Verificare si validare Lab 1

Reviewer name: Ursuta Claudiu gr. 235.
Review date: 13.03.2018

1. **Requirements Phase Defects Checklist**

R01 - Requirements are incomplete

It is specified that a professor wants this app. Will he be the single user or will there be others ?

At point “iii.” It is not completely defined what kind of statistics are desired and how they should be displayed.

At point “ii”. What happens if there aren’t 5 different domains with at least a question ? The app should probably show an appropriate message. Something like “Could not create the test. Not enough domains have questions. There should be at least 5.”.

R02 - Requirements are missing.

It is not specified what kind of application is required. Should it be a web, client-server or console.

When should the app’s data be saved ? After adding a new question or when the app closes ?

It is not specified to show the test once it is created. That is a must.

R03 - Requirements are incorrect

“I.” says that every question should have 3 choices for an answer. Maybe the questions should have a flexible number of answers.

“II” says that a test contains 5 questions. Maybe the question number / test should be flexible.

R04 - Initialization of the system state
No remarks

R05 - The functions have not been defined adequately
If all questions are addressed, then there are no remarks here

R06 - The user needs are inadequately stated
It is not clear what the user wants. Refer to R01, R02, R03.

R07 - Comments
All of the questions above should be clarified by the client before going further in the development process.

2. Architectural Design Phase Defects

A01 - Is the overall organisation of the program clear, including good architectural overview ?

The architecture is not clear.
SerialVersionUID is a field to define the version of a particular class while serializing & deserializing. Why is that needed ?

There is a great amount of overhead if the dictionary Statistica class is populated using the add method. The repository should keep the questions as a dictionary in the first place, then the Statistica class should take in an IQuestionRepository interface and take the data.

There should be three answer options to each question. Why does the Intrebare class only poses two possible answers. It should have a list that is limited to 3 items.

This way, if is is required at any moment to have more than 3 possible answers to a question, it is easy to modify it. (And it is highly likely that is will happen.)

Also, why is the variantaCorecta a string ? There is a high chance that some questions will have a variantaCorecta that is not even in the choice list.

The Intrebari repository violates the single responsability principle. There should be different classes for storing/loading and getting the a random question.

Why does the Intrebare class use Validation ? That is not the place where validation should be.
The controller should use a validator interface. The model should not know anything more than the fields it has and providing methods for getting and modifying those fields. Even more, it is not all right that a model can throw an exception.

Why does the validator have validators for each answer choice ? The validator is the same for each of those. (And it has 3 methods for that while the model has now two choices for an answer)

How can the validateDomeniu work ? If it is like in the diagram it means that it compares the “domeniu” to some hardcoded strings. The accepted fields for the questions should be persisted too.

A02 - Is the subsystem and package partitioning and layering logically consistent ?

There is no actual package partitioning on the diagram. The layering is mostly logical. Ther should be packages for: Controller, Repo, Models, Exceptions and Validators.

A03 - Does the architecture account for all the requirements ?

No, the Intrebare class has only two answer choices. The requirements state that there should be 3. That should be a list and it should have a maximum amount of 3 items.
Also, it is not clear what should be called to obtain the statistics. Should Statistica.ToString() be called ? That is counter-intuitive. There should be a method for getting the statistics data formatted.
Also, there is no 5 questions per test limitation. That was a requirement.
There is no save to file. Only load from file.

A05 - Are there classes in a subsystem supporting the services identified for the subsystem ?

There is no division in subsystems, tough it should exist.

A06 - Have classic design patterns been considered where they might be incorporated into the architecture ?

The only pattern applied here is the Repository Pattern.
If the has a graphical interface, the observer pattern should be used, so the GUI knows when there are changes in the Repository.
The Command Pattern might come in handy.
The controller should not be held responsible for creating Stastica instance. Dependency injection should be used there. That would mean creating an interface for Statistica.
There should also be a factory for the controller since an IStatistica should be injected at creation.

A07 - Is the name and description of each class clearly reflecting the played role ?

The naming mixes terms in English and Romanian. That makes it unclear. A decision should be made consistently name them using only one language. Other than that, the naming is good.

A08 - Is the description of each class clearly reflecting the played role ?

There is no description attached to the classed. The models should be documented and there should be interfaces for the repository, the controller and the statistics that should be documented. (At least.)

A09 - Are the role names of aggregation and associations accurately describing the relationship between the related classes ?

The composition relation between Intrebare and Test is wrong. The repository should be the only one owning instances of Intrebare.
The Test class really should not inherit from Intrebare.
It would be better if the repository was created in a factory, not directly by the controller.
The start app should not use the NotAbleToCreateStatisticsException nor the Statistica class. That controller should do that.

A10 - Are the key entity classes and their relationships consistent with the business model (if it exists), domain model (if it exists), requirements.

See A03.

3. **Coding Phase Defects Checklist.**

CO1 - Decision logic is erroneous or inadequate.

Controller - line 35. The if checks that there are more than 2 questions in a test. There should be exactly 5.

Controller - line 46 - There should be 5 questions per test. Not 7.fac

Statistica - getStatistica - The number of questions per domain is not right. The number of all the questions is added for each domain.

StartApp - the switch. The options 1 and 2 are not doing anything.

AppController - createNewTest, line 48. The condition will not allow the creation of any test. The test will not contain the randomly picked question.

C02 - Branching is erroneous.

Refer to C01 - the comment about the switch from StartApp.
Repository - line 100. The finally block has a faulty try and catch branching. The br.close will result in a null pointer exception when the file is not found.

C03 - There are undefined loop terminations

Repository - method loadIntrebariFromFile. If the file is not formatted right, those whiles could go on undefined.

C04 - I/O format errors exist.

The hardcoded persistence file name does not exist.

C05 - Subprogram invocations are violated.

Nope, this is not javascript.

C06 - There are errors in preparing or processing input data.

Refer to C01 - the comment about the switch from StartApp.

C07 - Output processing errors exist.

An exception will be thrown when the user requests the statistics. The file for loading the data is not found and in the finally clause, the buffered reader that will be null is closed. This will result in a null pointer exception.

C08 - Error message processing errors exist.

Refer to C07.

C09 - There is confusion in the use of parameters.

Stastica - intrebariDomenii field. This should have a different name. It is not clear what is was intended for. Maybe domainQuestionNumber.
All the names should be either completely in english or in romanian, not a mix of the two.

C10 - There are errors in loop counters.

Refer to C03.

C11 - Errors are made in writing out variable names.
-

C 12 - Variable type and dimensions are incorrectly declared.
-

Others

The questions are loaded when the user presses 3 to get a statistics.

Repository - line 47. That should not be a TreeSet. It should probably be a map.

Controller - method getStatistica. Only domains are added to the statistics.

The Intrebare model constructor could throw a validation exception. That is really ugly.