Final Examination

CS 1323 Section 1, 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. You will not have time to complete the test if you 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.

**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), 10 points for b), 4 points for c))

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

i) The number of votes a candidate for political office receives in an election.

int

ii) Whether or not a given candidate won an election.

boolean

iii) The name of a political candidate.

String

b) Show how the expression in the larger table below will be evaluated, one operation on a line. Show promotions from int to double on their own line.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 2.1 | - | 39 | / | 5 | % | 4 | \* | 10.1 |
|  |  |  | 7 |  |  |  |  |  |
|  |  |  |  |  | 3 |  |  |  |
|  |  |  |  |  |  |  | 30.3 |  |
|  | -28.2 |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |

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

double gallons = 9.2;

if (gallons < 10.0)

{

gallons = gallons + 20.0;

}

if (gallons > 10.0)

{

gallons = gallons + 10.0;

}

else if (gallons > 20.0)

{

gallons = gallons – 5.0;

}

**gallons contains: 39.2**

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

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

int[] data = {1, 4, 19, 21, 24, 29, 48, 57, 62};

int target = 33;

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

{

if (data[i] < target)

result = i;

}

**result is: 5**

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{

|  |  |  |
| --- | --- | --- |
|  | **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};

changeEmUp(x, y, z);

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

}

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

a=6;

b = new String(“GH”);

c[0] = 3.0;

return b;

}

}

|  |  |  |
| --- | --- | --- |
|  | **changeEmUp stack frame** | |
|  | **Address** | **Contents** |
| a | 200 | ~~5~~ 6 |
| b | 201 | ~~1000~~ 1003 |
| c | 202 | 1001 |
|  | 203 |  |

x is: 5

y is: “AB”

z is: {3.0, 2.0}

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.**

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

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 each element of the original list repeated the given number of times. So if the original array contained {“a”, “b”, “c”} and times was 3, the result would be {“a”, “a”, “a”, “b”, “b”, “b”, “c”, “c”, “c”}. The method signature and Javadoc is below. Both arrays are perfect size.

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

@param source A perfect sized array.

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

@return A perfect size array that contains each element of the original array repeated the given number of times.

\*/

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

{

String[] result = new String[source.length \* times];

for (int index = 0; index<source.length; ++index)

{

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

{

result[index\*times + count] = source[index]; // could also use an auxillary counter

}

}

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.**

int repeatElements(String[] source, int size, int times, String[] result)

1. Show the method you wrote in a) above being called by a code fragment using the variables declared below. The code below 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 50 elements that contains 5 “a”s followed by 5 “b”s and 5 “c”s. You may declare other variable(s) that you need.

String[] source;

int sourceSize;

String[] destination;

int destinationSize;

source = new String[10];

destination = new String[50];

source[0] = “a”;

source[1] = “b”;

source[2] = “c”;

destinationSize = repeatElements(source, 3, 5, destination);

1. (10 points; 4 points for a), 3 points for b) 3 points for c)) The Deque class is described in UML below. A deque contains a sequence of integer values, similar to an ArrayList. However, unlike an ArrayList, you can only add and remove values at the beginning (called the head) and the end (called the tail). You are also allowed to examine (peek) at the values at the head and tail. When you peek, it does not change the deque. Do not consider how this class would be implemented.

Deque

-data: int[]

-SIZE: int

-head: int

-tail: int

+Deque()

+addHead(value: int): void

+addTail(value: int): void

+peekHead(): int

+peekTail(): int

+removeHead(): int

+removeTail(): int

+isEmpty(): boolean

+toString(): String

a) Suppose we have a Deque object with **reference myDeque** 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 Deque, leaving it empty.

while (!myDeque.isEmpty()) {

myDeque.removeHead();

}

b) Which methods are accessors?

peekHead

peekTail

isEmpty

toString

c) Which methods are mutators?

addHead()

addTail()

removeHead()

removeTail()

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. You may not assume that the player names are in any particularly order. Each of the teams in the league then picks which players they want, one at a time, until all of the players have been chosen. 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:

Number of entries in the file

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 confirm the draft pick, remove the player from the list of players that are still available to be drafted, and move on to the next team. 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 Jacksonville Cougars will choose next. **// They get another turn!**

Aaron Colvin

Aaron Colvin 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 Detroit Bears will choose next.  **// They get another turn too!**

…

a) Write ***one of the two methods below***. Remember that the file is formatted as follows:

Number of entries in file

Name 1

Name 2

etc.

String[] readFileIntoArray(String fileName)

{

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

int size = file.nextInt();

String[] result = new String[size];

int count = 0;

while (count < size)

{

result[count] = file.nextLine();

++count;

}

return result;

}

ArrayList<String> readFileIntoArrayList(String fileName)

{

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

ArrayList<String> result = new ArrayList<String>();

int count = 0;

while(count < size)

{

result.add(file.nextLine());

++count;

}

return result;

}

b) Write the method that runs the draft. This method will take the ArrayList<String> of players and the String array of teams in order and report which player is assigned to which team. The signature of the method is below.

public static void assignPlayersToTeams(ArrayList<String> players, String[] teams, Scanner keyboard)

{

int teamIndex = 0;

while (!players.isEmpty())

{

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);

++teamIndex;

if (team == teams.length) // could use % here too

teamIndex = 0;

}

else

{

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

}

} // end while

} // end method

c) Write the main program that uses the methods above to implement the full program. The signatures of the methods are below for easy reference:

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

String[] readFileIntoArray(String fileName)

ArrayList<String> readFileIntoArrayList(String fileName)

The file with the team names is called “Teams.txt”. The file with the player names is called “Players.txt”.

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, keyboard);

}