ITITWE23014
Lê Thành Danh
Week 1 - DSA's Lab - OOP Reviews & Arrays

Problem 1.1:

```java
import java.lang.Math;

public class ArrayToNum {
    public static void main(String[] args) throws Exception {
        int[] digits = { 2, 0, 1, 8, 0, 0, 5 };
        int result = 0;
        int pow = digits.length - 1;
        for (int i = 0; i < digits.length; i++) {
            if (digits[i] != 0) {
                result += digits[i] * Math.pow(10, pow);
                System.out.println(result);
            }
            pow--;

        }
    }
}
```

Turn an array to a number, by iterating through the array and modify the result variable, a bit more detail, we take the first element of the array to multiply with 10^pow, start with digits.length, length of the array.
2*10^6=2000000
+
0*10^5=0
+
1*10^4=10000
….
result=2018005

```
PS C:\Users\Inugami\Documents\GitHub\lab-dsa\ITITWE23014_LTDANH_Lab1\src>  & 'C:\Program Files\Java\jre-1.8\bin\java.
4bc3c0f995eab4cad65d409cdeb\redhat.java\jdt_ws\src_fa1af3fe\bin' 'ArrayToNum'
 ;90a0251f-4084-49ca-9b64-ed5edfa78f5d2000000
2010000
2018000
2018005
```

Problem 1.2:

```java
import java.util.ArrayList;
import java.util.List;
```

```java
import java.util.Scanner;

public class ListMedian {
    public static void main(String[] args) throws Exception {
        List<Integer> BaseList = new ArrayList<>();
        Scanner sc = new Scanner(System.in);
        int iteration = 1;
        int result = 0;
        while (true) {
            try {
                System.out.print("input of " + (iteration) + "(Press other
chars to stop):");
                int InputNum = sc.nextInt();
                result += InputNum;
                BaseList.add(InputNum);
            } catch (Exception e) {
                break;
            }
        }
        int length = BaseList.toArray().length;
        double median = result / (length);
        System.out.println(median);
    }
}
```

Make a List instead of Array for dynamic adding value, without declaring the size of it. Simple while loop for input, using try-catch awaiting for input error (not a number, falsely input) to break and stop the input step.
Just add all up while in iteration and divide it with the length of the List (which need to convert the List to an array to get) to output the median.

```
4bc3c0f995eab4cad65d409cdeb\redhat.java\jdt_ws\src_fa1af3fe\bin' 'ListMedian'
 ;a432562e-d132-43ab-9194-3dbfd2f34835input of 1(Press other chars to stop):2
input of 1(Press other chars to stop):3
input of 1(Press other chars to stop):4
input of 1(Press other chars to stop):5
input of 1(Press other chars to stop):1
input of 1(Press other chars to stop):d
3.0
```

Problem 1.3

```java
import java.util.ArrayList;
```

```java
import java.util.List;
import java.util.Scanner;

public class MinGap {
    public static void main(String[] args) throws Exception {
        int[] intArray = new int[] { 1, 3, 6, 7, 12 };
        int result=0;
        for (int i = 0; i < intArray.length - 1; i++) {
            result= Math.abs(intArray[i]-intArray[i+1]);
            System.out.println(result);
        }
    }
}
```

Get the minGap like the question ask by subtracting n and n+1 element in the iteration and return the absolute value of it.

```
PS C:\Users\Inugami\Documents\GitHub\lab-dsa\ITITWE23014_LTDANH_Lab1\src>  & 'C:\Program Files\Ja
4bc3c0f995eab4cad65d409cdeb\redhat.java\jdt_ws\src_fa1af3fe\bin' 'MinGap'
135322-e8bf-4c9e-aa0b-69cff1fc18312
3
1
5
```

Problem 1.4:

```java
//********************************************************************
//  GasMileage.java       Author: Lewis/Loftus
//
//  Demonstrates the use of the Scanner class to read numeric data.
//********************************************************************

import java.util.Scanner;

public class GasMileage
{
   //-----------------------------------------------------------------
   //  Calculates fuel efficiency based on values entered by the
   //  user.
   //-----------------------------------------------------------------
   public static void main (String[] args)
   {
```

```
      double gallons, mpg;
      Scanner scan = new Scanner(System.in);
      for (int i = 0; i < 2; i++) {
        System.out.print("Enter the car name, miles, and gallons
(separated by blanks): ");

        String carName = scan.next();

        //double miles = scan.nextDouble();
        //1st way, we make sure that the input work with integer as float
        int miles = (int) Math.ceil(scan.nextDouble());
        //2nd way, we can round that number and convert it back to an
integer

        gallons = scan.nextDouble();

        mpg = miles / gallons;
        System.out.println(carName + " - Miles Per Gallon: " + mpg);
     }
   }
}
```

Like the comment I made, the first way is declaring miles as double to work with both integer and float number, second way is declaring it as an int type, round it down and make that round number an integer.
A for loop is use for multiple iteration of the input and output.

```
PS C:\Users\Inugami\Documents\GitHub\lab-dsa\ITITWE23014_LTDANH_Lab1\src>  & 'C:\Program Files\Ja
4bc3c0f995eab4cad65d409cdeb\redhat.java\jdt_ws\src_fa1af3fe\bin' 'GasMileage'
Enter the car name, miles, and gallons (separated by blanks): Tesla 20.3 23
Tesla - Miles Per Gallon: 0.9130434782608695
Enter the car name, miles, and gallons (separated by blanks): Mustang 20.5 26
Mustang - Miles Per Gallon: 0.8076923076923077
```

Problem 1.5:

```
public class Student
{
    public String fname, type, lname;
    public int grade;

    public Student(String fname, String lname, int grade)
```

```
    {
        this.fname = fname;
        this.lname = lname;
        this.grade = grade;
        if  (grade > 89) {
            this.type = "excellent";
        } else if  (grade >= 60) {
            this.type = "ok";
        } else this.type = "failure";

    }

    public String toString()
    {
        return fname + " " + lname + "\t" + grade + "\t" + type;
    }
}
```

Make the variable public, like the question asked. And add the if condition for student type.

```
/*
 * Reading student records from a file, generating Student objects,
counting and averaging
 * Suggested exercises:
 * - Use grade to determine the type of the student: excellent (> 89), ok
[60,89], and failure (< 60)
 * - Define an enum type {excellent, ok, failure} and use it to print the
student type
 * - Do counting and averaging within each student type (excellent, ok,
and failure)
 * - Count students by using a static variable in class Student
 */
import java.util.Scanner;
import java.io.*;

public class Students
{
    public static void main (String[] args) throws IOException
```

```
    {   String first_name, last_name;
        int grade, total=0, count=0;
        double average;
        Scanner fileInput = new Scanner(new File("students.txt"));
        for (;fileInput.hasNext();)
        {
            first_name = fileInput.next();
            last_name = fileInput.next();
            grade = fileInput.nextInt();


            Student st = new Student(first_name, last_name, grade);

            System.out.println(st);
            total = total + grade;
            count++;
        }
        average = (double)total/count;
        System.out.println("There are " + count + " students with average
grade " + average);
    }
}
```

Use a For loop instead of While, result the same.
I didn't do much here, just modifying the for loop.

```
PS C:\Users\Inugami\Documents\GitHub\lab-dsa\ITITWE23014_LTDANH_Lab1\src\Problem_1_05>  & 'C:\Progr
Storage\676f34bc3c0f995eab4cad65d409cdeb\redhat.java\jdt_ws\src_fa1af3fe\bin' 'Students'
John Smith      90      excellent
Barack Obama    95      excellent
Al Clark        80      ok
Sue Taylor      55      failure
Ann Miller      75      ok
George Bush     58      failure
John Miller     65      ok
There are 7 students with average grade 74.0
```

Problem 2
The question asked me to run ArrayApp

```
PS C:\Users\Inugami\Documents\GitHub\lab-dsa\ITITWE23014_LTDANH_Lab1\src>  & 'C:\Program Files\Java\jre-1.8\bin\
4bc3c0f995eab4cad65d409cdeb\redhat.java\jdt_ws\src_fa1af3fe\bin' 'ArrayApp'
7b02ad32-9ed2-4c56-a018-16f20474d3a8Display items:
77 99 44 55 22 88 11 0 66 33
Find item with key: 66
Found 66
Delete item with key: 55
Found item to be deleted: 55
Display items after deleting: 55
77 99 44 22 88 11 0 66 33
```

The question asked me to run ClassDataApp

```
PS C:\Users\Inugami\Documents\GitHub\lab-dsa\ITITWE23014_LTDANH_Lab1\src>  & 'C:\Program
4bc3c0f995eab4cad65d409cdeb\redhat.java\jdt_ws\src_fa1af3fe\bin' 'ClassDataApp'
   Last name: Evans, First name: Patty, Age: 24        vans
   Last name: Smith, First name: Lorraine, Age: 37
   Last name: Yee, First name: Tom, Age: 43
   Last name: Adams, First name: Henry, Age: 63
   Last name: Hashimoto, First name: Sato, Age: 21
   Last name: Stimson, First name: Henry, Age: 29
   Last name: Velasquez, First name: Jose, Age: 72
   Last name: Lamarque, First name: Henry, Age: 54
   Last name: Vang, First name: Minh, Age: 22
   Last name: Creswell, First name: Lucinda, Age: 18
Found    Last name: Stimson, First name: Henry, Age: 29
Deleting Smith, Yee, and Creswell
   Last name: Evans, First name: Patty, Age: 24
   Last name: Adams, First name: Henry, Age: 63
   Last name: Hashimoto, First name: Sato, Age: 21
   Last name: Stimson, First name: Henry, Age: 29
   Last name: Velasquez, First name: Jose, Age: 72
   Last name: Lamarque, First name: Henry, Age: 54
   Last name: Vang, First name: Minh, Age: 22
```

Both classes remove duplication in it own Array.

```
// lowArray.java
// demonstrates array class with low-level interface
// to run this program: C>java LowArrayApp
////////////////////////////////////////////////////////////////
class LowArray
   {
   private long[] a;                 // ref to array a
//--------------------------------------------------------------
   public LowArray(int size)         // constructor
      { a = new long[size]; }        // create array
//--------------------------------------------------------------
   public void setElem(int index, long value)  // set value
      { a[index] = value; }
//--------------------------------------------------------------
```

```
   public long getElem(int index)              // get value
      { return a[index]; }
//--------------------------------------------------------------
   public void removeMax(int nElems) {
      if (nElems == 0) {
         System.out.println("Array is empty, no max value to remove.");
         return;
      }

      int maxIndex = 0;
      for (int i = 1; i < nElems; i++) {
         if (a[i] > a[maxIndex]) {
            maxIndex = i;
         }
      }

      for (int i = maxIndex; i < nElems - 1; i++) {
         a[i] = a[i + 1];
      }

      a[nElems - 1] = 0;
   }

}  // end class LowArray
////////////////////////////////////////////////////////////////
class LowArrayApp
   {
   public static void main(String[] args)
      {
      LowArray arr;                 // reference
      arr = new LowArray(100);      // create LowArray object
      int nElems = 0;               // number of items in array
      int j;                        // loop variable

      arr.setElem(0, 77);           // insert 10 items
      arr.setElem(1, 99);
      arr.setElem(2, 44);
      arr.setElem(3, 55);
      arr.setElem(4, 22);
      arr.setElem(5, 88);
```

```
      arr.setElem(6, 11);
      arr.setElem(7, 00);
      arr.setElem(8, 66);
      arr.setElem(9, 33);
      nElems = 10;                // now 10 items in array

      for(j=0; j<nElems; j++)     // display items
         System.out.print(arr.getElem(j) + " ");
      System.out.println("");

      int searchKey = 26;         // search for data item
      for(j=0; j<nElems; j++)           // for each element,
         if(arr.getElem(j) == searchKey) // found item?
            break;
      if(j == nElems)                   // no
         System.out.println("Can't find " + searchKey);
      else                              // yes
         System.out.println("Found " + searchKey);

                                  // delete value 55
      for(j=0; j<nElems; j++)           // look for it
      if(arr.getElem(j) == 55)
         break;
      for(int k=j; k<nElems; k++)       // higher ones down
         arr.setElem(k, arr.getElem(k+1) );
         arr.removeMax(nElems);
//ii. Programming Projects 2.2 in Text-Book (lowArray.java)
//Not low, but the book says higharray, i dont really know if i doing this
right...
      nElems--;                         // decrement size

      for(j=0; j<nElems; j++)     // display items
         System.out.print( arr.getElem(j) + " ");
      System.out.println("");
      }  // end main()
   }  // end class LowArrayApp
////////////////////////////////////////////////////////////////
```

So the question in the book says modify the highArray, but the question ask to modify the lowArray instead, So i just add the removeMax, as the question ask me to do.

```
PS C:\Users\Inugami\Documents\GitHub\lab-dsa\ITITWE23014_LTDANH_Lab1\src>  c:; cd 'c:\Users
va.exe' '-cp' 'C:\Users\Inugami\AppData\Roaming\Code\User\workspaceStorage\676f34bc3c0f995e
77 99 44 55 22 88 11 0 66 33 ab4cad65d409cdeb\x5credhat.java\x5cjdt_ws\x5csrc_fa1af3fe\x5cb
Can't find 26
77 44 22 88 11 0 66 33 0
```

```java
// highArray.java
// demonstrates array class with high-level interface
// to run this program: C>java HighArrayApp
////////////////////////////////////////////////////////////////

import java.util.Random;

class HighArray {
   private long[] a; // ref to array a
   private int nElems; // number of data items
   // ------------------------------------------------------------

   public HighArray(int max) // constructor
   {
      a = new long[max]; // create the array
      nElems = 0; // no items yet
   }

   // ------------------------------------------------------------
   public int find(long searchKey) {
      int comparisons = 0;
      for (int j = 0; j < nElems; j++) {
          comparisons++;
          if (a[j] == searchKey) {
              return comparisons;
          }
      }
      return comparisons;
  }// end find()
   // ------------------------------------------------------------

   public void insert(long value) // put element into array
   {
```

```
        a[nElems] = value; // insert it
        nElems++; // increment size
    }

    // ------------------------------------------------------------
    public boolean delete(long value) {
        int j;
        for (j = 0; j < nElems; j++) // look for it
            if (value == a[j])
                break;
        if (j == nElems) // can't find it
            return false;
        else // found it
        {
            for (int k = j; k < nElems; k++) // move higher ones down
                a[k] = a[k + 1];
            nElems--; // decrement size
            return true;
        }
    } // end delete()
    // ------------------------------------------------------------

    public void display() // displays array contents
    {
        for (int j = 0; j < nElems; j++) // for each element,
            System.out.print(a[j] + " "); // display it
        System.out.println("");
    }

    // ------------------------------------------------------------
    public long getMax() {
        if (nElems == 0) {
            return -1;
        }
        long max = a[0];
        for (int i = 1; i < nElems; i++) {
            if (a[i] > max) {
                max = a[i];
            }
        }
```

```
      return max;
   }

   public void noDups() {
      for (int i = 0; i < nElems; i++) {
         for (int j = i + 1; j < nElems; j++) {
            if (a[i] == a[j]) {
               delete(a[j]);
               j--;
            }
         }
      }
   }
   public void randomInsertions(int numInsertions) {
      Random rand = new Random();
      for (int i = 0; i < numInsertions; i++) {
          long randomValue = rand.nextInt(1000);
          insert(randomValue);
      }
  }
  public double computeAverageComparisons(int trials, int arraySize) {
   randomInsertions(arraySize);
   Random rand = new Random();
   int totalComparisons = 0;

   for (int i = 0; i < trials; i++) {
       long randomKey = a[rand.nextInt(nElems)];
       totalComparisons += find(randomKey);
   }

   return (double) totalComparisons / trials;
}

} // end class HighArray
////////////////////////////////////////////////////////////////

class HighArrayApp {
   public static void main(String[] args) {
      int maxSize = 1000; // Array size
      HighArray arr = new HighArray(maxSize);
```

```
      arr.randomInsertions(100);
      arr.display();

      System.out.println("Max value: " + arr.getMax());

      arr.noDups();
      arr.display();

      double avgComparisons = arr.computeAverageComparisons(100, 100);
      System.out.println("Average comparisons: " + avgComparisons);
   } // end main()
} // end class HighArrayApp
```

```
PS C:\Users\Inugami\Documents\GitHub\lab-dsa\ITITWE23014_LTDANH_Lab1\src>  c:; cd 'c:\Users\Inugami\Documents\GitHub\lab-dsa\ITITWE23014_LTDANH_La
b1\src'; & 'C:\Program Files\Java\jre-1.8\bin\java.exe' '-cp' 'C:\Users\Inugami\AppData\Roaming\Code\User\workspaceStorage\676f34bc3c0f995eab4cad6
5d409cdeb\redhat.java\jdt_ws\src_fa1af3fe\bin' 'HighArrayApp'
704 211 815 113 607 819 591 246 407 2 419 841 882 840 226 889 639 180 117 457 988 321 186 145 844 554 158 414 523 90 440 955 558 874 984 569 759 2
40 422 539 566 256 726 679 340 626 873 641 265 471 984 722 197 953 866 360 811 271 116 707 282 170 248 996 997 438 670 169 378 488 534 953 424 977
 384 121 177 356 459 809 711 274 802 502 115 991 229 732 672 344 195 767 798 804 780 478 443 940 749 325
Max value: 997
704 211 815 113 607 819 591 246 407 2 419 841 882 840 226 889 639 180 117 457 988 321 186 145 844 554 158 414 523 90 440 955 558 874 569 759 240 4
22 539 566 256 726 679 340 626 873 641 265 471 984 722 197 866 360 811 271 116 707 282 170 248 996 997 438 670 169 378 488 534 953 424 977 384 121
 177 356 459 809 711 274 802 502 115 991 229 732 672 344 195 767 798 804 780 478 443 940 749 325
Average comparisons: 87.56
```

```
// orderedArray.java
// demonstrates ordered array class
// to run this program: C>java OrderedApp
////////////////////////////////////////////////////////////////

import java.util.Random;

class OrdArray
   {
   long[] a;                 // ref to array a
   private int nElems;               // number of data items
   private int comparisons;
   //--------------------------------------------------------------
   public OrdArray(int max)          // constructor
      {
      a = new long[max];             // create array
      nElems = 0;
      comparisons = 0;
```

```
    }
//-------------------------------------------------------------
public int size()
    { return nElems; }
//-------------------------------------------------------------
public int find(long searchKey) {
    int lowerBound = 0;
    int upperBound = nElems - 1;
    int curIn;
    comparisons = 0;

    while (true) {
        comparisons++;
        curIn = (lowerBound + upperBound) / 2;
        if (a[curIn] == searchKey)
            return curIn;
        else if (lowerBound > upperBound)
            return nElems;
        else {
            if (a[curIn] < searchKey)
                lowerBound = curIn + 1;
            else
                upperBound = curIn - 1;
        }
    }
}
//-------------------------------------------------------------
public void insert(long value)    // put element into array
    {
    int j;
    for(j=0; j<nElems; j++)       // find where it goes
        if(a[j] > value)          // (linear search)
            break;
    for(int k=nElems; k>j; k--)   // move bigger ones up
        a[k] = a[k-1];
    a[j] = value;                 // insert it
    nElems++;                     // increment size
    }  // end insert()
//-------------------------------------------------------------
public boolean delete(long value)
```

```
      {
      int j = find(value);
      if(j==nElems)                  // can't find it
         return false;
      else                           // found it
         {
         for(int k=j; k<nElems; k++) // move bigger ones down
            a[k] = a[k+1];
         nElems--;                   // decrement size
         return true;
         }
      }  // end delete()
   //-----------------------------------------------------------
   public void display()             // displays array contents
      {
      for(int j=0; j<nElems; j++)       // for each element,
         System.out.print(a[j] + " ");  // display it
      System.out.println("");
      }
   //-----------------------------------------------------------
   public int getComparisons() {
      return comparisons;
   }

   public void resetComparisons() {
      comparisons = 0;
   }
}  // end class OrdArray
////////////////////////////////////////////////////////////////
class OrderedApp
   {
   public static void main(String[] args)
      {
      int maxSize = 1000;
      Random rand = new Random();
      OrdArray arr;

      for (int size = 100; size <= 1000; size += 100) {
         arr = new OrdArray(size);
```

```
         for (int i = 0; i < size; i++) {
            arr.insert(rand.nextInt(10000));
         }

         long totalComparisons = 0;
         for (int trial = 0; trial < 100; trial++) {
            long randomKey = arr.find(arr.a[rand.nextInt(size)]);
            totalComparisons += arr.getComparisons();
         }

         double averageComparisons = totalComparisons / 100.0;
         System.out.println("Array size: " + size + " - Average
comparisons: " + averageComparisons);
      }
      }  // end main()
   }  // end class OrderedApp
```

```
PS C:\Users\Inugami\Documents\GitHub\lab-dsa\ITITWE23014_LTDA
b1\src'; & 'C:\Program Files\Java\jre-1.8\bin\java.exe' '-cp'
5d409cdeb\redhat.java\jdt_ws\src_fa1af3fe\bin' 'OrderedApp'
ceStorage\x5c676f34bc3c0f995eab4cad65d409cdeb\x5credhat.java\
rray size: 100 - Average comparisons: 5.6
Array size: 200 - Average comparisons: 6.78
Array size: 300 - Average comparisons: 7.29
Array size: 400 - Average comparisons: 7.59
Array size: 500 - Average comparisons: 7.97
Array size: 600 - Average comparisons: 8.36
Array size: 700 - Average comparisons: 8.74
Array size: 800 - Average comparisons: 8.7
Array size: 900 - Average comparisons: 8.76
Array size: 1000 - Average comparisons: 9.04
```

The modified highArray and orderedApp is pointing out that on the large scale, Binary search take less comparison than linear search, which linear search make every comparison possible, while binary search divide and divide, making less comparison but increasing in trials on larger scale.