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.
2. Read all of the instructions for each question and answer what is asked. Do not write down random stuff if you don’t know the answer.
3. All questions that have the same question number are related to each other. However, they are not all necessarily dependent on each other so you can skip around if needed.
4. 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.
5. You will write your answers on the test pages. If additional space is needed, you may use the backs or bottoms of the pages or extra paper. Please make a note on the test page whenever your answer continues onto another page and indicate where the answer is. Please staple all extra pages to your test when you turn it in.
6. Please write legibly and large enough to read easily. Your writing should be 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.
7. For multiple choice questions, circle the letter of your answer choice. Circle only one choice for your answer unless the instructions direct otherwise.
8. If you read this instruction and then write a phrase or sentence at the bottom of the extra credit page which tells me that you read this instruction, you’ll get two extra points.
9. 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.
10. You have 1 hour and 20 minutes to complete the test.

1.) Contents of file “pets.txt” Line #s

*1 Horses; 1 Quarter Horse 2* 1

*1A Quarter Horse; Baby Two 2018 Dancer 2001* 2

*2 Dogs; 3 Chihuahua 1 Dachshund 3 Corgi 1* 3

*2A Chihuahua; Buddy 2005* 4

*2B Dachshund; Ruthie 2003 Winston 1997 Inigo 2017* 5

*2C Corgi; Goldie Bear 2002* 6

*3 Cats; 2 Domestic Shorthair 4 Birman 2* 7

*3A Domestic Shorthair; Smidgen 2009 Cassat 2006 Erte 1999 Phouka Cat 1997* 8

*3B Birman; Upbeat Ringo 1995 Ultimate Sugar 1995* 9

int animalLabel1, animalLabel2, animalLabel3; // numbers at beginnings of lines listing animal types

String animalType1, animalType2, animalType3; // names of animal types that ends at “;”, ex. “Cats”

int numOfBreeds; // value after “;” telling number of breed of this animal in the file

String breed1, breed2, breed3; // names of the breeds for this animal in the file, ex. “Birman”

int breedCnt1, breedCnt2, breedCnt3; // number of animals of each breed in the file

String breedLabelA, breedLabelB, breedLabelC; // number-letter labels on each breed, ex. “1A”, “2C”

int breedNumA, breedNumB, breedNumC ; // number part of a breedLabel, ex. 2 from 2C

String breedLetA, breedLetB, breedLetC; // letter part of a breedLabel, ex. “C” from 2C

String breedTypeA, breedTypeB, breedTypeC; // the specific breed name that ends at “;”, ex. “Corgi”

String animalName1, animalName2, animalName3, animalName4; //ex. “Ultimate Sugar”

int animalYOB1, animalYOB2, animalYOB3, animalYOB4; // Year Of Birth = YOB, ex. 2002

File petsFile = new File(“pets.txt”);

Scanner inFile; // assume this will be correctly assigned the File variable in a try-catch block

1.a) Look at the data given in the file above and the variables that are defined and then list which variables should be used to store the data on line # 7. This means to list the names of the variables that would be used to match up with the data in line # 7 in order. {10 points}

*//3 Cats; 2 Domestic Shorthair 4 Birman 2* 7

animalLabel3 animalType3 numOfBreeds breed1 breedCnt1 breed2 breedCnt2

1.b) Use the file info given above, the variables you listed in question 1.a, and the Java code fragment below to answer the questions below: {9 points; 3 pts each}

animalLabel2 = 0;

boolean validTopLabel = false;

int animalLabel = 0;

String tempLine = “”;

while ((validTopLabel == false) && inFile.hasNext())

{ try

{ animalLabel = inFile.nextInt();

validTopLabel = true;

inFile.nextLine();

}

catch (InputMismatchException ime)

{ validTopLabel = false;

tempLine = inFile.nextLine();

}

}

validTopLabel = false;

animalLabel2 = animalLabel;

animalLabel = 0;

1.b.i) In the code above, *validTopLabel* indicates: (Circle the letter of your choice)

A. that the top level of the file is valid

B. that the code was able to read an integer for the value of *animalLabel*

C. that the next value in the file will be an integer

D. that inFile has another value to read

1.b.ii) In the code above, the *InputMismatchException* : (Circle the letter of your choice)

A. could be the result of inFile.nextInt()

B. will only be true if *animalLabel* is an integer

C. may occur because of inFile.nextLine()

D. would be used if *validTopLabel* is true

1.b.iii) In the code above, *animalLabel2*: (Circle the letter of your choice)

A. is the temporary variable used to read in the data from the file

B. is used to store the label “1A”

C. is the name of a particular animal in the file

D. is the variable used to store the second animalLabel number that is found

1.c) Use the code below to answer the following questions by filling in the blanks:

{24 points total; 4 pts each}

if (breed1.equalsIgnoreCase(breedTypeA))

{ animalName1 = readWith09Delim(inFile);

animalYOB1 = inFile.nextInt();

if (breedCnt1 >= 2)

{ animalName2 = readWith09Delim(inFile);

animalYOB2 = inFile.nextInt();

if (breedCnt1 >= 3)

{ animalName3 = readWith09Delim(inFile);

animalYOB3 = inFile.nextInt();

} } }

*//… more Java code will be here and then main method ends*

public static String readWith09Delim(Scanner inFile)

{ inFile.useDelimiter("[0-9]");

String value = inFile.next();

inFile.reset();

return value;

}

1.c.i) In the code above, any digit from 0 to 9 is being set as the \_delimiter\_ for inFile.

1.c.ii) The method named \_\_\_\_\_\_readWith09Delim\_\_\_ has 1 input, a Scanner.

1.c.iii) If the data in the input file is wrong and an error occurs with the method call to *nextInt()*,

the most likely Java exception is \_\_\_\_\_InputMismatchException\_\_\_\_

1.c.iv) The variable \_\_\_breedCnt1\_\_\_\_\_ will indicate whether there are more animals to read from the file.

1.c.v) *if* statements inside other *if* statements are referred to as nested\_\_ *ifs*.

1.c.vi) The method *equalsIgnoreCase* is used to compare \_\_String\_\_ variables.

1.d) Write the Java statements needed to read the *breedType* for line #2 (1B should be 1A) from the file. Your program must use the semi-colon (;) as a delimiter for the Scanner attached to the input file and must read the *breedType*, i.e. the kind of animal breed, by setting the delimiter, reading the *breedType* as a word, then resetting the delimiter with the reset method on the Scanner. {8 pts}

// don't need to deal with Label

inFile.useDelimiter("; "); // name ends with :

breedTypeA = inFile.next(); // name

inFile.reset();

breedTypeA = breedTypeA.trim();

1.e) Assume that your program has read in animalName1, animalName2, animalName3, animalName4, animalYOB1, animalYOB2, animalYOB3, and animalYOB4 and that Name1 goes with YOB1, etc. Write the Java code to output these values in a table. The table should have three columns with the first column being a number (1, 2, 3, and 4), the second column being Name# and the third column being YOB#. The table should have 5 rows where the first row is titles of “”, “Name”, and “Birth Year”. The first column should be 3 characters wide, the second column should be 20 chars wide, and the last column should be 10 chars wide. {14 points}

System.out.printf("%3s%20s%10s\n","", "Name", "Born In");

System.out.printf("%3s%20s%10d\n", 1,animalName1,animalYOB1);

if (animalYOB2 != 0)

System.out.printf("%3s%20s%10d\n", 2,animalName2,animalYOB2);

if (animalYOB3 != 0)

System.out.printf("%3s%20s%10d\n", 3,animalName3,animalYOB3);

if (animalYOB4 != 0)

System.out.printf("%3s%20s%10d\n", 4,animalName4,animalYOB4);

2. Use the following information and pseudocode to answer the questions that follow:

A file of input data exists, called grades.txt, that has some number of lines of data

Each line of data is a set of 5 grades for one student

The data on one line represents:

Test1 Test2 HW1 HW2 Quiz1 LastName, FirstName

The Test grades are floating point numbers between 0 and 100

The HomeWork grades are integer numbers between 0 and 100

The Quiz grades are integer numbers between 0 and 10

After the last grade on the line there is a name followed by a comma which is the student's last name. The name can have more that one word like “de la Cruz” or “von Neiderhauser Kiesling”

After the comma to the end of the line is the student’s first name. The name can have more than one word like “Irma Louise” or “Billy Jo”

For all lines of the data in the file,

Read in the 5 grades from one line of the file into five variables

that use the data types that are defined above

Read the names into two String variables

Calculate the final grade for student LastName, FirstName with an equation equivalent to:

FinalGrade = (Test1\*.27)+(Test2\*27/100)+(2\*(HW1+HW2)/10)+(Quiz1\*.6)

and make sure that the calculations using HW1, HW2, and Quiz1 are

handled as floating point

Print out all the grades and the FinalGrade for student FirstName LastName

2.a) Give the percentage values of each grade toward calculating the final grade: {5 points}

Test1 = 27 % Test2 = 27 %

HW1 = 20 % HW2 = 20 %

Quiz1 = 6 %

2.b) The file has an unknown number of lines of data in it. Write the loop header that would let the program read all the lines in the file and stop when there are no more lines of data to read.

{10 pts}

while(inFile.hasNext() ) // is OK but reads words instead of lines -3

while(inFile.hasNextLine() ) // is desired

2.c) In this problem, there might be bad data in the input file. Write Java code to read in the Test1 value in such a way that, if the value is invalid (it is a string for example), the program catches any bad value and replaces it with the value of 0 for the Test1 grade without crashing. Also, use the boolean variable declared below to indicate if the Test1 was invalid. Declare any temporary variables that you might want to use. {12 points}

boolean varValid = true;

try

{

Test1 = grades.nextDouble();

}

catch (InputMismatchException ime)

{

Test1 = 0;

varValid = false;

}

2.d) Write an output statement that would print the five input grades, the final grade, the first name and then the last name of the student. Print each integer in 4 spaces, print each floating point number in 7 spaces including 2 places to the right of the decimal, and print each name in 12 spaces. Be sure to take out unnecessary data in the name strings. {8 points}

System.out.printf("%7.2f%7.2f%4d%4d%4d%9.2f%12s%12s\n",Test1, Test2,

HW1, HW2, Quiz1, finGrade, firstName, lastName);

Extra credit

XC1) Look at the question 1. data given in the file and the variables that are defined and then list which variables should be used to store the data on line # 8. This means to list the names of the variables that would be used to match up with the data in line # 8 in order. {3 points}

// *3A Domestic Shorthair; Smidgen 2009 Cassat 2006 Erte 1999 Phouka Cat 1997*

breedLabelA breedTypeA animalName1 animalYOB1

animalName2 animalYOB2

animalName3 animalYOB3

animalName4 animalYOB4

could use breedNumA and breedLetA instead of breedLabelA

XC2) Using the info from question 2, assume that you want to keep track of the student with the highest final grade in the file as you go through and calculate the grades. Using the variables given below, write a few lines of pseudocode that would go inside the loop (2.b) and would find and save the highest grade and the student with that grade. {5 points}

double highGrade = 0;

String highLastName = “”;

String highFirstName = “”;

// declarations have to be outside of loop

if (FinalGrade > highGrade)

{

highGrade = FinalGrade;

highLastName = LastName;

highFirstName = FirstName;

}

XC3) Make up a word to describe your feelings about writing Java programs in CSE 1310.

{ANY polite answer will receive 2 points.}

1. If you read this instruction and then write a phrase or sentence at the bottom of the extra credit page which tells me that you read this instruction, you’ll get two extra points.