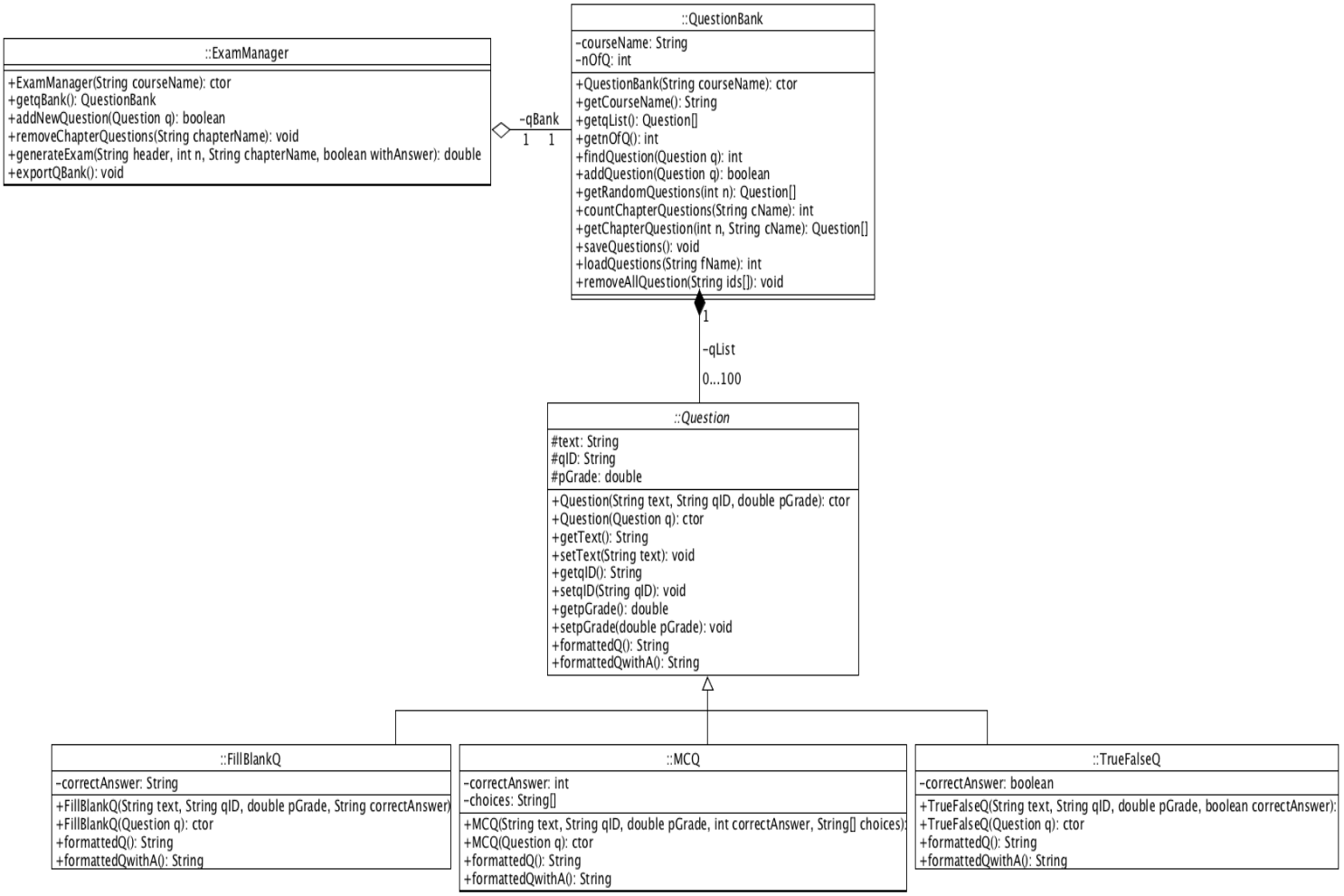
KING SAUD UNIVERSITY COLLEGE OF COMPUTER AND INFORMATION SCIENCES INFORMATION TECHNOLOGY DEPARTMENT		
CSC 113: Java Programming 2	**Project Phase-1**	**Spring 2019**

You are designing and implementing a simple java application to help your teachers in preparing course exams. More than one teacher can use it, by entering course name only. All questions in this application persist beyond the execution of each session, which means after each session you should save questions in object file.

The following UML describes the required classes in general (**check the pdf file**):

1. Class Diagram:



2. Class Description:

Implement the above UML using the following description

Class Question

- String text: question text
- String qID: question identifier, unique for each question start with chapter number followed by "_" followed by question number. For example (ch08_1), means question 1from chapter 8.
- double pGrade: possible grade
- Question (String text, String qID, double pGrade): constructor to initialize question's attributes
- Question (Question q): copy constructor
- String formattedQ():**an abstract** method that return a formatted string of question depends on the type of question
- String formattedQwithA():**an abstract** method that return a formatted string of question along with correct answer depends on the type of question.
- Setters and Getters as needed

Class MCQ

- int correctAnswer; index of correct answer
- String [] choices: List of 4 string represents choices for this question
- String formattedQ():method that return a formatted string of MCQ question using the following format

Choose the correct answer: qID:text 1:choices[0] 2: choices[1] 3: choices[2] 4: choices[3]	Choose the correct answer: ch04_1:Which of the following is true about inheritance in Java? 1:We cannot override private methods 2:We cannot override protected methods 3:We cannot override private methods 4:None

- String formattedQwithA(): return a formatted string of question along with correct answer. Use formattedQ() and "Correct Answer is " correctAnswer

Class FillBlankQ

- String correctAnswer: contain answer
- String formattedQ():method that return a formatted string of Fill Blank question using the following format.

Fill the following blank qID:text	Fill the following blank Ch04_2:We cannot override ---------- methods

- String formattedQwithA():return a formatted string of question along with correct answer Use formattedQ() and "Correct Answer is " correctAnswer

Class TrueFalseQ

- boolean correctAnswer: answer for question
- String formattedQ():method that return a formatted string of True or False question using the following format

True or False qID:text	True or False ch04_1:We cannot override private methods

- String formattedQwithA():return a formatted string of question along with correct answer. Use formattedQ() and "Correct Answer is " correctAnswe

Class QuestionBank

- int noQ: number of questions in qList
- String courseName: name of the course
- Question [] qList: array of question
- QuestionBank(String cname): constructor to initialize attribute
- boolean addQuestion(Question q): this methods adds the received question object in the first empty position in qList. If and only if the question q is not **found** in qList and return true. False otherwise.
- void removeAllQuestion (String ids []): this methods remove **all questions** with qID equals to IDs in the received array **ids.** Use replacement with shifting.
- int findQuestion (Question q): searches the qList array for the question q, returns its location if found, or -1 otherwise.

- Note: to avoid redundant questions, two questions considered equals if they have same qID or same text

- int countChapterQuestions (String cName): this methods return number of questions in qList belongs to received chapter name cName
  - Note: chapter name is included in the question ID
  - Hint: to avoid logical error when comparing two String, compare both String after converting them to lower case
- int loadQuestions(String fname) :This method read questions objects (object by object) from the object file **fname** and then adds them into the **qList** array. The method ends when no more object left for reading. Then returns the number of objects successfully added to the **qList** array.
  - Note: file name should be same as course name, otherwise an appropriate exception – Illegal Argument Exception -should be generated.
- void saveQuestions() :This method write to an object file called (courseName.ser) all the questions in the **qList** array (object by object).
- Question [] getRandomQuestions(int n): This method return an array of **n** different questions randomly selected from **qList**. If the total number of questions in **qList** is less than n, an IllegalArgumentException should be generated.
- Question [] getChapterQuestions(int n,String cName): This method return the first n questions from **qList** that belongs to the received chapter name cName. If number of questions for this chapter is less than n it should return an array containing the available questions only.
  - Note: The size of returned array is equal to the number of questions from specified chapter
  - Hint: to avoid logical error when comparing two String, compare both String after converting them to lower case

Class ExamManager

- QuestionBank qBank: question bank for specific course
- ExamManager (String courseName): constructor to initialize attribute and **fill** question bank **qBank** with question objects from file (courseName.ser) if file exists. If file not found the appropriate message should be printed.
- boolean addNewQuestion (Question q): this method adds the received question q object to **qBank**
- void removeChapterQuestions(String chapterName): this method removes all questions belong to the received chapter name chapterName
  - Hint: you should use removeAllQuestion (String ids[])

- double generateExam (String header, int n, String chapterName, boolean withAnswers):
  This method should write into a text file called (coursenameExam.txt). The received header (which contains the exam information) should appear at the beginning of the text file.
  If the received **chapterName** is an empty string, then **n** questions are randomly selected from qBank and should be written to this text file, otherwise n questions from the specified chapter **chapterName** should be written. If the **withAnswers** is true then the generated file should contain correct answer for each question. This method should return the total number of possible grades for this exam.
- void exportQBank() this method should save all questions in qBank to be used in the next session
  - Hint: Use saveQuestions()

Class TestApp

- main (String args [])
  - Given to you to test your implementation

3. Deliverable and rules

You must deliver:

- Source code submission to LMS. You have to upload all classes in a zipped file. (section#_group#_CSC113Project)
- The submission deadline is **9 March 2019.**

You have to read and follow the following rules:

- **The specification given in the assignment (class and interface names, and method signatures) must not be modified. Any change to the specification results in compilation errors and consequently the mark zero. (Use the same name in UML)**
  - **For testing purpose, you should add the setters/getters methods even if you didn't use them.**
- This assignment is a group assignment. Each group contains **3-4 students only** from the same section.
- The submitted code will be evaluated automatically using Junit Tests and discussion during week 10 Lab time.
- **All submitted code will be automatically checked for similarity, and if plagiarism is confirmed penalties will apply, (-2) in the total marks of the project.**