Final Examination Solution

CS 1323 Section 10, Spring 2015

Name (printed legibly):

Student number:

**Do not write on the back of any pages.**

Answer all programming questions in Java.

Unless otherwise indicated, each part of a problem is worth the same number of points.

Show your work to receive partial credit.

You do not need to include import statements, throw any FileNotFoundException or other exceptions, or use constants in any problem. Although comments can help me understand your solution, and should always be included in programs, they are not required on the examination.

Pay careful attention to whether the question asks for a signature, a code fragment, a method, or a complete program. Do not write a whole program when you are asked for only a few lines of code.

Tables may contain too many or too few lines. If there are too many, leave the extras blank. If there are too few, add additional lines to the table.

Things in solid boxes are required. Things shown in dotted boxes are not required, but may be used if you wish. Make sure that there are values in all solid boxes. For example, make sure that you’ve written your name and student number above and signed the integrity statement below.

Pay attention to things that are in bold (especially if they are underlined and italicized). These are important statement that I think students will miss.

No problem on this examination requires more than fifteen to twenty lines of code. Most require ten or less. If you are writing pages of code, stop and think about whether there is an easier way to do this.

**Integrity Pledge**

**On my honor, I affirm that I have neither given nor received inappropriate aid in the completion of this exercise.**

**Signature:**

1. (20 points; 6 points for a), 8 points for b), 4 points for c), 2 points for d))

a) Select the best type (int, double, boolean, String, char) for each data element listed below.

i) The percent of voters who fail to vote in an election, rounded to the nearest hundredth.

double

ii) The type of mascot that a political party has (e.g. elephant, donkey).

String

iii) Whether or not voter fraud is detected in an election.

boolean

b) For the mathematical expression, fill in the table below by indicating which operation is performed first, second, etc. and whether a promotion has to be done to complete the operation.

2.1 \* 39 + 5 % 4 / 10.1

|  |  |  |
| --- | --- | --- |
| **Order** | **Operation** | **Promotion (Yes/No)** |
| **First** | \* | Yes |
| **Second** | % | No |
| **Third** | / | Yes |
| **Fourth** | + | No |

c) What value does the variable gallons have at the end of execution of this code?

double gallons = 19.2;

if (gallons < 10.0)

{

gallons = gallons + 20.0;

}

else if (gallons > 20.0)

{

gallons = gallons + 10.0;

}

if (gallons > 20.0)

{

gallons = gallons – 5.0;

}

**gallons contains: 34.2**

d) 39 % 7 = 4

1. (10 points; 4 points for a), 6 points for b))

a) What value does result contain at the end of the loop?

int result = -1;

int[] data = {1, 4, 19, 21, 24, 29, 33, 57, 62};

int target = 33;

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

{

if (data[i] < target)

result = data[i];

}

**result is: 29**

b) What values do x, y, and z contain at the end of the main program? You may use a memory diagram on the right to answer this question if you wish to, or may answer based on your understanding of parameter passing mechanisms. **The question will be graded only on the answer in the box.**

public class Example{ ***OPTIONAL***

|  |  |  |
| --- | --- | --- |
|  | **main stack frame** | |
|  | **Address** | **Contents** |
| X | 100 | 5 |
| Y | 101 | 1000 |
| z | 102 | 1001 |

public static void main(String[] args) {

int x=5;

String y = new String(“AB”);

double[] z = {1.0, 2.0};

y=changeEmUp(x, y, z);

|  |  |
| --- | --- |
| **heap** | |
| **Address** | **Contents** |
| 1000 | AB |
| 1001 | 1.0 |
| 1002 | 2.0 |
| 1003 | GH |
| 1004 | 7.0 |
| 1005 | 5.0 |
| 1006 | 3.0 |

}

public static String changeEmUp(int a, String b, double[] c){

a=6;

b = new String(“GH”);

c = new double[3];

c[0] = 7; c[1] = 5; c[2] = 3;

return b;

}

}

|  |  |  |
| --- | --- | --- |
| **changeEmUp stack frame** | | |
|  | **Address** | **Contents** |
| a | 200 | 5 6 |
| b | 201 | 1000 1003 |
| c | 202 | 1001 1004 |

x is: 5

y is: “GH”

z is: {1.0, 2.0} ***OPTIONAL***

1. (10 points) Perform ***insertion sort*** on the array below. **Do not use selection sort.**

**Show each assignment of data on its own row in the table. Show only the data that is changed.**

***There should be exactly one value in each line of the table that is used.***

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 3 | 5 | 7 | 4 | 2 | 6 | Auxiliary |
|  |  |  |  |  |  | 3 |
|  |  |  |  |  |  | 5 |
|  |  |  |  |  |  | 8 |
|  |  |  |  |  |  | 4 |
|  |  |  | 7 |  |  |  |
|  |  | 5 |  |  |  |  |
|  | 4 |  |  |  |  |  |
|  |  |  |  |  |  | 2 |
|  |  |  |  | 7 |  |  |
|  |  |  | 5 |  |  |  |
|  |  | 4 |  |  |  |  |
|  | 3 |  |  |  |  |  |
| 2 |  |  |  |  |  |  |
|  |  |  |  |  |  | 6 |
|  |  |  |  |  | 7 |  |
|  |  |  |  | 6 |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

1. (10 points) Write a method called repeatElements that takes a reference to an array of Strings and a positive number times and creates a new array of Strings that contains the elements of the original list repeated, as a group, the given number of times. So if the original array contained {“a”, “b”, “c”} and times was 3, the result would be {“a”, “b”, “c”, “a”, “b”, “c”, “a”, “b”, “c”}. The method signature and Javadoc is below. The original array is oversize and the returned array is perfect size.

/\*\* Create a new perfect size array that contains the elements of the original oversize array repeated the given number of times.

@param source A oversized array.

@param sourceSize The number of elements in the source array that have been used.

@param times The number of times each item should be repeated.

@return A perfect size array that contains the elements of the original array repeated, as a group, the given number of times.

\*/

public static String[] repeatElements(String[] source, int sourceSize, int times)

{

String[] result = new String[times\*sourceSize];

for (int t = 0; t < times; ++t)

{

for(int index = 0; index < sourceSize; ++index)

{

result[t\*sourceSize + index] = source[index];

}

}

return result;

}

1. (10 points; 5 points each for a) and b))
2. ***Rewrite the method signature*** (return type, name, parameters) for problem #4 using oversize arrays for both the input and output. **Do not write the body of the method.**

public int repeatElements(String[] source, int sourceSize, String[] result, int times)

1. Show the method from Problem #4 ***being called in code fragment*** using the variables declared below. The code should create an array of 10 String elements with reference source. This array should contains “a” at index 0, “b” at index 1, and “c” at index 2. After the method is called, the array reference destination should point to an array of 15 elements that contains 5 repetitions of “a”, “b”, “c”. You may declare other variables that you need.

String[] source;

int sourceSize;

String[] destination;

source = new String[10];

source[0] = “a”;

source[1] = “b”;

source[2] = “c”;

sourceSize = 3;

destination = repeatElements(source, sourceSize, 5);

1. (10 points; 6 points for a), 2 points for b) 2 points for c)) An Iterator<String> object is used to step through items in the ArrayList<String> class. The UML for the Iterator class is below. The data is not shown in the UML.

Iterator<String>

-hasNext(): boolean

-next():String

-remove() : void

The API describes these methods as follows.

The hasNext() method returns true when there are more elements to be returned.

The next() method returns the next element in the iteration.

The remove() method removes the last element returned by this iterator. This method can be called only once per call to next().

The ArrayList<String> class contains a method with the signature below:

public Iterator<String> iterator()

This method is called a factory, because its job is to construct and return an Iterator object for an ArrayList<String> object.

a) Suppose we have a ArrayList<String> object with **reference myList** that has been declared, constructed, and may or may not have had some values added to it. Write a code fragment that will remove all of the elements from the myList, leaving it empty.

Iterator<String> iter = myList.iterator();

while(iter.hasNext())

{

iter.next();

iter.remove();

}

b) List one method in the Iterator class that is an accessor.

next(), hasNext()

c) List one method in the Iterator class that is a mutator.

remove(), next()

1. (30 points; 10 points each for a), b) and c))

Write a program that will manage the National FoozBall League Draft. The draft starts with a list of players who wish to play in the National FoozBall League. This list is stored in a file. The player names are in sorted order. Each of the teams in the league then picks which players they want, one at a time, for a given number of rounds. The order in which the teams choose is stored in a second file and is the same for every round.

The format of both files is:

Name 1

Name 2

**The team names are stored in a perfect sized array.**

**The player names are stored in an ArrayList.**

The program should announce the name of the team that chooses next. Then the program should allow the user to enter the name of a player. The program should verify that the player is both in the draft and not already drafted by checking the list of players who are still available. If so, the program should announce “Team X has drafted player Y”, and remove the player from the list of players that are still available to be drafted. You may assume all player names are unique. The interaction is as below.

Welcome to the NFL Draft.

The New York Planes will choose next.

Jalen Saunders

Jalen Saunders is drafted by the New York Planes.

The Jacksonville Cougars will choose next.

Aaron Calvin

There is no Aaron Calvin available in the draft.

The Arizona Garter Snakes will choose next. **// Notice—they lost their turn**

Sam Bradford **// from a typo!**

Sam Bradford is drafted by the Jacksonville Cougars.

The Detroit Bears will choose next.

Aaron Colvin

There is no Aaron Colvin available in the draft. **// He’s already been drafted!**

The Chicago Tigers will choose next. **//Detroit loses their turn too!**

…

a) Write the method below. Remember that the file is formatted as follows:

Name 1

Name 2

etc.

String[] readFileIntoArray(String fileName)

{

int size = 0;

Scanner file = new Scanner(new File(filename));

while (file.hasNextLine())

{

++size;

file.nextLine();

}

file.close();

String [] result = new String[size];

file = new Scanner(new File(filename));

int count = 0;

while (file.hasNextLine())

{

result[count] = file.nextLine();

++count;

}

return result;

}

b) Write the method that runs the draft. This method will take the ArrayList<String> of players, the String[] of teams (in order of their pick), and the number of rounds and report which player is assigned to which team. The signature of the method is below. You may assume that you do not run out of players since there is always an excess of people who wish to play Foozball professionally for a living.

public static void assignPlayersToTeams(ArrayList<String> players, String[] teams, int rounds,

Scanner keyboard)

{

for(int round = 0; round < rounds; ++rounds)

{

for (int teamIndex = 0; teamIndex<teams.length; ++teamIndex)

{

System.out.println(“The “ + teams[teamIndex] + “ have the next pick”);

String name = keyboard.nextLine();

if (players.contains(name))

{

System.out.println(“The “ + teams[teamIndex] + “ have chosen “ + name);

players.remove(name);

}

else

{

System.out.println(“There is no “ + name + “ in the draft.”);

}

} // end for

} // end for

} // end method

c) Write the ***main program*** that uses the methods above to implement the full program.

The players’ names are stored in a file called “Players.txt”.

The teams’ names are stored in a file called “Teams.txt”.

This draft will have eight rounds.

The signatures of the other methods are below for easy reference. **You may assume that the readFileIntoArrayList method has been implemented already and do not have to implement it here.**

void assignPlayersToTeams(ArrayList<String> players, String[] teams, int rounds, Scanner keyboard)

String[] readFileIntoArray(String fileName)

ArrayList<String> readFileIntoArrayList(String fileName)

public static void main(String[] args)

{

Scanner keyboard = new Scanner (System.in);

String[] teams = readFileIntoArray(“Teams.txt”);

ArrayList<String> players = readFileIntoArrayList(“Players.txt”);

assignPlayersToTeams(players, teams, 8, keyboard);

}