Homework 10

**Name: Peter Dobbs**

**MU ID: 005803169**

**Date: 9 April, 2015**

**Homework Problem 1**

**Requirements**

Create two arrays and fill one with random integers. Copy the values from the one array into the other and display them side by side.

**Design**

*copyArray*

for loop that sets the contents of array 2 equal to the contents of array 1 and the appropriate index

*main*

Create arrays and fill the first array with random integers. Call copyArray and display the values of the two arrays side by side.

**Iterative Development Steps**

1. Create copyArray method that has two arrays as parameters and sets the values of the two arrays equal to each other.
2. Create a main method that creates the arrays and assigns random values to one of them and passes them both to the copyArray method.

**Tests**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Description of test | Input | Expected result | Actual result | Cause |
| method1 normal input | n/a | Equal values side by side | Equal values side by side | Correct code |

**Homework Problem 2**

**Requirements**

Create a program that mirrors the functions of ArrayList.

**Design**

*class ExpandableArray*

public ExpandableArray()

public void add(int index, int input)

adds input at index

public int get(int index)

public void set(int index, int input)

inserts input at index

public int size()

returns the current size of the array

*class Test\_ExpandableArray*

main

tests the ExpandableArray

**Iterative Development Steps**

1. Create an expandable array according to specifications similar to ArrayList
2. Create a test class that tests the expandable array

**Tests**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Description of test | Input | Expected result | Actual result | Cause |
| method1 normal input | n/a | Size: 10  Contents:  1  2  3  4  5  6  7  8  9  10 | Size: 10  Contents:  1  2  3  4  6  7  8  9  10  0 | Some sort of problem |

**Homework Problem 3**

**Requirements**

Create program that assigns numbers to an array that may have entries that override other entries and solve the problem therein.

**Design**

*class IntHasher*

public IntHasher()

Constructor initializes the fields

public boolean enter(int x)

enters the numbers into the arrays

public int getIndex(int x)

returns the index of the parameter number

public long getProbeCount()

returns the probe count

public int getEntryCount()

returns the entry count

public void clear()

clears the fields of all assignments

*class Test\_IntHasher*

main

tests the IntHasher program

**Iterative Development Steps**

1. Create the IntHash class according to specifications
2. Test the IntHash according to specifications

**Tests**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Description of test | Input | Expected result | Actual result | Cause |
| method1 normal input | n/a | Increasing average probe count with increasing number of entries | Increasing average probe count with increasing number of entries | Good code |

**References**

An Introduction to Programming Using Java, Anthony J Dos Reis.