CSCI 145 Homework 2

Due Monday, 4/24/2023

Name: Ivan Leung

Chapter 5

Ex 5.2

```
if (total == MAX) {
        if (total < sum)
                System.out.println("total == MAX and < sum.");
}
else
        System.out.println("total is not equal to MAX");
```

Ex 5.4

Apple

orange

pear

Ex 5.9

The while loop is infinite because count continues to increase without an upper limit. One way to remove the flaw is set an upper limit for the count. For example, “while (count >= 0 && count <= 100)”.

Another way to remove the flaw is decrement count instead of increment in the statement such as “count = count – 1;”. The third way to remove the flaw is by setting count to 0 and the condition is set to “while (count <= 50)”.

Ex 5.17

```
boolean isIsosceles(int a, int b, int c) {
        if ((a == b && a ! =c) || (a == c && a != b) || (b == c && b != a))
                return true;
        return false;
```

}

Ex 5.18

a) Radio buttons. You only have one favorite book genre.

b) Radio buttons. It can only be either visible or not visible, but not both.

c) Radio buttons. An image file can only have one format.

d) Check boxes. People may know more than one programming language.

PP 5.1

Source code below:

```
package hw2;

import java.util.Scanner;

/*  Java Class: CSCI 145
Author: Ivan Leung
Class: Mon/Wed
Date: Mar 22 2023
Description: Determine if a user input is a leap year.

I certify that the code below is my own work.

Exception(s): N/A

*/

public class LeapYear {

	public static void main(String[] args) {
		final int FIRST_YEAR = 1582;
		int year;
		boolean isLeapYear;
		Scanner scan = new Scanner(System.in);

		do {
			System.out.print("Enter a year: ");
			year = scan.nextInt();
			if (year < FIRST_YEAR)
				System.out.println("It must be equal to or
greater than year " + FIRST_YEAR + "!\nTry again!");
```

```
            } while (year < FIRST_YEAR);

            scan.close();

            if (year % 4 == 0) {
                  if (year % 100 == 0) {
                        if (year % 400 == 0)
                              isLeapYear = true;
                        else
                              isLeapYear = false;
                  }
                  else
                        isLeapYear = true;
            }
            else
                  isLeapYear = false;

            System.out.println(year + " is a " + ((isLeapYear) ? "" :
"not ") + "leap year.");
      }

}
```

Input/output below:

```
Enter a year: 2003
2003 is a not leap year.


Enter a year: 1900
1900 is a not leap year.


Enter a year: 2004
2004 is a leap year.


Enter a year: 2000
2000 is a leap year.


Enter a year: 1500
It must be equal to or greater than year 1582!
Try again!
Enter a year: 1582
1582 is a not leap year.
```

Chapter 6

Ex 6.1

a) 20

b) 20

c) 15

d) 20

e) 10

f) 5

Ex 6.7

```
for (int i = 1; i < 100; i += 2) {
        System.out.print(i + " ");
}
```

Ex 6.10

```
int count = 0;
for (int i = 0; i < name.length(); ++i) {
        if (name.charAt(i) == 'a')
                ++count;
}
System.out.println("a appears " + count + ((count == 1) " time." : " times."));
```

Ex 6.16

```
int sumRange(int a, int b) {
        int sum = 0;
        if (a < b)
                return sum;
        for (int i = a; i <= b; ++i) {
                sum += i;
```

```
        }
        return sum;
}
```

Ex 6.18

```
String reverse(String str) {
        String reverse = "";
        for (int i = str.length() - 1; i >= 0; --i) {
                reverse += str.charAt(i);
        }
        return reverse;
}
```

PP 6.7a

Source code below:

```
package hw2;

/*  Java Class: CSCI 145
Modified by: Ivan Leung
Class: Mon/Wed
Date: Mar 22 2023
Description: Determine if a user input is a leap year.

I certify that the code below is my own work.

Exception(s): N/A

*/


//********************************************************************
//Stars.java Author: Lewis/Loftus
//
//Demonstrates the use of nested for loops.
//********************************************************************

public class Stars {
```

```
	// -------------------------------------------------------------
---
	// Prints a triangle shape using asterisk (star) characters.
	// -------------------------------------------------------------
---
	public static void main(String[] args)
	{
		final int MAX_ROWS = 10;
		for (int row = MAX_ROWS; row >= 1; --row)
		{
			for (int star = row; star >= 1; --star)
				System.out.print("*");
			System.out.println();
		}
	}
}
```

Input/output below:

```
**********
*********
********
*******
******
*****
****
***
**
*
```

PP 6.7c

Source code below:

```
package hw2;

/*  Java Class: CSCI 145
Modified by: Ivan Leung
Class: Mon/Wed
Date: Mar 22 2023
Description: Determine if a user input is a leap year.

I certify that the code below is my own work.

Exception(s): N/A

*/
```

```
//********************************************************************
//Stars.java Author: Lewis/Loftus
//
//Demonstrates the use of nested for loops.
//********************************************************************

public class Stars {
	// -----------------------------------------------------------------
	// Prints a triangle shape using asterisk (star) characters.
	// -----------------------------------------------------------------
	public static void main(String[] args)
	{
		final int MAX_ROWS = 10;
		int currentRow = 0;
		for (int row = MAX_ROWS; row >= 1; --row)
		{
			for (int space = 0; space < currentRow ; ++space)
				System.out.print(" ");
			for (int star = row; star >= 1; --star)
				System.out.print("*");
			++currentRow;
			System.out.println();
		}
	}
}
```

Input/output below:

```
**********
 *********
  ********
   *******
    ******
     *****
      ****
       ***
        **
         *
```

Ex 7.4

```
String multiConcat(String str, int count) {
        String concatStr = “”;
        for (int i = 0; i < count; ++i) {
                concatStr += str;
        }
        return concatStr;
}
```

Ex 7.5

```
String multiConcat(String str) {
        return str + str;
}
```

Ex 7.10

Yes, it is consistent between primitive types and objects. In Java, parameters are always passed by value regardless of their types.

Ex 7.11

Static method cannot reference instance variables because instance variables do not exist until an object exists

Ex 7.15

```
public interface Breakable {
        public void break();
        public boolean broken();
}
```

PP 7.4

Source code below:

```
package hw2;

/*  Java Class: CSCI 145
Modified by: Ivan Leung
Class: Mon/Wed
Date: Apr 03 2023
Description: Testing the compareTo method

I certify that the code below is my own work.

Exception(s): N/A

 */

public class TestRationalNumber {

	public static void main(String[] args) {
		// TODO Auto-generated method stub
		RationalNumber r1 = new RationalNumber(5, 10);
		RationalNumber r2 = new RationalNumber(5, 9);
		RationalNumber r3 = new RationalNumber(5, 9);

		System.out.println("r1's value is " + r1);
		System.out.println("r2's value is " + r2);
		System.out.println("r3's value is " + r3);
		System.out.println();

		if (r1.compareTo(r2) == 0)
			System.out.println("r1 and r2 are equal.");
		else if (r1.compareTo(r2) == 1)
			System.out.println("r1 is greater than r2.");
		else
			System.out.println("r1 is less than r2.");

		if (r2.compareTo(r3) == 0)
			System.out.println("r2 and r3 are equal.");
		else if (r2.compareTo(r3) == 1)
			System.out.println("r2 is greater than r3.");
		else
			System.out.println("r2 is less than r3.");

		if (r3.compareTo(r1) == 0)
			System.out.println("r3 and r1 are equal.");
		else if (r3.compareTo(r1) == 1)
```

```
                System.out.println("r3 is greater than r1.");
            else
                System.out.println("r3 is less than r1.");

    }

}
```

Input/output below:

```
r1's value is 1/2
r2's value is 5/9
r3's value is 5/9

r1 is less than r2.
r2 and r3 are equal.
r3 is greater than r1.
```

Chapter 8

Ex 8.1

int primes = {2, 3, 4, 5, 7, 11};

It is invalid; the brackets are required to declare an array.

float elapsedTimes[] = {11.47, 12.04, 11.72, 13.88};

It is valid; the brackets can placed either behind the data type or the variable name.

int[] scores = int[30];

It is invalid; the left hand side must either have an initializer list or using the new keyword “new int [30];”

int[] primes = new {2,3,5,7,11};

It is invalid; it is a syntax error when using new keyword in the initializer list.

int[] scores = new int[30];

It is valid; it meets Java syntax requirements.

char grades[] = {'a', 'b', 'c', 'd', 'f'};

It is valid; it meets Java syntax requirements.

char[] grades = new char[];

It is invalid; the size of array must be specified when the array is instantiated.

Ex 8.5

a)

```
String[] studentNames = new String[25];
```

b)

```
String testGrades = new String[40];
```

c)

```
Transaction[] transactions = new Transaction[size];
public class Transaction {
	private int transactionNumber;
	private String merchantName;
	private double charge;
}
```

d)

```
Student[] students = Student[size];
public class Students {
	private String studentName;
	private int totalHomework;
	private double[] homeworkGrades = new double[totalHomework];
}
```

e)

```
Empolyee[] employees = Empolyee[size];
public class Empolyee {
	private int employeeNumber;
	private String hireDate;
	private double[] raises = new double[5];
}
```

Ex 8.8

```
for (int i  = 0; i < flags.length; ++i) {
```

```
        if ((i & 1) == 0)
                flags[i] = true;
        else
                flags[i] = false;
}
```

Ex 8.10

```
void switchThem(int array1, int array2) {
        int[] tmp;
        if (array1.length != array2.length) {
                System.out.println("The two arrays have different size!");
                return;
        }
        Else
                tmp = new int[array1.length];
        for (int i = 0; i  < array1.length; ++i) {
                tmp[i] = array1[i];
        }
        for (int i = 0; i  < array1.length; ++i) {
                array1[i] = array2[i];
        }
        for (int i = 0; i  < array2.length; ++i) {
                array2[i] = tmp[i];
        }
}
```

Ex 8.11

A list of employees objects would use ArrayList since the number of the employees in a company can change monthly and ArrayList has dynamic size. A monthly planner would use an array to represent 12 months in a year since the number of months in a year never change hence ArrayList is not needed.

PP 8.2

Source code below:

```
package hw2;

import java.util.Scanner;

public class CountOccurrence {

	public static void main(String[] args) {
		final int MIN_RANGE = -25;
		final int MAX_RANGE = 25;
		final int SENTINEL_VALUE = 26;

		int input;
		int[] container = new int[MAX_RANGE - MIN_RANGE + 1];
		Scanner scan = new Scanner(System.in);

		do {
			System.out.println("Enter " + SENTINEL_VALUE + " to
exit.");
			System.out.print("Enter an integer between " +
MIN_RANGE + " to " + MAX_RANGE + ": ");
			input = scan.nextInt();
			if (input >= MIN_RANGE && input <= MAX_RANGE)
				++container[input + 25];
			else if (input != SENTINEL_VALUE)
				System.out.println("Input is out of range!\nTry
again!");

		} while (input != SENTINEL_VALUE);

		System.out.println();

		for (int i = 0; i < (MAX_RANGE - MIN_RANGE + 1); ++i) {
			if (container[i] > 0) {
				System.out.println("Input number: " + (i - 25));
				System.out.println("Number of occrrences: " +
container[i]);
				System.out.println();
```

```
                }
            }
        }

}
```

Input/output below:

```
Enter 26 to exit.
Enter an integer between -25 to 25: -25
Enter 26 to exit.
Enter an integer between -25 to 25: 25
Enter 26 to exit.
Enter an integer between -25 to 25: -26
Input is out of range!
Try again!
Enter 26 to exit.
Enter an integer between -25 to 25: 27
Input is out of range!
Try again!
Enter 26 to exit.
Enter an integer between -25 to 25: -8
Enter 26 to exit.
Enter an integer between -25 to 25: -8
Enter 26 to exit.
Enter an integer between -25 to 25: 19
Enter 26 to exit.
Enter an integer between -25 to 25: 19
Enter 26 to exit.
Enter an integer between -25 to 25: 19
Enter 26 to exit.
Enter an integer between -25 to 25: 19
Enter 26 to exit.
Enter an integer between -25 to 25: 6
Enter 26 to exit.
Enter an integer between -25 to 25: 26

Input number: -25
Number of occrrences: 1

Input number: -8
Number of occrrences: 2

Input number: 6
Number of occrrences: 1
```

```
Input number: 19
Number of occrrences: 4

Input number: 25
Number of occrrences: 1
```