CSCI 145 PA __9__ Submission

Due Date:___May 2, 2023___ Late (date and time):_____

Name(s):_____Ivan Leung_____ & _____

Exercise 1 -- need to submit source code and I/O
-- check if completely done __x__ ; otherwise, discuss issues below

Pseudocode below if applicable:

Source code below:

```
package pa9;

/*  Java Class: CSCI 145
Modified by: Ivan Leung
Class: Mon/Wed
Date: Apr 26 2023
Description:

I certify that the code below is my own work.

Exception(s): N/A

*/


//*******************************************************************
*****
//Coin.java Author: Lewis/Loftus
//
//Represents a coin with two sides that can be flipped.
//*******************************************************************
*****

public class Coin {
      private final int HEADS = 0;
      private final int TAILS = 1;
      private int face;
      private double bias;
```

```
     // ------------------------------------------------------------
--------
     // Sets up the coin with no bias and flipping it initially.
     // ------------------------------------------------------------
--------
     public Coin()
     {
          bias = 0.5;
          flip();
     }

     // ------------------------------------------------------------
--------
     // Sets up the coin with bias and flipping it initially.
     // ------------------------------------------------------------
--------
     public Coin(double bias) {
          this.bias = (bias >= 0 && bias <= 1 ? bias : 0.5);
          flip();
     }

     // ------------------------------------------------------------
--------
     // Flips the coin by randomly choosing a face value.
     // ------------------------------------------------------------
--------
     public void flip() {
          face = (Math.random() <= bias ? HEADS : TAILS);
     }

     // ------------------------------------------------------------
--------
     // Returns true if the current face of the coin is heads.
     // ------------------------------------------------------------
--------
     public boolean isHeads() {
          return (face == HEADS);
     }

     // ------------------------------------------------------------
--------
     // Returns the current face of the coin as a string.
     // ------------------------------------------------------------
--------
     public String toString()
```

```
	{
		String faceName;
		if (face == HEADS)
			faceName = "Heads";
		else
			faceName = "Tails";
		return faceName;
	}

}
```

```
package pa9;

/*  Java Class: CSCI 145
Author: Ivan Leung
Class: Mon/Wed
Date: Apr 26 2023
Description:

I certify that the code below is my own work.

Exception(s): N/A

*/


public class TestCoin {

	public static void main(String[] args) {
		Coin fairCoin = new Coin();
		final double biasedRate1 = 0.8;
		final double biasedRate2 = 0.2;
		Coin biasedCoin1 = new Coin(biasedRate1);
		Coin biasedCoin2 = new Coin(biasedRate2);
		int totalHeadsFair = 0;
		int totalHeadsBiased1 = 0;
		int totalHeadsBiased2 = 0;

		for (int i = 0; i < 100; ++i) {
			fairCoin.flip();
			biasedCoin1.flip();
			biasedCoin2.flip();

			if (fairCoin.isHeads()) {
				++totalHeadsFair;
			}
```

```
                if (biasedCoin1.isHeads()) {
                    ++totalHeadsBiased1;
                }
                if (biasedCoin2.isHeads()) {
                    ++totalHeadsBiased2;
                }
            }

            System.out.println("Total heads...");
            System.out.println("Fair coin: " + totalHeadsFair);
            System.out.println("Coin with " + (int) (biasedRate1 *
100) + "% bias: " + totalHeadsBiased1);
            System.out.println("Coin with " + (int) (biasedRate2 *
100) + "% bias: " + totalHeadsBiased2);
     }

}
```

Input/output below:

```
Total heads...
Fair coin: 49
Coin with 80% bias: 76
Coin with 20% bias: 20
```

Exercise 2 -- need to submit source code and I/O
-- check if completely done __x__ ; otherwise, discuss issues below

Pseudocode below if applicable:

Source code below:

```
package pa9;

/*  Java Class: CSCI 145
Author: Ivan Leung
Class: Mon/Wed
Date: Apr 26 2023
Description:

I certify that the code below is my own work.
```

```
Exception(s): N/A

*/

import java.util.Scanner;

public class ReverseArray {

	public static void main(String[] args) {
		int[] array;
		int count;
		Scanner scan = new Scanner(System.in);

		System.out.print("How many values? ");
		count = scan.nextInt();

		array = new int[count];

		System.out.print("Enter " + count + " values: ");
		for (int i = 0; i < count; ++i) {
			array[i] = scan.nextInt();
		}

		scan.close();

		System.out.println("Before reverse: ");
		for (int element : array) {
			System.out.print(" " + element);
		}
		System.out.println();

		for (int i = 0; i < count / 2; ++i) {
			int tmp = array[i];
			array[i] = array[count - 1 - i];
			array[count - 1 - i] = tmp;
		}

		System.out.println("After reverse: ");
		for (int element : array) {
			System.out.print(" " + element);
		}
		System.out.println();
	}

}
```

Input/output below:

```
How many values? 6
Enter 6 values: 89 35 75 12 62 12
Before reverse:
 89 35 75 12 62 12
After reverse:
 12 62 12 75 35 89
```

Exercise 3 -- need to submit source code and I/O
-- check if completely done __x__ ; otherwise, discuss issues below

Pseudocode below if applicable:

Source code below:

```
package pa9;

/*  Java Class: CSCI 145
Modified by: Ivan Leung
Class: Mon/Wed
Date: Apr 26 2023
Description:

I certify that the code below is my own work.

Exception(s): N/A

*/

//*************************************************************
//Square.java
//
//Define a Square class with methods to create and read in
//info for a square matrix and to compute the sum of a row,
//a col, either diagonal, and whether it is magic.
//
//*************************************************************

import java.util.Scanner;
```

```java
public class Square {
	int[][] square;

	// ------------------------------------
	// create new square of given size
	// ------------------------------------
	public Square(int size) {
		square = new int[size][size];
	}

	// ------------------------------------
	// return the sum of the values in the given row
	// ------------------------------------
	public int sumRow(int row) {
		int sum = 0;
		for (int col = 0; col < square.length; ++col) {
			sum += square[row][col];
		}
		return sum;
	}

	// ------------------------------------
	// return the sum of the values in the given column
	// ------------------------------------
	public int sumCol(int col) {
		int sum = 0;
		for (int row = 0; row < square.length; ++row) {
			sum += square[row][col];
		}
		return sum;
	}

	// ------------------------------------
	// return the sum of the values in the main diagonal
	// ------------------------------------
	public int sumMainDiag() {
		int sum = 0;
		for (int i = 0; i < square.length; ++i) {
			sum += square[i][i];
		}
		return sum;
	}

	// ------------------------------------
```

```
	// return the sum of the values in the other ("reverse")
diagonal
	// ------------------------------------
	public int sumOtherDiag() {
		int sum = 0;
		for (int row = 0, col = square.length - 1; row <
square.length; ++row, --col) {
			sum += square[row][col];
		}
		return sum;
	}

	// ------------------------------------
	// return true if the square is magic (all rows, cols, and
diags have
	// same sum), false otherwise
	// ------------------------------------
	public boolean magic() {
		int equal = sumMainDiag();
		if (sumOtherDiag() != equal) {
			return false;
		}
		for (int i = 0; i < square.length; ++i) {
			if (sumRow(i) != equal || sumCol(i) != equal) {
				return false;
			}
		}
		return true;
	}

	// ------------------------------------
	// read info into the square from the input stream
associated with the
	// Scanner parameter
	// ------------------------------------
	public void readSquare(Scanner scan) {
		for (int row = 0; row < square.length; row++)
			for (int col = 0; col < square.length; col++)
				square[row][col] = scan.nextInt();
	}

	// ------------------------------------
	// print the contents of the square, neatly formatted
	// ------------------------------------
	public void printSquare() {
		for (int row = 0; row < square.length; ++row) {
```

```
                for (int col = 0; col < square.length; ++col) {
                     System.out.print(" " + square[row][col]);
                }
                System.out.println();
           }
      }

}
```

```
package pa9;

/*  Java Class: CSCI 145
Modified by: Ivan Leung
Class: Mon/Wed
Date: Apr 26 2023
Description:

I certify that the code below is my own work.

Exception(s): N/A

*/

//*************************************************************
//SquareTest.java
//
//Uses the Square class to read in square data and tell if
//each square is magic.
//
//*************************************************************

import java.util.Scanner;
import java.io.File;
import java.io.IOException;

public class SquareTest {
     public static void main(String[] args) throws IOException {
          Scanner scan = new Scanner(new
File("C:\\Users\\ivanl\\OneDrive\\Desktop\\MagicData.txt"));
          int count = 1;
          // count which square we're on
          int size = scan.nextInt(); // size of next square
          // Expecting -1 at bottom of input file
          while (size != -1) {
               // create a new Square of the given size
               Square square = new Square(size);
```

```
                // call its read method to read the values of the
square
                square.readSquare(scan);
                System.out.println("\n******** Square " + count +
" ********\n");
                // print the square
                square.printSquare();
                // print the sums of its rows
                System.out.println("\nThe sum of each row...");
                for (int i = 0; i < size; ++i) {
                     System.out.println("Row " + (i + 1) + ": "
+ square.sumRow(i));
                }
                // print the sums of its columns
                System.out.println("\nThe sum of each
column...");
                for (int i = 0; i < size; ++i) {
                     System.out.println("Column " + (i + 1) + ":
" + square.sumCol(i));
                }
                // print the sum of the main diagonal
                System.out.println("\nThe sum of the main
diagonal: " + square.sumMainDiag());
                // print the sum of the other diagonal
                System.out.println("The sum of the other
diagonal: " + square.sumOtherDiag());
                // determine and print whether it is a magic
square
                System.out.println("The square is " +
(square.magic() ? "" : "not") + " a magic square.");
                // get size of next square
                ++count;
                size = scan.nextInt();
           }
     }

}
```

Input/output below:

```
******** Square 1 ********

 8 1 6
```

```
 3 5 7
 4 9 2

The sum of each row...
Row 1: 15
Row 2: 15
Row 3: 15

The sum of each column...
Column 1: 15
Column 2: 15
Column 3: 15

The sum of the main diagonal: 15
The sum of the other diagonal: 15
The square is  a magic square.

******** Square 2 ********

 30 39 48 1 10 19 28
 38 47 7 9 18 27 29
 46 6 8 17 26 35 37
 5 14 16 25 34 36 45
 13 15 24 33 42 44 4
 21 23 32 41 43 3 12
 22 31 40 49 2 11 20

The sum of each row...
Row 1: 175
Row 2: 175
Row 3: 175
Row 4: 175
Row 5: 175
Row 6: 175
Row 7: 175

The sum of each column...
Column 1: 175
Column 2: 175
Column 3: 175
Column 4: 175
Column 5: 175
Column 6: 175
Column 7: 175

The sum of the main diagonal: 175
```

```
The sum of the other diagonal: 175
The square is  a magic square.

******** Square 3 ********

 48 9 6 39
 27 18 21 36
 15 30 33 24
 12 45 42 3

The sum of each row...
Row 1: 102
Row 2: 102
Row 3: 102
Row 4: 102

The sum of each column...
Column 1: 102
Column 2: 102
Column 3: 102
Column 4: 102

The sum of the main diagonal: 102
The sum of the other diagonal: 102
The square is  a magic square.

******** Square 4 ********

 6 2 7
 1 5 3
 2 9 4

The sum of each row...
Row 1: 15
Row 2: 9
Row 3: 15

The sum of each column...
Column 1: 9
Column 2: 16
Column 3: 14

The sum of the main diagonal: 15
The sum of the other diagonal: 14
The square is not a magic square.
```

```
******** Square 5 ********

 3 16 2 13
 6 9 7 12
 10 5 11 8
 15 4 14 1

The sum of each row...
Row 1: 34
Row 2: 34
Row 3: 34
Row 4: 34

The sum of each column...
Column 1: 34
Column 2: 34
Column 3: 34
Column 4: 34

The sum of the main diagonal: 24
The sum of the other diagonal: 40
The square is not a magic square.

******** Square 6 ********

 17 24 15 8 1
 23 5 16 14 7
 4 6 22 13 20
 10 12 3 21 19
 11 18 9 2 25

The sum of each row...
Row 1: 65
Row 2: 65
Row 3: 65
Row 4: 65
Row 5: 65

The sum of each column...
Column 1: 65
Column 2: 65
Column 3: 65
Column 4: 58
Column 5: 72

The sum of the main diagonal: 90
```

```
The sum of the other diagonal: 60
The square is not a magic square.

******** Square 7 ********

 30 39 48 1 10 28 19
 38 47 7 9 18 29 27
 46 6 8 17 26 37 35
 5 14 16 25 34 45 36
 13 15 24 33 42 4 44
 21 23 32 41 43 12 3
 22 31 40 49 2 20 11

The sum of each row...
Row 1: 175
Row 2: 175
Row 3: 175
Row 4: 175
Row 5: 175
Row 6: 175
Row 7: 175

The sum of each column...
Column 1: 175
Column 2: 175
Column 3: 175
Column 4: 175
Column 5: 175
Column 6: 175
Column 7: 175

The sum of the main diagonal: 175
The sum of the other diagonal: 168
The square is not a magic square.
```

Add more exercises as needed

Exercise 4 -- need to submit source code and I/O
-- check if completely done __x__ ; otherwise, discuss issues below

Pseudocode below if applicable:

Source code below:

Please see extra credit.

Input/output below:

Answer for Question 1

Interface contains only abstract methods while class can contain all type of methods. Interface can only contain static variables, and constant variables while class can contain all type of variables. Interface cannot have constructors while class can have multiple constructors.

Answer for Question 2

Since array has fixed size, ArrayList is great choice when dynamic size is needed. Also, inserting elements into ArrayList can be easily done.

Extra Credit – provide if applicable

Pseudocode below if applicable:

Source code below:

```
package pa9;

/*  Java Class: CSCI 145
Author: Ivan Leung
Class: Mon/Wed
Date: Apr 26 2023
Description:

I certify that the code below is my own work.

Exception(s): N/A

*/
```

```java
import java.util.Scanner;

public class VoteCount {

	public static void main(String[] args) {
		final int SENTINEL_VALUE = 0;
		final int TOTAL_CANDIDATES = 10;
		final int MIN_ID = 1;
		final int MAX_ID = 10;
		int[] candidates = new int[TOTAL_CANDIDATES];
		int vote;
		int validVotes;
		int maxVote;
		int winner;
		int totalTie;
		int[] tieCandidateID;
		boolean[] isTie = new boolean[TOTAL_CANDIDATES];
		Scanner scan = new Scanner(System.in);

		validVotes = 0;
		System.out.print("Enter votes: ");
		do {
			vote = scan.nextInt();
			if (vote >= MIN_ID && vote <= MAX_ID) {
				++candidates[vote - 1];
				++validVotes;
			}
		} while (vote != SENTINEL_VALUE);
		scan.close();

		winner = 0;
		maxVote = candidates[0];
		System.out.println("\nNumber of valid votes: " +
validVotes);
		System.out.println("Results (candidate id and number
of votes):");
		for (int id = 0; id < TOTAL_CANDIDATES; ++id) {
			if (candidates[id] > 0) {
				System.out.println((id + 1) + "\t" +
candidates[id]);
				if (candidates[id] > maxVote) {
					maxVote = candidates[id];
					winner = id;
				}
			}
		}
```

```
            totalTie = 0;
            for (int id = 0; id < TOTAL_CANDIDATES; ++id) {
                  if (candidates[id] == maxVote) {
                        isTie[id] = true;
                        ++totalTie;
                  }
            }

            if (totalTie > 1) {
                  totalTie = (int) (Math.random() * totalTie + 1);
                  for (int id = 0; id < TOTAL_CANDIDATES; ++id) {
                        if (isTie[id] && totalTie > 0) {
                              winner = id;
                              --totalTie;
                        }
                  }
            }

            System.out.println("\nWinner: candidate " + (winner +
1));
      }

}
```

Input/output below:

```
Enter votes: 10 3 4 8 3 5 8 8 11 5 5 3 0

Number of valid votes: 11
Results (candidate id and number of votes):
3     3
4     1
5     3
8     3
10    1

Winner: candidate 5
```