Lab 05: The Pirate Solution

CO225: Software Construction

July 12, 2016

1 Introduction

This lab requires you to write a Java Class similar to *.java.util.ArrayList*. Your Implementation should allow the users to create an *Array* with Integers.

2 Problem

In Java, "normal" arrays are fixed-size. You have to give them a size and can't expand them or contract them.

```
int[] arrayOfInts = new int[10];
```

If you want to change the size of the array, you have to make a new array and copy the data you want. In this lab you are going to implement a custom implementation of an Integer *Array* where the size of array can grow/shrink dynamically.

You will have to create appropriate methods to do the following operations.

```
Array intList = new Array(); // Initialize an empty
Array intList = new Array(n); // Initialize an array with given number of elements 'n'.
    default element values is '0'

intList.add(int d); //add the integer 'd' to the end of the array
intList.add(int index, int d) // add the integer 'd' to the given index
intList.replace(int index, int d); //replace the element in index with 'd'

intList.get(int index); //get the element in the given index
intList.remove(int index); //remove the element in the given index

intList.isEmpty(); //return true/false based on wether or not the array is empty
intList.size(); //return the size of the array

intList.contains(int d); // return true/false based on wether or not the element is found on
    the array

intList.trimToSize(int size); //trim the array so that only first 'size' elemnts are
    available in the array
intList.clear(); //remove all the element from the array

intList.toString(); //return a string with the elements in the array eg. [3, 4, 5, 1, 2, 5]
```

In your implementation you are expected to make use of the Java *Exception* in appropriate places.

3 Exception

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.

After a method throws an exception, the runtime system attempts to find something to handle it. Using exceptions to manage errors has some advantages over traditional error-management techniques.

Example

```
public static void main(String args[]){
      int num[]={1,2,3,4};
      System.out.println(num[5]);
}
```

Even though the above method has a valid *Java* syntax, It will produce an error at the runtime. If you compile and execute the above program you will get exception as shown below.

```
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 5
   at Exceptions.Main.main(Main.java:8)
```

Catching Exceptions

A method catches an exception using a combination of the try and catch keywords. A try/catch block is placed around the code that might generate an exception. Code within a try/catch block is referred to as protected code, and the syntax for using try/catch looks like the following:

```
try
{
   //Protected code
}catch(ExceptionName e1)
{
   //Catch block
}
```

The code which is prone to exceptions is placed in the try block, when an exception occurs, that exception occurred is handled by catch block associated with it. Every try block should be immediately followed either by a catch block or finally block.

A catch statement involves declaring the type of exception you are trying to catch. If an exception occurs in protected code, the catch block (or blocks) that follows the try is checked. If the type of exception that occurred is listed in a catch block, the exception is passed to the catch block much as an argument is passed into a method parameter.

Example

The following is an array is declared with 2 elements. Then the code tries to access the 3rd element of the array which throws an exception.

```
// File Name : ExcepTest.java
import java.io.*;
public class ExcepTest{

   public static void main(String args[]){
      try{
         int a[] = new int[2];
         System.out.println("Access element three :" + a[3]);
      }catch(ArrayIndexOutOfBoundsException e){
         System.out.println("Exception thrown :" + e);
      }
      System.out.println("Out of the block");
   }
}
```

This would produce the following result:

```
Exception thrown :java.lang.ArrayIndexOutOfBoundsException: 3
Out of the block
```

You can learn more in the Advantages of Exceptions Section here `http://www.tutorialspoint.com/java/java_exceptions.htm`.

4 What to turn in?

Array.Java File with the implementation.
ArrayTest.java File with unit testing. You need to cover all the corner cases.