Name: Key UTA ID:

Instructions:

1. The test is worth 100 points. The point value of each question is given with the question. There are also extra credit questions at the end worth 10 points.
2. The test is open book and open notes for all printed and hand-written material. You may NOT bring an electronic book or any electronic device to use during the test (no computer, no smart phone, etc.) You may use as much printed or written material as desired including copies of code examples.
3. You will write your answers on the test pages. If additional space is needed, you may use the back of the pages. Please make a note on the test page whenever your answer continues onto another page and indicate where the answer is.
4. All questions that start with the same number are related so look back at the earlier parts of the questions as needed.
5. Please write legibly. Your writing should readable if the test is sitting on a desk in front of me. I am not looking for perfect handwriting but it does need to be legible. Also it needs to be LARGE enough for me to read. I will deduct points if your answers are much more difficult to read than those of the general student whether that is from poor handwriting or microscopic handwriting.
6. If you have a question during the test, please raise your hand. The TA and I will be available to come hear your question. Sometimes we may not be able to answer your question because it gives you too much information but you should always ASK. This includes asking about words you don’t understand, etc.
7. If you write the phrase “I read the instructions” after your answer to the final extra credit problem, you will get another 2 points.
8. I recommend that you read quickly through all the questions before starting to spend a lot of time on any one question. You may decide to work on a question that is worth more points first.
9. You have 2 hours and 30 minutes to complete the test.

1.a) Given the following java code, what will be printed to the screen? Below or beside the code, write what the output will look like on the screen. {10 points}

public class TestQuestion

{

 public static void main(String[] args)

 {

 int arr[] = {10, 20, 30, 40, 50};

 printBox(arr);

 }

 public static void printBox(int x[])

 {

 for(int i = x.length - 1; i >= 0; i--) //for loop header

 {

 System.out.println(x[i]); //output statement

 }

 }

}

[Note that the index goes from the highest value x.length-1 DOWN to 0 in the printBox method]

50

40

30

20

10

1.b) Change the *for* above loop to print the numbers in the array in the opposite order from the previous question. Just write the new for loop header below. {6 points}

 for(int i = 0; i < x.length; i++) //for loop header

1.c) If the output was to be written to a file named “printBoxFile.txt” instead of to the screen, write only the Java statements that would be needed to set up the file for output, and then rewrite the output statement that would write to this file instead of the screen. {6 points}

[There is more than one way to do this but basically create some kind of a File variable and hook it to some kind of Writer variable.]

 FileWriter fw = new FileWriter("printBoxFile.txt");

 BufferedWriter bw = new BufferedWriter(fw);

 bw.println(x[i]);

2. Write a method *sumOdd* that asks the user to enter a non-negative integer less than or equal to 100. Take the user’s input *userInput*, validate that the input is an integer, and if it is, make sure that it is non-negative. If the input is not valid, print a message that the user’s input was not valid. [Make sure your program will not crash if the user enters a letter.]

 If the input is a valid integer, the print the sum of all the ODD numbers from 1 up to and including the *userInput*. As an example, if the user enters *8*, then the method should print *16* because this is the sum of 1+3+5+7. Also, if the user enters zero (0), output a zero. {16 points}

public static int sumOdd() // not the only answer, just has to meet description above

 {

 int sum = -1;

 int start = -1;

 String userInput = new String();

 Scanner input = new Scanner(System.in);

 System.out.print("Please enter a non-negative integer: "); //asks the user

 userInput = input.next(); // user’s input *userInput*,

 System.out.println("\nYou entered "+userInput);

 Scanner user = new Scanner(userInput); // will not crash

 if (user.hasNextInt()) // validate that the input is an integer

 { start = Integer.parseInt(userInput);

 if (start == 0)

 { sum = 0;

 }

 else if (start > 0) // make sure that it is non-negative

 { sum = 1;

 int odd = 3;

 while(odd <= start) //the print the sum of all the ODD

 { sum = sum + odd;

 odd = odd + 2;

 }

 }

 else

 { System.out.println("Your input was not a non-negative integer."); // print a message

 }

 }

 else

 { System.out.println("Your input was not a non-negative integer."); // print a message

 }

 return sum;

 }

3. Given the class Multiples below, answer the questions that follow.

package multiples;

import java.util.Arrays;

public class Multiples

{

 public static void main(String[] args) {

 int[] result;

 result = new int[10]; // line A

 result = multiples(4,60,10); // line B

 }

 public static int[] multiples(int a, int b, int arraySize){

 int index=0;

 int multi=a; // line C

 int[] result;

 result = new int[arraySize]; // line D

 while((multi < b) && (index < arraySize)){ // line E

 multi = a \* (index + 1); // line F

 result[index] = multi; // line G

 index++;

 }

 return result;

 } }

3.a) List the values that will be in the array *results* when the main routine ends. List the value above the array element name. Add as many lines as needed to list all the values: {10 points}

 4 8 12 16 20 24 28 32 36 40

*results[0] results[1]*

3.b) Using the line letters (A to G) to the right of the code, tell which ONE line would change and what its new values would have to be if we wanted to have *multi* be initialized to *2*, *result.length = 8*, and the maximum value in *result* to be *14*. {8 points}

line B : result = multiples(2,14,8);

3.c) Describe the pattern of values that are put into *result*, what value the pattern starts with, i.e. which thing in the code represents the first value in the pattern, what value it ends with (which variable in the code), and how you would determine the number of values that are put into *result*. {10 points}

The values are multiples of *a*

The pattern starts with the value of *a*  // partial credit for specific number

Pattern ends with *b* // partial credit for specific number

Number of values in *results* is *arraySize*

4. Convert the numbers below. {6 points}

 0100101000000012 = 16

0010 0101 0000 0001

 2 5 0 1 = 250116

 2453110 = 101 1111 1011 1111 2

24531 - rewriting rewriting

16384 979 - 31 -

8147 - 512 16

4096 457 - 15 -

4051 - 256 8

2048 191 - 7 -

2003 – 128 4

1024 63 - 3 -

979 - 32 2

512 31 1 –

 1 0

5. Using the code fragment below, what will be printed to the screen? Beside the code, write what the output will look like on the screen. {10 points}

 ArrayList<Integer> x,y,z; Write the output below here

 x=new ArrayList<Integer>(); Array x

 y=new ArrayList<Integer>(); 1 3 7 8 9

 z = x;

 Array y

 x.add(1); 4 5 6

 x.add(2);

 x.add(3); Array z

 1 3 7 8 9

 y.add(4);

 y.add(5);

 y.add(6);

 z.add(7);

 z.add(8);

 z.add(9);

 x.remove(1);

 System.out.println("Array x");

 for(Integer d: x) {

 System.out.print(d+" ");

 }

 System.out.println();

 System.out.println("\nArray y");

 for(Integer d: y) {

 System.out.print(d+" ");

 }

 System.out.println();

 System.out.println("\nArray z");

 for(Integer d: z) {

 System.out.print(d+" ");

 }

 }

6) Given a Scanner called qTPi connected to a file, read in the data from a single line in the file. The data, in order, is a floating point number, a phrase, an integer, then the remainder of the current line of data. There will be a ~~semi-colon~~ comma and a space at the end of the last word in the phrase, otherwise there is one space between each number or word. A phrase consists of one or more words. Example input is : *98.6 plus or minus, 1 degree F is a normal body temperature*

Your code must read in the data as typed input not as one entire string. You may not use the split method but you may use delimiters. Use the declarations below for the variables to store the data and write your code below the declarations. {10 points}

Scanner qTPi = new Scanner(file);

double dogDare; // or use double doubVal;

int itNice; // or use int intVal;

String justAPhrase = new String(); // or use String phrase = new String();

String itAlong = new String(); // or use String endLine = new String();

// just write the lines of Java needed to read in and store the input data.

dogDare = qTPi.nextDouble();

qTPi.useDelimiter(“, “);

justAPhrase = qTPi.next();

qTPi.findInLine(“, “);

qTPi.useDelimiter(“ “);

itNice = qTPi.nextInt();

itAlong = qTPi.nextLine();

7. The code below is a modification of the code in question 3. There is a missing loop in this revised main method. Read through this revised code, decide what looping structure to use, then below the code, write your chosen loop header and your chosen loop footer. Remember that your loop structure must include some sort of test to stop the loop. An infinite loop, a continue statement, or a break statement will get 0 points. {8 points}

public static void main(String[] args) {

 int[] result = new int[10];

 boolean moreNum = false;

 Random start = new Random(2000);

 Random limit = new Random(5000);

 int st, lmt;

 int iter = 1;

 String yORn;

 Scanner userIn = new Scanner(System.in);

// missing loop header - Position AA

 st = start.nextInt(20);

 lmt = limit.nextInt(50); // note that in the multiple method,

 result = multiples(st,lmt,10); //(multi < b) is changed to (multi < b\*b)

 System.out.println("The values in results on iteration "+

 iter++ +" are:");

 for (int r:result){

 System.out.print(r+" ");

 }

 System.out.println(); System.out.println();

 System.out.println("Do you want to see another example run? "

 +"Type Y or y to continue, N to end: ");

 yORn = userIn.next();

// missing end of loop - Position ZZ

}

/\* Assume the multiple method from question 3

 is here with the change mentioned above. \*/

The loop header at Position AA code should be:

//There are multiple possible answers

do {

The loop footer at Position ZZ code should be:

} while (yORn.equalsIgnoreCase(“y”));

Extra Credit Questions

XC1. If a new project is created in Netbeans named *Maverick*, what is the name of the package that Java creates and what is the name of the class that is created? Be exact. {2 points}

Package name: maverick

Class name: Maverick

XC2. Name three primitive data types in Java excluding *int* and *double.* {2 points}

long, short, byte, boolean,

XC3. Name three Java classes that you have used in writing your lab assignments. {2 points}

Scanner, String, Random,

XC4. For the set of words below, draw an arrow from each word or phrase to the correct part of the method header that it refers to. One arrow is done as an example. {2 points}

 Inputs Visibility Name Formal parameters Scope Return type

 public static double tempKelvin( double tempFahr) {

XC5. Write a (politely worded) rhyme about Java. {Any answer will receive 2 points}

// answer to question

“I read the instructions”