CS. 67125

Object Oriented Programming - Exercise 0 - Knesset

In this exercise you will practice basic Java and usage of IntelliJ IDEA as shown in the Tirgul. We remind you that you are required to work alone, submit on time, and use the forum for questions.

Basic Info

Due date: 15.3.17

Work is done individually:
When writing the code, testing it and submitting your solution, you must be work individually. You cannot look at the code other students wrote or ask them to describe it to you, and you most certainly cannot copy other people's code. You can, however, and are encouraged to, use code examples from the lectures, tirguls and the coding style document as a starting point.

Responsible Tzar: Naomi.

Requirements:

Your task in this exercise is to write three Java classes: Law(חוק), KnessetMember (חבר כנסת) and Assembly (מליאה), where each is implemented in a separate Java file; Law.java, KnessetMember.java and Assembly.java.

The three classes composing the program, including the two classes you need to implement from scratch, are well documented in the end of this file and in the attached Law.java file on moodle. The API defines for each class, the list of public fields and methods it contains, and for each such method - details about the return type and the parameters it receives (this is called the method's signature). You will have to implement the given API. For the matter of this exercise, the API simply contains all the method signatures that the different classes have to contain. You may add methods and fields not mentioned in the API, but they should only be used to implement the API, not change it. Think carefully before deciding which additional methods your program needs.

You are provided with a class skeleton, or a template, for the Law class, and an implementation of its method stringRepresentation(), and are required to implement the rest of its methods. You are also required to create the class files of the other two classes and implement all their required methods from scratch.

Classes Composing This Exercise

1. Law

Will represent a new law that a single KnessetMember is initiating toward the next assembly. He should get as much support and agreement for his law, hence he wants as many KnessetMembers as possible to join his law.
The Law java class file is provided and there you can see the full API.

2. KnessetMember

A KnessetMember can join as many laws as he wishes. He decides whether to join a law or not based on two factors:

a. A Survey Threshold (int) - for every law a survey is conducted to see how much support it gets from the people. An attentive KnessetMember will only join laws that get high support in the survey. Every KnessetMember decides for himself how attentive he wants to be. A KnessetMember will never support a law that hasn't passed his threshold of public support, no matter what the law is.
b. Whether to support a law or not is also affected by the preferences of the KnessetMembers: each law has three characteristics that compose its value - its social value, its economic value and its political value. Different KnessetMembers ascribe different levels of importance to these three characteristics; this is reflected by the individual set of weights each KnessetMember assigns them (weights should be positive numbers between 0 to 1, and should sum to 1). The weighted sum of the above characteristics reflects the total value a KnessetMember assigns to some law, and a KnessetMember will only support a law if its weighted sum is above 5, **and it passed the survey.**

3. Assembly

Assembly represents a single Knesset meeting (where laws are discussed). In each assembly there can be multiple laws that are presented to a vote, for each law the assembly holds a survey of public support of the law. A KnessetMember can participate in an assembly, a law can be added to discussion in this assembly. In addition, the assembly can suggest a law to a KnessetMember. See full API in the end of this document.

KnessetMember AND Assembly API's ARE DESCRIBED IN THE END OF THIS DOCUMENT AND MUST BE IMPLEMENTED.
Law API IS DESCRIBED IN THE ATTACHED FILE THAT CAN BE FOUND ON MOODLE.

Java arrays

In your implementation of the Assembly class you will have to use some representation for the lists of laws and participating KnessetMembers, and the basic candidates in Java are arrays. We will learn more about Java arrays further along in the course, but for now it is enough to know that they are somewhat similar to the lists you know from Python. The basic principles needed for using arrays in this exercise are:

- Like all variables in Java, arrays are typed, so you need to declare the type of primitives or objects you want to hold in the array. The type of an int array is int[], that of a boolean array is boolean[], and an array of Dog objects is of the type Dog[].
- In Java you declare the size of the array on initialization, and it cannot change afterwards. You can declare an array of 7 integers with int[] myIntArray = new int[7];, and an array of 12 Dog objects with Dog[] dogArray = new Dog[12];. The size can also be determined using a variable, as in:

  int x = 43;
  Dog[] dogArray = new Dog[x];.
- Access to array elements is done with square brackets holding the index of the desired element, when the first element has index 0 and the ith element has index i − 1 (so the last element in an array of n elements has index n − 1). You can access the first element of your int array with int firstElement = myIntArray[0], and the 5th element of your Dog objects array with Dog fifthDog = dogArray[4];. Also, if the Dog class has a bark() method, you can 'make' the fifth dog bark by writing dogArray[4].bark(); Also note here you cannot use negative indices (as opposed to Python).
- Assignment to array elements is done using the same syntax, so you can assign 82 as the first element of your int array with myIntArray[0] = 82;, and 'adopt' a new dog as the 7th element of your Dog objects array with dogArray[6] = buster; (assuming Dog buster =new Dog("Buster"); was defined earlier, of course). You can't remove primitives from an array (so it is common to assign 0 instead of the 'deleted' value in int arrays, for example, although the right value for a deleted cell is context-dependent), but you can remove objects from an object array by assigning null to the corresponding cell (which is a little like pointing that cell at 'nothing') with dogArray[10] = null; (so null is to Java what None is to Python).

Documentation and Testing Requirements

- Since the program you are required to write as part of this exercise has no main method, you should write a separate test class, which will contain a main method; in this method you will be able to use elements of your program (in this case, Laws, KnessetMembers and Assemblies) and check for their correct behavior. You can then run your test class as detailed above (that is if your test class with the main method is named Test, have your Test.java file in the same folder as your other classes, execute javac for each java file and then execute java Test). If you do write such additional classes, do not submit their source files (.java) or class files (.class)!

- Please make sure you write a self-explanatory code, use documentation if you are not sure the checker will understand the code, but know that well-written code doesn't need to be documented, and is understandable from reading the code itself.

Submission

- Your code is required to compile and run correctly on the Linux environment that is currently installed on CS lab computers. You can write your code using any tool and in any OS you wish to use, as long as you make sure that it compiles properly and returns the correct output when it runs on CS computers, and upon submission to the course Moodle site - when you submit your code, it will be compiled and executed automatically, and you will receive a report detailing how it went. Nevertheless, we strongly advise that you use the CS computers to compile and execute your code in the CS lab Linux environment (see how in the following subsections).
- You should submit only the following files:
  Law.java, KnessetMember.java, Assembly.java and README.
- Make sure you do not submit any .class files by mistake.
- Create a JAR file named ex0.jar containing only these files by invoking the shell command "jar cvf ex0.jar Law.java KnessetMember.java Assembly.java README" with the shell's current working directory set to the folder containing the above files. Your JAR file should not contain any other files.
- Make sure that the JAR file you submit passes the presubmit script by carefully reading the response file generated by your submission. Exercises failing this presubmit script will get an automatic 0!

Good Luck!

KnessetMember API:

```
static double knessetMembersEnthusiasmThreshold = 5;

/**
* Creates a new KnessetMember with the given characteristics.
* @param knessetMemberFirstName The first name of the KnessetMember.
* @param knessetMemberLastName The last name of the KnessetMember.
* @param socialTendency The weight the KnessetMember assigns to the social aspects of laws.
* @param economyTendency The weight the KnessetMember assigns to the economy aspects of laws.
* @param politicalTendency The weight the KnessetMember assigns to the political aspects of laws.
* @param knessetMemberSurveyThreshold The minimal public support a law must have for this KnessetMember
to join it.
*/
KnessetMember(String knessetMemberFirstName, String knessetMemberLastName,double socialTendency, double
economyTendency,
  double politicalTendency, int knessetMemberSurveyThreshold)

/**
* Returns a string representation of the KnessetMember, which is a sequence of its first name, last name and title
* separated by a single white space. For example, if the KnessetMember's first name is "Yehudah" and his last name
* is "Glick", this method will return the String "Knesset Member Yehudah Glick".
* @return the String representation of this KnessetMember.
*/
String stringRepresentation()

/**
* Returns the interest value this KnessetMember assigns to the given Law.
* @param law The law to assess.
* @param surveyResult The result of a survey made for this law
* @return the interest value this KnessetMember assigns to the given law. 0 if survey result does not pass threshold.
*/
double getLawScore(Law law, int surveyResult)

/**
* Returns true of this KnessetMember will join the given law (law score is bigger than
knessetMembersEnthusiasmThreshold), false otherwise.
* @param law The law to asses.
* @return true of this KnessetMember will join the given law, false otherwise.
*/
boolean willJoinLaw(Law law, int surveyResult)
```

Assembly API:

```
/**
* Creates a new assembly with the given parameters.
* @param maxLawCapacity The maximal number of laws this assembly can hold.
* @param maxSupportedLawsPerKnessetMember The maximal number of laws this assembly allows a single
KnessetMember to support at the
* same time.
*/
Assembly(int maxLawCapacity, int maxSupportedLawsPerKnessetMember)

/**
* Adds the given law to this assembly, if there is place available, and it isn't already in the assembly.
* @param law The law to add to this library.
* @param surveyResult The survey result of the law.
* @return a non-negative id number for the law if there was a spot and the law was successfully
* added, or if the law was already in the assembly; a negative number otherwise.
*/
int addLawToAssembly(Law law, int surveyResult)

/**
* updates the survey result of lawId with a new value
* @param law law to be updated
* @param newSurveyValue new survey value.
*/
void updateSurveyResultOfLaw(Law law, int newSurveyValue)

/**
* Returns true if the given number is an id of some law in the assembly, false otherwise.
* @param lawId The id to check.
* @return true if the given number is an id of some law in the assembly, false otherwise.
*/
boolean isLawIDValid(int lawId)

/**
* Returns the non-negative id number of the given law if he is discussed by this assembly, -1 otherwise.
* @param law The law for which to find the id number.
* @return a non-negative id number of the given law if he is discussed by this assembly, -1 otherwise.
*/
int getLawId(Law law)

/**
* Registers the given KnessetMember to this assembly, if there is a spot available.
* @param KnessetMember The KnessetMember to register to this assembly.
* @return a non-negative id number for the KnessetMember if there was a spot and the patron was successfully
* registered, a negative number otherwise.
*/
int registerKnessetMember(KnessetMember KnessetMember)

/**
* Returns true if the given number is an id of a KnessetMember in the assembly, false otherwise.
* @param KnessetMemberId The id to check.
* @return  true if the given number is an id of a KnessetMember in the assembly, false otherwise.
*/
boolean isKnessetMemberIdValid(int KnessetMemberId)

/**
* Returns the non-negative id number of the given KnessetMember if he is registered to this assembly, -1 otherwise.
* @param KnessetMember The KnessetMember for which to find the id number.
* @return a non-negative id number of the given KnessetMember if he is registered to this assembly, -1 otherwise.
*/
int getKnessetMemberId(KnessetMember KnessetMember)

/**
* adds KnessetMember to law, if the KnessetMember will support the law according to the survey results.
* @param lawId The id number of the law to borrow.
* @param KnessetMemberId The id number of the KnessetMember that will support the law.
* @param surveyResult The survey result of the law to support.
* @return true if the KnessetMember was added successfully, false otherwise.
*/
boolean supportLaw(int lawId, int KnessetMemberId, int surveyResult)



/**
* Suggest to the KnessetMember with the given id a law which suits him the best (maximum score of the laws in the
assembly).
* @param KnessetMemberId The id number of the KnessetMember to suggest the law to.
* @return The best law to match the KnessetMember preferences. Null if there aren't any (case all laws get a zero
score).
* available.
*/
Law suggestLawToKnessetMember(int KnessetMemberId)
```