‘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. I will deduct points if your answers are much more difficult to read than those of the general student.
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. You have 1 hour and 20 minutes to complete the test.

1. Assume that you have a list of ages, given in months, for a large group of pets. The ages will be read in from a file and you do not know how many ages will be read in. You do know that there are no more than 40 ages in the file.

1.a) Declare an array that will hold all the ages in the file. Use the class Integer for the data type. Make sure that space is allocated for your array by Java.

 {4 points}

Integer[] ages = new Integer[40];

1.b) Assume that the ages are in a file called “*ages.txt”* . Write the Java code to declare and initialize a File variable and connect it to the file. Then use the File variable to create a Scanner variable called *inFileAge*. {4 points }

// Accepted answer:

File inFile = new File(“ages.txt”);

Scanner inFileAge = new Scanner(inFile);

//---------------- More accurate answer – gets 2 points XC

 File inFile = new File("ages.txt");

 Scanner inFileAge;

 try

 {

 inFileAge = new Scanner(inFile);

 }

 catch (FileNotFoundException fnf)

 {

 }

1.c) Using *inFileAge* , write a loop in Java code that will read all of the ages from *inFileAge* and store those ages into the array you declared in part a). When you have read all the ages, make sure that you also know how many ages you have read in. Be sure to declare any new variables you use. {10 points }

int ageCount = 0; // will have the count of elements in the array when loop is done

 // you can’t use .length because array is size 40 but there may be less values in file

while (inFileAge.hasNextInt())

{

ages[ageCount] = inFileAge.nextInt();

ageCount ++;

}

1.d) Now that all the ages are in your array, write an algorithm in pseudocode to find the smallest age and return the index of that age. {8 points }

Declare variables to hold the smallest value (*smVal*) and the index of the smallest value (*smIndex*)

Loop through the array and stop when loop counter *l* == *ageCount*

 If age[l] is smaller than *smVal*

 *smVal* gets age[l] and

 *smIndex* gets l

1.e) Write a complete method that will take in an age in months as its input parameter. The method should take in the number of months and then return a double that is the year equivalent of the input number of months. For example, if the input is 28 months, then the output should be 2.3333 years. {10 points }

public static double moToYear( int months) // will consider int and double as well Integer

{

 return months / 12.0;

}

2) Use this code fragment to answer the questions below

Scanner input = new Scanner(System.in);

int[] seconds = { 577, 340, 221, 655, 443, 814, 457, 143, 200, 925 };

int difference = 100000;

int cIndex = -1;

System.out.printf("Enter your choice of seconds: "); 427 0

int userSeconds = input.nextInt(); // 150 190 1

 71 2

for (int i=0; i < seconds.length; i++) 7 7

{

 if (Math.abs(userSeconds - seconds[i]) < difference)

 {

 difference = Math.abs(userSeconds - seconds[i]);

 cIndex = i;

 }

}

System.out.println("Index "+cIndex+" of value "+seconds[cIndex]

+" is the output for your preferred input of "+userSeconds);

2.a) Fill in the blanks in the questions below: {8 points }

The length of the seconds array is : 10

The input comes from where in Java ? keyboard / System.in (tricky!)

The method used from the Math class finds : absolute value

The maximum value of cIndex is : 9

2.b) Why is cIndex initialized to -1? {4 points }

Because -1 cannot be used as an array index so initially it does not indicate any particular value in the array as the preferred output element

2.c) For the following inputs, what are the corresponding outputs? {8 points }

 Input Output

 450 4, 443

 204 8, 200

 755 5, 814

 387 1, 340

2.d) Describe in words what this code fragment is doing. Don’t just write the Java lines in words. Explain the purpose. Keep your answer to four sentences or less. {8 points }

The code has a list of numbers which indicate some number of seconds. The user enters their choice of seconds and the code returns the value in the list that is closest to the entered number and the index of where that value is in the list.

3) You have been asked to create a program to hold a playlist of songs so that the listener can choose different groups of songs based on some of the information in the playlist. The data for the playlist will be in a file. A listener will start the program and define what group of songs they want to hear by certain criteria. The program will then look through the list of songs in the file and will print out all of the songs that meet the criteria for the group. The possible criteria will be chosen from the following song information:

Song number

Artist\_name

Length of song in minutes and seconds (two separate numbers: *minutes* *seconds*)

Genre (single word, one of *pop, rock, blues, r&b, folk*, *alternative, heavy, classical, other* )

Performance style (choice of *solo, vocal group, instrumental*) // removed underscore

Decade (*40, 50, 60 , 70, 80, 90, 00, 10*)

Title

The file will have lines of data in the following form:

*Song number Artist name, Minutes Seconds Genre Performance style, Decade Title*

As an example:

*1 The Beatles, 4 23 pop vocal group, 70 All You Need Is Love*

The user will be told how to choose their criteria and will select up to 3 criteria for a song to meet. The program will go through the file and print the information about all songs that meet the criteria.

3.a) Why does the line of input data have two commas in it and why are they in the specific places that they are located? Keep your answer no longer than 3 sentences. {10 points}

Both commas delimit the end of elements in the input that are Strings that can have multiple words, *Artist name* and *Performance style*. Java can read single words (.next) or entire lines/end of lines (.nextLine) but there is no direct way to read a group of words.

3.b) Write declarations for the variables that would be needed to read and save one line of song data from the file. {10 points}

int songNumber = 0; // Song number

String artistName; // Artist\_name

int minutes, seconds; // Length of song in minutes and seconds

// (two separate numbers: *minutes* *seconds*)

String genre; // Genre (single word, one of *pop, rock, blues, r&b,*

// *folk*, *alternative, heavy, classical, other* )

String performanceStyle; //Performance style (choice of *solo, vocal group, instrumental*)

int decade; // Decade (*40, 50, 60 , 70, 80, 90, 00, 10*)

String title; // Title

3.c) Describe in words how you would read the *Artist name* from the file. List the command(s) that you would use and describe why you would read this value in this way. {8 points}

Java can read single words (.next) or entire lines/end of lines (.nextLine) but there is no direct way to read a group of words like Artist name. Therefore the setDelimiter command would be used to indicate that there is a comma delimiter to look for at the end of the next element to read.

Commands used: setDelimiter(”, “);

3.d) Write an algorithm that will convert the two values *minutes* and *seconds* into a single value of *songSeconds*. This value will contain the whole length of the song in seconds. This is a very small algorithm. {8 points}

// this can be pseudocode or Java

Given *minutes* and *seconds* , multiply *minutes* times 60 and add this product to *seconds*

Extra Credit

XC1) Using the array *seconds* from question 2 and the variables defined below, write a loop in Java that goes through every element of the array and performs two tasks. The loop should find the smallest value in the array and the loop should work on finding the average of the array. Following the loop, there should be one Java statement to find the average. You do not have to re-declare the *seconds* array. {5 points}

double secSmallest = 10000; // At the end, this should have the smallest value and

int smallest = 10000; // this should be the index of the smallest value

double secSum = 0; // This will collect the sum of the values and

int secCount = 0; // this could count the number of elements in *seconds*

double secAverage = 0;

int[] seconds = { 577, 340, 221, 655, 443, 814, 457, 143, 200, 925 };

for (int j = 0; j < seconds.length; j++)

{

if (seconds[j] < secSmallest)

{

secSmallest = seconds[j];

smallest = j;

}

secSum += seconds[j];

secCount++;

}

secAverage = secSum/secCount;

-----------------------------------

for (int secCount = 0; secCount < seconds.length; sceCount++)

{

if (seconds[secCount] < secSmallest)

{

secSmallest = seconds[secCount];

}

secSum += seconds[secCount];

}

secAverage = secSum/secCount;

XC2) Write a few lines of Java code to read in a string from the user (keyboard), check to see if the string has a double as the first element and if so, parse the Double out of the string and store it to a variable called *percentage*. {3 points}

// fix before posting

Scanner input = new Scanner(System.in);

String checkDouble = input.nextLine();

Scanner inString = new Scanner(checkDouble)

if (checkDouble.hasNext())

{

 Double percentage = inString.nextDouble();

}

XC3) What is the most difficult thing about the class so far? Keep your answer short - no more than two sentences. {Any answer will receive 2 points}