Inspectaţi documentele: cerinţe, arhitectura şi codul sursă de mai jos. Prezentați concluziile într-un raport de inspectare.

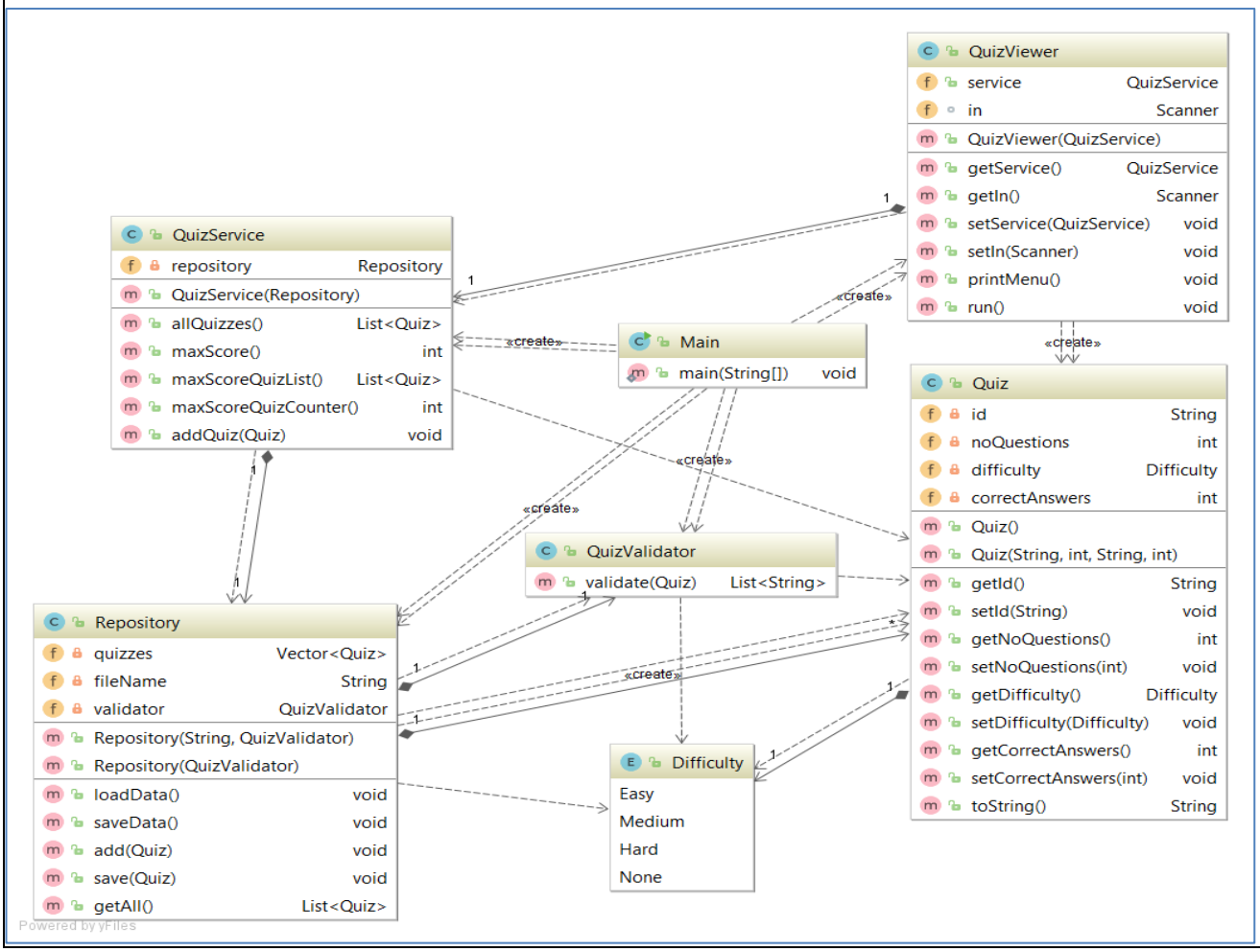
Cerinţe	
01	**Context şi cerinţe.** Albert este student de la Facultatea de Filosofie și se pregăteşte pentru proba scrisă a examenului de licenţă. Are la dispoziţie câteva fişiere care conţin un număr considerabil de teste cu trei niveluri de dificultate şi număr de întrebări variat. Studentul doreşte să îşi monitorizeze procesul de pregătire şi consideră că o aplicaţie l-ar putea ajuta. Pentru fiecare încercare de efectuare a unui test doreşte să ştie numărul de răspunsuri corecte obţinut şi data încercării testului. Prietenul său, George, student la Facultatea de Matematică și Informatică în anul 3, se oferă să îl ajute.
02	George a proiectat şi implementat o aplicaţie Java cu funcţionalităţile de mai jos. A folosit o arhitectură stratificată, iar informaţiile referitoare la teste sunt preluate dintr-un fişier text cu structura: id test, număr de intrebări, dificultatea testului, numărul de răspunsuri corecte obținute.
03	**F01.** Aplicaţia afişează informațiile referitoare la un test efectuat: numărul testului, numărul de intrebări, dificultatea testului, numărul de răspunsuri corecte la fiecare incercare a testului.
04	**F02.** Afişarea numărului de teste distincte efectuate (fără a lua în calcul numărul de încercări pentru fiecare test).
05	**F03.** Clasamentul testelor, descrescător după numărul de răspunsuri corecte raportat la numărul de intrebări din test.
06	**F04.** Afişarea numărului de încercări pentru testele la care s-a obținut cel mai mare număr de răspunsuri corecte al studentului.

Arhitectura

packages:

- `domain [Quiz, Difficulty, QuizValidator];`
- `validator[];`
- `repository [Repository];`
- `service [QuizService];`
- `ui [QuizViewer].`

class: `Main`



Cod sursă

```
01  public class QuizService {
02
03  private Repository repository;
04  public QuizService(Repository repository) {
05     this.repository = repository;
06  }
07  public List<Quiz> allQuizzes() {
08     return repository.getAll();
09  }
10  public int maxScore(){
11   int maxScore=0;
12   List<Quiz> quizList = allQuizzes();
13
14   for (int i = 0; i <= quizList.size(); i++)
15     if (quizList.get(i).getCorrectAnswers() >= maxScore)
16        maxScore = quizList.get(i).getCorrectAnswers();
17
18    return maxScore;
19  }
20  public List<Quiz> maxScoreQuizList() {
21   List<Quiz> quizList = allQuizzes();
22   List<Quiz> maxScoreQuizList = new ArrayList<Quiz>();
23   int maxScore = maxScore();
24
25   for (int i = 0; i <= quizList.size()-1; i++)
26     if (quizList.get(i).getCorrectAnswers() == maxScore)
27        maxScoreQuizList.add(quizList.get(i));
28
29   return maxScoreQuizList;
30  }
31  //input: quizzes - list of quizzes;
32  //output: k - number of quizzes having the highest value for the correct
    answers field,
33  //          if quizzes is empty then k = 0;
    //          if no quiz with correct answers!=0 then k = 0
34  public int maxScoreQuizCounter(){
35   int k, i, p;
36
37   List<Quiz> quizList = allQuizzes();
38
39   i=0;k=1;p=0;
40   while (i<quizList.size()){
41     if
    (quizList.get(i).getCorrectAnswers()>quizList.get(p).getCorrectAnswers()){
42     p=i;
43  }
44   else
45    if (quizList.get(p).getCorrectAnswers()==quizList.get(i).getCorrectAnswers())
46     k++;
47   i++;
48   }
49   return k;
50  }
51  public void addQuiz(Quiz quiz) {
52   repository.add(quiz);
53  }
54  }
```

Requirements Phase Defects Checklist [1]

Nr.	Check Point / Defect Statement	Check Mark (√) the Appropriate Column	
		Yes	N/A
R01	Requirements are incomplete.		
R02	Requirements are missing.		
R03	Requirements are incorrect.		
R04	Initialization of the system state has not been considered.		
R05	The functions have not been defined adequately.		
R06	The user needs are inadequately stated.		
R07	Environment information is inadequate or partially missing.		

Architectural Design Phase Defects Checklist [2]

Nr.	Check Point / Defect Statement	Check Mark (√) the Appropriate Column	
		Yes	N/A
A01	Is the overall organization of the program clear, including good architectural overview?		
A02	Is the subsystem and package partitioning and layering logically consistent?		
A03	Does the architecture account for all of the requirements?		
A04	Are the classes in a subsystem supporting the services identified for the subsystem?		
A05	Is there a coherent error handling strategy provided?		
A06	Have classic design patterns been considered where they might be incorporated into the architecture?		
A07	Is the name and description of each class clearly reflecting the played role ?		
A08	Is the description of each class accurately capturing the responsibilities of the class?		
A09	Are the role names of aggregations and associations accurately describing the relationship between the related classes?		
A10	Are the key entity classes and their relationships consistent with the business model (if it exists), domain model (if it exists), requirements?		

[1] sursa: http://www.softwaretestinggenius.com/requirements-phase-defects-checklist
[2] surse: http://deg.egov.bg/LP/core.base_rup/guidances/checklists/software_architecture_document_D261D8F3.html

Coding Phase Defects Checklist [3]

Nr.	Check Point / Defect Statement	Check Mark (√) the Appropriate Column	
		Yes	N/A
C01	Decision logic is erroneous or inadequate.		
C02	Branching is erroneous.		
C03	There are undefined loop terminations.		
C04	I/O format errors exist.		
C05	Subprogram invocations are violated.		
C06	There are errors in preparing or processing input data.		
C07	Output processing errors exist.		
C08	Error message processing errors exist.		
C09	There is confusion in the use of parameters.		
C10	There are errors in loop counters.		
C11	Errors are made in writing out variable names.		
C12	Variable type and dimensions are incorrectly declared.		

Review Report

Document Title:	
Author Name	
Reviewer Name:	
Review date:	

Crt. No.	Checked Item	Doc. page/line	Comments/ improvements
1	C06		
2	…		
3			
4			
…			
…			
16			

Effort to review document (hours):	

[3] sursa: http://www.softwaretestinggenius.com/program-coding-phase-defects-checklist