Alan Palayil

Anwar Adam

**CS 115 Fall 2019 Lab #10**

Due: **Monday, November 25th, 5:00 PM**

Points: **20**

**Instructions:**

1. Use this document template to report your answers and create separate java files for your classes. Enter all lab partner names at the top of first page.
2. You don’t need to finish your lab work during the corresponding lab session.
3. ZIP your Java files and lab report into a single file. Name the file as follows:

LastName\_FirstName\_CS115\_Lab10\_Report.zip

1. Submit the final document to Blackboard Assignments section before the due date. No late submissions will be accepted.
2. ALL lab partners need to submit a report, even if it is the same document.

**Objectives:**

1. (2 points) Break a problem down into classes with attributes and methods.
2. (3 points) Design, code and test programs that require arrays.
3. (5 points) Write a program that utilizes reading from file with a Loop (use hasNext() method to recognize EOF (End of File)). Design, code and test programs that require arrays.
4. (5 points) Design, code and test programs that require arrays.
5. (5 points) Design, code and test programs that require arrays.

**Problem 1 [2 points]:**

Break a problem down into classes with attributes and methods. Design the following classes (**no code necessary**).

What are attributes necessary, their data types and valid ranges?

What are the methods necessary, their argument data types and their return value data type?

* Create classes that simulate a weather forecast tracking system (we recommend 3 classes). A weather forecast for a day has the high temperature forecast, low temperature forecast, and the sky condition (sunny, cloudy, rainy, snowy). There are three forecasts for each day, 3 days prior, two days prior, and prior day. We also want to keep track of the actual high and low temperature and sky condition so we can later calculate some statistics on the accuracy of the predictions. We want to keep track of 30 days-worth of weather predictions/actuals.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Variable name | Input or output | Data type | Constraints | Special cases | Comments |
| highTemp | Input | float data type  with Fahrenheit as the unit | The range has to be between -50 and 150 | The value has to be between -50 and 150 | This variable is used to measure the highest temperature of a day for a month |
| lowTemp | Input | float data type with Fahrenheit as the unit | The range has to be between -50 and 150 | The value has to be in the range | This variable is used to measure and keep record of the lowest temperature of a certain day for a month |
| skyConditions | Input | String data type | The values has to be (sunny, cloudy, rainy, and snowy) | The values have to be string literals and nothing else | This variable is used to measure tell the weather conditions |

Methods: getAverageTemperature, getHighestTemperature, getLowestTemperature

**Problem 2 [3 points]:**

Design, code and test programs that require arrays.

Recall the BobsLife class from one of the previous labs. After writing the class, and within the LongLiveBob program, you were trying to come up with a sequence of a series of three statements - move(you determine the location), nextTime(), and outputting Bob - again and again without causing Bob to die or go to jail. Your code might have looked something like this

// repeated sequence

b.move("home");

b.nextTime();

System.out.println(b);

b.move("home");

b.nextTime();

System.out.println(b);

b.move("home");

b.nextTime();

System.out.println(b);

b.move("work");

b.nextTime();

System.out.println(b);

b.move("gym");

b.nextTime();

System.out.println(b);

// and so on

Since then we have learned about loops and arrays, which should help you make the code in the LongLiveBob program simpler, and be able to run longer simulations of repeated statements. Your challenge is to implement the pattern of locations to move (and keep Bob alive) to in an array, and use a loop to simulate 1000 moves.

**Problem 3 [5 points]:**

Write a program that utilizes **reading from file with a Loop** (use hasNext() method to recognize EOF (End of File)). Design, code and test programs that require arrays.

Consider a system for processing student test scores. The following class will be used as part of this system and contains a student's name and the student's answers for a multiple-choice test. The answers are represented as "char" with an omitted answer being represented by the question mark '?'. These answers are stored in an array in which the position of the answer corresponds to the question number on the test (question numbers start at zero). A student's score on the test is computed by comparing the student's answers with the corresponding answers in the answer key for the test. One point is awarded for each correct answer and 1/4 point is deducted for each incorrect answer. Omitted answers do not change the student's score.

Write the class StudentAnswerSheet including the following attributes/fields & methods:

* Any necessary instance variables, class variables and constants.
* Non-default constructor - takes a student name (string) and an array of char (student answers) as argument s and loads this StudentAnswerSheet. No set\* methods are required, **do all data validation in the constructor**.
* public String getName() - returns the name of the student for this StudentAnswerSheet
* public static void setKey(char [] answerKey) - sets the class variable with the answer key
* public double getScore() - scores this StudentAnswerSheet using the class variable answer key and according to the rules explained above. You should verify that the "answerKey" has as many answers as the StudentAnswerSheet or else return negative infinity (Double.NEGATIVE\_INFINITY).
* public String toString() method that **outputs the student name, their answers and their score**.

Now write a program to read a comma-delimited input file containing the answer key in the first row and student names/answers in other rows, one per line (Go to Blackboard -> Assignments and Projects -> Labs -> Lab files -> Lab #10 files to get the file: answers.txt).

First read and load the key. Then read each student, grade and output. Also determine the highest scoring student (if ties for highest scoring save all their names). You do not know how many questions are on the test until you read the first line of the file, which is always the key. Here is sample output:

Matt a e b b ? b c d a e score=7.75

Cami e e b b ? b c d a e score=9.0

John ? ? ? ? ? ? c d a e score=4.0

Mary a a a a a a c d a e score=2.5

Fred e e b b ? b c d a e score=9.0

Best Student(s): Cami Fred

**Problem 4 [5 points]:**

Write a class that requires class attributes/methods and iteration.

Write a class for the following problem. The traits of a particle determine A, B, C, D - real number coefficients for location and velocity polynomials.

location(t) = At^4 + Bt^3 + Ct^2 + D\*t

velocity(t) = 4At^3 + 3Bt^2 + 2C\*t

where "t" is the time in real seconds. Scientists want to be able to create a table (given start, end and increment times) of the particle's location and velocity (each to three decimal places of precision). The table should also output a message within the table if the difference between the previous location and current location is more than one unit distance.

Finally, they also want the class to keep track of the largest velocity output in all particle object calls to the table method. Here is an example:

Enter the values for a, b, c, d: -.015 .005 -.02 .1

Particle Object Created

location(t) = -0.015t^4+0.005t^3+-0.02t^2+0.1t

Enter the start time: 2

Enter the end time: 5

Enter the increment: .5

Time Loc Vel

2.0 -0.080 -0.500

2.5 -0.383 -0.944

3.0 -0.960 -1.605

3.5 -1.932 -2.529

4.0 -3.440 -3.760 More than 1 unit movement

4.5 -5.650 -5.344 More than 1 unit movement

5.0 -8.750 -7.325 More than 1 unit movement

Max Velocity in all tables = -0.500

Enter the values for a, b, c, d: -.015 .005 .02 .1

Particle Object Created

location(t) = -0.015t^4+0.005t^3+0.02t^2+0.1t

Enter the start time: 2

Enter the end time: 5

Enter the increment: .5

Time Loc Vel

2.0 0.080 -0.340

2.5 -0.133 -0.744

3.0 -0.600 -1.365

3.5 -1.442 -2.249

4.0 -2.800 -3.440 More than 1 unit movement

4.5 -4.840 -4.984 More than 1 unit movement

5.0 -7.750 -6.925 More than 1 unit movement

Max Velocity in all tables = -0.340

Design (what are attributes and methods needed, constants, static attributes/methods), code, and test your Particle class.

To get the test program with method names expected go to Blackboard -> Assignments and Projects -> Labs -> Lab files -> Lab #10 files to get the file particleApp.java.

**Problem 5 [5 points]:**

Design, code and test programs that require arrays.

We want to write a program to do some integer vector arithmetic, addition and dot product. You can represent the vector using an array of integers. Write three methods for the below program, and call those methods to create the sample output.

* vectorToString - takes an integer array argument and returns a nicely formatted String (see below). A null or empty vector will return "<>"
* add - takes 2 integer array arguments and returns the vector sum as an integer array. Output an error message if the vector sum cannot be performed and return null.
* dotProduct - takes 2 integer array arguments and returns the vector dot product as an integer. Output an error message if the vector dot product cannot be performed and return 0l.

public class Vector {

public static void main(String[] args) {

int [] v1 = { 1, 3, 5};

int [] v2 = { -1, -3, -5};

int [] v3 = { 1, 2, 3, 4};

int [] v4 = { 1, 1, 1, 1};

// call methods to sum each pair of the above vectors and output the vectors and the result

// call methods to dot each pair of the above vectors and output the vectors and the result

}

// write your three methods here

}

Sample Output:

<1,3,5>+<-1,-3,-5>=<0,0,0>

Sum not defined, vectors of different length<1,3,5>+<1,2,3,4>=<>

Sum not defined, vectors ofdifferent length<-1,-3,-5>+<1,2,3,4>=<>

<1,2,3,4>+<1,1,1,1>=<2,3,4,5>

<1,3,5>.<-1,-3,-5>=-35

Dot Product not defined, vectors of different length<1,3,5>.<1,2,3,4>=0

Dot Product not defined, vectors of different length<-1,-3,-5>.<1,2,3,4>=0

<1,2,3,4>.<1,1,1,1>=10