner for input
16
17       System.out.print( "Please enter an integer numerator: " );
18       int numerator = scanner.nextInt();
19       System.out.print( "Please enter an integer denominator: " );
20       int denominator = scanner.nextInt();
21
22       int result = quotient( numerator, denominator );
23       System.out.printf(
24          "\nResult: %d / %d = %d\n", numerator, denominator, result );
25    } // end main
26 } // end class DivideByZeroNoExceptionHandling
```

```
Please enter an integer numerator: 100
Please enter an integer denominator: 7

Result: 100 / 7 = 14
```

Stack Trace

```
Please enter an integer numerator: 100
Please enter an integer denominator: 0
Exception in thread "main" java.lang.ArithmeticException: / by zero
        at DivideByZeroNoExceptionHandling.quotient(
           DivideByZeroNoExceptionHandling.java:10)
        at DivideByZeroNoExceptionHandling.main(
           DivideByZeroNoExceptionHandling.java:22)
```

Name of method where exception occurred.

Line number where exception occurred.

Exception Handling

- Once a line of code in a try block throws an exception, the program jumps immediately to the appropriate catch block ignoring the remaining lines of code in the try block.

Exception Handling

- The finally block is an optional block that must be placed after the last catch block.
- If there are no catch blocks, the finally block immediately follows the try block.
- The finally block executes regardless if an exception is thrown in the corresponding try block.

Exception Handling

```
try {
    //Statements that may cause an exception
}
catch {
   //Handling exception
}
finally {
   //Statements to be executed
}
```

From https://beginnersbook.com/2013/04/java-finally-block/

Exception Handling

```
class Example
{
   public static void main(String args[]) {
      try{
         int num=121/0;
         System.out.println(num);
      }
      catch(ArithmeticException e){
         System.out.println("Number should not be divided by zero");
      }
      /* Finally block will always execute
       * even if there is no exception in try block
       */
      finally{
         System.out.println("This is finally block");
      }
      System.out.println("Out of try-catch-finally");
   }
}
```

Output:

```
Number should not be divided by zero
This is finally block
Out of try-catch-finally
```

From https://beginnersbook.com/2013/04/java-finally-block/

Exception Handling

```java
import java.util.Random;
public class Test
{
    public static void main (String [] args)
    {
        int d = 0;
        int n = 20;
        try
        {
            int fraction = n / d;
            System.out.println("This line will not be Executed");
        }
        catch (ArithmeticException e)
        {
            System.out.println("In catch Block #1 due to Exception");
        }
        catch (NullPointerException e)
        {
            System.out.println("In catch Block #2 due to Exception");
        }
        finally
        {
            System.out.println("In the finally block");
        }
            System.out.println("End Of Main");
    }
}
```

Result

```
$javac Test.java
$java -Xmx128M -Xms16M Test
In catch Block #1 due to Exception
In the finally block
End Of Main
```