Sample Final Examination

CS 1323, Spring 2014

Name

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

Signature

Date

Show your work to receive partial credit. Unless otherwise indicated, each part of a problem is worth the same number of points.

Read code carefully. When writing code, include meaningful variable names and explanatory comments. All variables must be declared before they are used, and initialized if they are not given as initialized.

You do not need to use Javadoc unless you are specifically asked to.

Tables to be filled in may or may not contain the correct number of lines. If you need more space, just write the data below the table. If you have extra space, leave it blank.

If you need more space to complete an exercise, make sure that you indicate at the bottom of the page containing the problem where the remainder of the exercise is located.

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

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

i) The number of seconds that a song plays.

ii) The name of the artist who recorded your favorite song.

iii) Whether a particular album has been purchased or not.

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

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **10** | **%** | **4** | **+** | **2** | **\*** | **23** | **/** | **4** | **+** | **1.0** |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |

c) Trace the execution of the code fragment below by recording each time data was assigned or changed.

int data = 3;

if (data < 7) {

data = 5;

}

else if (data < 9) {

data = 2;

}

if (data == 5)

++data;

|  |
| --- |
| **data** |
|  |
|  |
|  |
|  |

1. (10 points; 4 points for a), 6 points for b)) Trace the execution of the loops below using the tables at the right.

int result = 8;

|  |  |
| --- | --- |
| **sum** | **result** |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

int sum = 0;

while (sum < 15)

{

sum += result;

result = result - 2;

}



int sum = 0;

|  |  |  |
| --- | --- | --- |
| **i** | **j** | **sum** |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

for (int i=1; i<3; ++i)

{

for(int j=i; j<7; j += 2) **// Watch the increment, it is not 1!**

{

sum = sum + i + j;

}

}

3. (10 points) Perform **selection sort** on the array below. Show each swap of data on its own row in the table. Show only the data that is changed.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **5** | **3** | **1** | **7** | **9** | **2** | **6** | **8** | **4** |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |

1. (15 points) Implement the method with the given signature from the instructions in the Javadoc.

/\*\* Concatenate first and last names held in two arrays and create a new array with complete names (first, followed by a space, followed by last). The first names are in array first. The last names are in array last. If these arrays are not the same length, the length of the returned array should match that of the shorter array.

@param first An array of first names.

@param last An array of last names.

@return If the first and last arrays are the same length, an array of the same length that contains the first and last names joined together will be returned. If the first and last arrays are not the same length, the returned array should contain joined names only to the length of the shorter array.

\*/

public static String[] joinNames(String[] first, String [] last)

{

}

1. (15 points; 12 points for a), 3 points for b)) Trace the execution of the program below using memory diagrams.

**public** **class** Mystery {

**private** **static** **int**[] mystery(**int** count, **int**[] initial){

**int**[] result = **new** **int** [count];

int fill = -1;

**for**(**int** i=0; i<count; ++i){

**if** (count < initial.length)

result[i] = initial[i];

**else**

result[i] = fill;

}

**return** result;

}

**public** **static** **void** main(String[] args) {

**int**[] data = new int[3];

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

data[i] = 2\*i+1;

**int** size = 5;

data = *mystery*(size, data);

}

}

stack frame for main

heap

stack frame for mystery

data

size

count

initial

result

fill

* 1. Based on your memory diagram, what does the array data in the main program contain at the end of execution of the program?

1. (10 points; 2 points for each part) You have been asked to design a program for the myTunes store at Banana, Inc. This store sells music to customers over the internet. Customers purchase songs and store them in their playlist. **The playlist consists of an array of the unique ID numbers (int values) for the songs that have been purchased.**

Find a signature for each method below. Include the method name, parameters and return type. All of these methods will be public and static. **Do not implement any methods.**

The playlist will be declared and constructed in the main program. The playlist’s size is stored separately, as shown below:

final int SIZE = 1000;

int[] playlist = new int[SIZE]; // initially containing all zeros, will change as elements are added

int size = 0; // initially, this will be changed as elements are added

1. Read one individual’s playlist in from a file with a given file name.
2. Sort a playlist by ID number.
3. Search a playlist for a given ID number.
4. Add an ID number to a playlist, if the number is not already in the playlist. If the ID number is in the playlist, do nothing.
5. Remove an ID number from a playlist, if the ID number is in the playlist.
6. (20 points; 8 points for a) and c), 4 points for b))Banana Inc. would now like you to implement some software for their myTunes website. The design for the Song, and SongCatalog classes are shown below.

Song

-name: String

-genre: String

-artist: String

-distributor: String

-price: double

-id: int

+Song(name: String, genre: String, artist: String,

distributor: String, price: double, id: int)

+getName(): String

+getGenre(): String

+getArtist(): String

+getDistributor(): String

+getPrice(): double

+getID(): int

+setPrice(newPrice: double): void

+equals(otherSong: Song): boolean

+toString(): String

SongCatalog

-inventory: ArrayList<Song>

+SongCatalog()

+readInventoryFromFile(filename: String): void

+removeFromInventory(target: Song): boolean

+addToInventory(source: Song): boolean

+printInventory(): void

+changePriceOfAll(increment: double): void

+findSongsByGenre(genre: String): ArrayList<Song>

+writeInventoryToFile(filename: String) : void

a) Write a class that includes a main method that uses the Song and SongCatalog classes to read the inventory from a file named “Songs 5-8-14.txt”, increase the price of each song in the inventory by $.10, and write the inventory out to the same file. You may use methods in the SongCatalog class without implementing them. If you do not remember how to read from a file, read data from the keyboard without input prompts.

b) Implement the constructor in the SongCatalog class.

c) Implement the findSongsByGenre method in the SongCatalog class.

public ArrayList<Song> findSongsByGenre(String genre)

{

}