**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

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 6.7 a and c

Chapter 7

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

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