Heart Transplant – 80 course points

This assignment consists of creating an abstract data type called HeartTransplant, where you will match available donor hearts to recipients waiting for transplant.

Refer to our Programming Assignments FAQ for instructions on how to install Visual Studio Code, how to use the command line and how to submit your assignments.

Programming

We provide this ZIP FILE containing `HeartTransplant.java`. Update and submit the file on Autolab.

Observe the following rules:

DO NOT use System.exit().
DO NOT add the project or package statements.
DO NOT change the class name.
DO NOT change the headers of ANY of the given methods.
DO NOT add any new class fields.
ONLY display the result as specified by the example for each problem.
You may USE any of the libraries provided in the zip file.

Overview

A heart transplant is a surgical procedure that replaces the person's heart with a donor heart. A person may require a heart transplant for several reasons including congenital, arterial and muscle diseases or for unforeseen reasons such as accidents or viral infections.

The donor heart is matched to the recipient by blood type. Additional variables are also used to decide which recipient receives a heart if there are not enough hearts available for all recipients waiting for a transplant. The variables included in the decision may include the recipient state of health, cause of the heart condition and the urgency of the transplant.

The primary goal of this project is for you to write a HeartTransplant class that can be run by the main in the HeartTransplantDriver.

Files Provided

- **HeartTransplantDriver.java**
  - This is the program you should compile and run.
  - It builds a HeartTransplant object and calls instance methods on it.
  - No need to make changes to it, but you are welcome to change it if you want. It is for testing purposes only.
- **SurvivabilityByAge.java**
  - Stores information about the survivability of patients, by age, after *n* years post heart transplant surgery.
- **SurvivabilityByCause.java**
  - Stores information about the survivability of patients, by heart condition cause, after *n* years post heart transplant surgery.
- **Patient.java**
  - Stores patient information.
  - Contain constants explaining the meaning of each code (health condition code, urgency code, ethnicity code…)
- **Data.txt**
  - Contains patient information, as well as survivability by age and cause rates.
  - This file should be piped when executing HeartTransplantDriver
  - Example:
    - **Compile: javac** HeartTransplantDriver.java
    - **Execute: java** HeartTransplantDriver < data.txt

- **HeartTransplant.java**
  - This is the file you will write your code on.
  - Provided as an empty file, complete it based on documentation.

Understanding the Driver:

1. The HeartTransplantDriver will create an HeartTransplant object using its default constructor.
2. The driver will then initialize the three instance variables in the object using the following instance methods:
   1. **readPatients**(int numberOfLines)
   2. **readSurvivabilityByAge**(int numberOfLines)

    2. **readSurvivabilityByAge**(int numberOfLines)
    3. **readSurvivabilityByCause**(int numberOfLines)
3. Finally, the driver will test the three instance methods on the object:
    1. **getPatientsWithAgeAbove**(int age)
    2. **getPatientsByUrgency(** int urgency)
    3. **getPatientForTransplant**()

How to use the SurvivabilityByAge, SurvivabilityByCause, and Patient classes:

SurvivabilityByAge

- **Constructor: default**
- **addData**(int age, int year, double rate)
    - add the survivability rate relating to age and number of years after the transplant to the object can be accessed by using getData() method in the future.
- **getRate(int age, int year)**
    - Returns rate of survivability with given age and years post-transplant.
        - If not found, returns -1
- **getDataSize**()
    - Returns an integer representing how much data is currently stored in the object
- **printAllData**()
    - prints out all data
- **toString**(int a, int y)
    - Returns the string representation of the survivability at a age after y years.
    - returns "NE" if not found

SurvivabilityByCause

- **Constructor: default**
- **addData**(int cause, int year, double rate)
    - add the survivability rate relating to cause and number of years after the transplant to the object can be accessed by using getData() method in the future.
- **getRate(int cause, int year)**
    - Returns rate of survivability with given cause and years post-transplant.
    - If not found, returns -1
- **getDataSize**()
    - Returns an integer representing how much data is currently stored in the object
- **printAllData**()
    - prints out all data
- **toString**(int cause, int year)
    - Returns the string representation of the survivability with cause after y years.
    - returns "NE" if not found

    ▪ returns "NE" if not found

Patient **(Please read the java file)**

- Stores 19 **public constants** describing the meaning of all the codes from the input file Data.txt; details in Patients.java
- **Instance variables** (all private, can be only accessed by getter methods):
    - int id;
        - unique identification of the patient
    - int ethnicity;
        - An integer between [10, 12] representing the ethnicity of the patient
    - int gender;
        - An integer between [13, 15] representing the gender of the patient
    - int age;
        - An integer representing the age of the patient
    - int cause;
        - An integer between [0, 4] representing the cause of the condition
    - int stateOfHealth
        - An integer between [5, 7] representing the current state of health
    - int urgency;
        - An integer between [8, 9] represents the patient's urgency, a higher integer meaning more urgent.
    - boolean needHeart;
        - boolean representing if the patient is still waiting for a heart fo transplant; if true, the patient is still waiting for a heart transplant; if false, the patient no longer needs a heart, the patient already had surgery.
    - **Constructor**: Patient (int id, int ethnicity, int gender, int age, int cause, int urgency, int stateOfHealth)
        - needHeart is default to true, other values are initialized to the data passed in
- Instance Methods
    - int getAge()
        - Returns the Patient's age
    - int getEthnicity()
        - Returns the Patient's ethinicity
    - int getGender()
        - Returns the Patient's gender
    - int getCause()
        - Returns the Patient's cause for the heart condition
    - int getUrgency()
        - Returns the Patient's urgency for the transplant
    - int getStateOfHealth()
        - Returns the Patient's state of health
    - boolean getNeedHeart()
        - Returns boolean for if the Patient is still waiting for a heart

for a heart

- Void setNeedHeart(boolean needHeart)
    - Set needHeart's status
- boolean equals (Object other)
    - returns true if the current object stores the same value as other
- String toString()
    - Returns the string representation of the Patient

Detailed description for HeartTransplant.java:

*****the name of the methods/instance variables MUST match with the description*****

(Side note: this might seem like a lot of work, but all the methods are really short, and some methods even share the same logic, meaning the general structure for one method can be the same for multiple methods, so DON'T PANIC, you have got this)

Instance variables (more details in their respective java files)

- **patients**: an array of Patient objects
- **survivabilityByAge**: a SurvivabilityByAge object
- **survivabilityByCause**: a SurvivabilityByCause object

Constructors:

- **HeartTransplant**()
    - Default constructor.
    - Initializes patients to null.
    - Initializes survivabilityByAge to a SurvivabilityByAge object using each class default constructor.
    - Initializes survivabilityByCause to a SurvivabilityByAge object using each class default constructor.
- **HeartTransplant** (SurvivabilityByAge sba, SurvivabilityByCause sbc)
    - Two argument constructor.
    - Overloaded constructors.
    - Initializes patients to null.
    - Initializes survivabilityByAge to sba.
    - Initializes survivabilityByCause to sbc.

Instance Methods:

- **getPatients** ()
    - Description:
        - returns the instance variables – patients
    - Return type:
        - array of Patient objects
- **getSurvivabilityByAge()**
    - Description:

        - returns the instance variables – survivabilityByAge
    - Return type:
        - a SurvivabilityByAge object
- **getSurvivabilityByCause** ()
    - Description:
        - returns the instance variables – survivabilityByCause
    - Return type:
        - a SurvivabilityByCause object
- **readPatients** (int numberOfLines)
    - Description:
        - Assign instance variable patients with an array of patient objects with the size of numberOfLines. The information needed to create the patient objects are read through the command line. (by design, user should pipe data from a text file into command line when executing the program)
    - Return type:
        - An integer representing how many lines were read from the command line.
    - Data format:
        - Each line refers to one Patient; each line has the format of (all are integers): ID Ethnicity Gender Age Cause Urgency StateOfHealth
- **readSurvivabilityByAge** (int numberOfLines)
    - Description:
        - Assign instance variable **survivabilityByAge** with a SurvivabilityByAge object
        - Then make multiple calls on survivabilityByAge.addData(int age, int year, double rate); to add data to survivabilityByAge
    - Return type:
        - An integer representing how many lines were read from the command line
    - Data format:
        - Each line refers to one set of data, each line has the format of (age and yearsPostTransplant are integers, rate will be in double): age yearsPostTransplant rate.
- **readSurvivabilityByCause** (int numberOfLines)
    - Description:
        - Assign instance variable **survivabilityByCause** with a SurvivabilityByCause object
        - Then make multiple calls on survivabilityByCause.addData(int cause, int year, double rate); to add data to survivabilityByCause
        - The meaning of the integer cause can be found in Patient.java
    - Return type:
        - An integer representing how many lines were read from the command line

    - Data format:
        - Each line refers to one set of data; each line has the format of (cause and yearsPostTransplant are integers, the rate will be in double): cause yearsPostTransplant rate cause yearsPostTransplant rate
- **getPatientsWithAgeAbove** (int maxAge)
    - Description:
        - Find all patient objects with age **greater or equal** to **maxAge** in the instance variable patients, and return them as an array with that exact size.
        - For example, assume **patients** currently stores 10 people with the age of {1,2,3,4,5,6,7,8,9,10}, if **getPatientsWithAgeAbove(5)** is called, then it should returns a size 6 array with the 6 patient objects with age greater or equal to 5.
    - Return type:
        - Array of Patient objects
- **getPatientsByUrgency** (int urgency)
    - Description:
        - Find all patient objects with the targeted urgency in the instance variable patients, and return them as an array with that exact size.
        - For example, assume **patients** currently stores 10 people with the age of {1,2,3,4,5,6,7,8,9,10}, if **getPatientsWithAgeAbove(5)** is called, then it should returns a size 6 array with the 6 patient objects with age greater or equal to 5.
    - Return type:
        - Array of Patient objects
- **getPatientForTransplant** ()
    - Description:
        - Among all patients who **need a heart** and with the **highest urgency**, find the patient with the **highest survivability rate.** Then the target patient‘s **needHeart** should be marked as false and return the patient object.
        - This **survivability rate** is calculated by finding the average of SurvivabilityByAge and SurvivabilityByCause.
        - You can get the survivability by calling on non-static methods getRate(int age, int year) and getRate(int cause, int year) in SurvivabilityByAge and SurvivabilityByCause
        - Because the method marks the target patient’s needHeart to false, repetitive calls on this method n times will give the top n patients with the highest survivability rate.
    - Return type:
        - A Patient object

Data File and Execution

We provide a data file with *real data* that is the input for the program. The data file contains information on 21 people of the same blood type, 24 survivability by age rates, and 15 survivability by heart condition cause rates. More on the data file format below.

Execute the program as follows:

```
java HeartTransplantDriver < data.txt
```

The file data.txt is redirected as the input for the program HeartTransplant and can be read using the StdIn library functions StdIn.readInt() and StdIn.readDouble().

HeartTransplant

Following are the methods to be completed in HeartTransplant.java:

```
public HeartTransplant()
```

Initializes all instance variables to null.

```
public int addPerson(Person p, int arrayIndex)
```

Inserts the parameter Person p into the instance variable array listOfPatients. Returns the integer 0 if it successfully inserts p into the array, and -1 if there is not enough space to insert p into array.

```
public int readPersonsFromFile(int numberOfLines)
```

Allocates the listOfPatients array with numberOfLines length, then reads numberOfLines persons from the data file (each line refers to one Person). For each person read from the data file (a)instantiates a Person object, and (b) inserts the Person object into the listOfPatients array. The method returns the number of patients read from file.
File Format: ID, Ethinicity, Gender, Age, Cause, Urgency, State of health.

```
public int readSurvivabilityRateByAgeFromFile (int numberOfLines)
```

Allocates the survivabilityByAge array with numberOfLines length, then reads numberOfLines survivability by age rates from the data file (each line refers to one survivability rate by age). For each rate read from the data file (a) instantiates a SurvivabilityByAge object, and (b) inserts the object into the survivabilityByAge array. The method returns the number of survivabilities rates read from file.

```
public int readSurvivabilityRateByCauseFromFile (int numberOfLines)
```

`public int readSurvivabilityRateByCauseFromFile (int numberOfLines)`

Allocates the survivabilityByCause array with numberOfLines length, then reads numberOfLines survivability by cause rates from the data file (each line refers to one survivability rate by cause). For each rate read from the data file (a) instantiates a SurvivabilityByCause object, and (b) inserts the object into the survivabilityByCause array. The method returns the number of survivabilities rates read from file.

`public Person[] getPatientsWithAgeAbove(int age)`

Returns a Person array with every Person that has age above the parameter age from the listOfPatients array. The return array has to be completely full with no empty spots, that is the array size should be equal to the number of persons with age above the parameter age. Return null if there is no Person with age above the parameter age.

`public Person[] getPatientsByStateOfHealth(int state)`

Returns a Person array with every Person that has the state of health equal to the parameter state from the listOfPatients array. The return array has to be completely full with no empty spots, that is the array size should be equal to the number of persons with the state of health equal to the parameter state. Returns null if there is no Person with the state of health equal to the parameter state.

`public Person[] getPatientsByHeartConditionCause(int cause)`

Returns a Person array with every person that has the heart condition cause equal to the parameter cause from the listOfPatients array. The return array has to be completely full with no empty spots, that is the array size should be equal to the number of persons with the heart condition cause equal to the parameter cause. Return null if there is no Person with the heart condition cause equal to the parameter cause.

`public Person[] match(int numberOfHearts)`

Assume there are numberOfHearts available for transplantation surgery. Also assume that the hearts are of the same blood type as the persons on the listOfPatients. This method finds a set of persons to be the recepients of these hearts. The method returns a Person array from the listOfPatients array that have the highest potential for survivability after the transplant. The array size is numberOfHearts. If numberOfHeartsAvailable is greater than listOfPatients array size all Persons will receive a transplant. If numberOfHeartsAvailable is smaller than listOfPatients array size find the set of people with the highest potential for survivability. There is no correct solution, you may come up with any set of persons from the listOfPatients array.

the listOfPatients array.
For you to compare how effective your matching function is we will provide, 5 days before the due date, the data file with the condition of each patient after 3 years of the transplant. Data file where the last column has the condition of each patient 3 years post transplant (`Person.java` with condition code).

Data File Format

The data file is divided in three sections. The first section is the *person* section, the second the *rates of survivability by age* section, and the third is the *rates of survivability by heart condition cause* section.

First section: **person**

The first line of this section has an integer that refers to the number of persons in the file. The file has one person per line in the following format:
`PersonID Ethinicity Gender Age Cause Urgency StateOfHealth`

The example below has 3 persons, the first person's ID 4101, Ethnicity 10, Gender 13, Age 75, Cause 3, Urgency 8, and StateOfHealth 7. You can see the meaning of each code in the `Person.java` file.

Second section: **survivability by age**

The first line of this section has an integer that refers to the number of survivability by age rates in the file. The file has one rate per line in the following format:
`Age YearsPostTransplant Rate`

The example below has 8 rates, the first rate specifies that people with age LESS than 1 year old, 5 years post transplant have a survivability rate of 83.6%.

Third section: **survivability by heart condition cause**

The first line of this section has an integer that refers to the number of survivability by cause rates in the file. The file has one rate per line in the following format:
`Cause YearsPostTransplant Rate`

The example below has 5 rates, the first rate specifies that people with heart condition cause 4 (heart muscle disease), 1 year post transplant have a survivability rate of 89.4%.You can see the meaning of each code in the `Person.java` file.

```
4101  10  13  75  3  8  7
4102  11  14  78  4  9  5
4103  12  14  40  3  8  5
8
 1  5 83.6
 6  5 87.4
 11 5 86.8
 18 5 90
 35 5 86.9
 50 5 88.8
 65 5 87.8
120 5 84.3
5
4 1 89.4
1 1 82.1
3 1 87.1
0 1 88.2
2 1 88.2
```

Check out this video explaining the assignment.

Before submission

1. *Collaboration policy*. Read our collaboration policy here.
2. *Update @author.* Update the @author tag of the files with your name, email and netid.
3. *Submitting the assignment*. Submit *HeartTransplant.java* via the web submission system called Autolab. To do this, click the *Assignments* link from the course website; click the *Submit* link for that assignment.

Getting help

If anything is unclear, don't hesitate to drop by office hours or post a question on Piazza. Find instructors office hours by clicking the *Staff* link from the course website.
In addition to office hours we have the CAVE (Collaborative Academic Versatile Environment), a community space staffed with lab assistants which are undergraduate students further along the CS major to answer questions.

Assignment by Haolin (Daniel) Lin and Ana Paula Centeno