CC/CS12320
Introduction to Object Oriented Programming
Main individual assignment: The Question Bank Project

Dr. Faisal I Rezwan
far8@aber.ac.uk

1872 PRIFYSGOL ABERYSTWYTH UNIVERSITY

Assessments

Schedule (when ready):
https://teaching.dcs.aber.ac.uk/dates/Year/One

- No exam!

	Details	**%**	**Dates**
1. Mini assignments	5 In-class quizzes Best 4 out of 5 each 10%	40	Quiz 1: 16/02/2024 Quiz 2: 01/03/2024 Quiz 3: 22/03/2024 Quiz 4: 19/04/2024 Quiz 5: 03/05/2024 (Provisional)
2. Main assignment	One relatively large individual assignment	50	11 March – May 08 Provisional
3. Tutorials	Assessed based on your performance during sessions	10	

Problem statement

A local school requires a prototype quiz program to help its students evaluate their understanding and revise for exams.

This prototype must be text based with a textual menu to:

1. enable teachers to build a bank of questions with answers for a specific module.

2. enable students to take a quiz associated with a module, where a selection of questions are presented in a random order.

Problem statement

- Type of users:

Teacher

Student

For this prototype there are no security requirements.

Problem statement

A question bank is associated with just one module, but one module can have many question banks.

CS12320	CS12020	CS11010
• QuestionBank01 • QuestionBank02	• QuestionBank01 • QuestionBank02 • QuestionBank03	• QuestionBank01

Your Task: To develop the question bank project with provided functional and non-functional requirements

Your task: Functional Requirement 1

FR1:

- A teacher can create a new empty question bank and associate it with one module identifier.
- Each question bank has a unique identifier of the form *module-identifier:bank-identifier* where bank identifiers must be unique for a specific module.
- The format of the identifiers (both module and question bank) is up to the teacher, that is, it can be any text such as a number or a word or even a phrase. However, the maximum length of module and bank identifiers should be 7 and 15, respectively.

Example:

module-identifier: CS12320

bank-identifier: QuestionBank01

Question bank unique identifier would be: CS12320:QuestionBank01

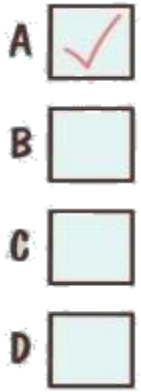
Your task: Functional Requirement 2

FR2: A teacher can add new questions to an existing question bank in two formats:

- **FR2a:Single choice**

Select one answer from ***N*** options (here, ***N*** is between 1 and 10). For the single-choice question type, the teacher:

  - enters the question as text
  - provides each question answer as text, up to ***N*** answers
  - indicates which of the answers is correct; there must only be one right answer.



Your task: Functional Requirement 2

FR2: A teacher can add new questions to an existing question bank in two formats:

- **FR2b: Fill the blanks**

For the fill-the-blanks question type, the teacher:

- enters a single text statement, but within that text indicates where the blank words are missing. Note that the blanks can represent one or more words; it is up to you to decide how this is done,
- enters the correct words for the blanks; again, it is up to you to decide how this is done.

```
Abstraction, ___ ___, polymorphism, and ____ are the four
main theoretical principles of object-oriented
programming
```

Your task: Functional Requirements 3, 4, 5

FR3: A teacher can list and then remove questions from an existing bank.

FR4: A teacher can list all the question banks for a specific module.

FR5: A teacher can delete a question bank, but only when it is empty.

Your task: Functional Requirement 6

FR6: A student can select quiz. When they take the quiz, they can:

- **FR6a:** List the question banks for a specific module identifier. If the module identifier is unknown, then an error message should be displayed.

- **FR6b:** Take a specific question bank quiz using question bank unique identifier (see **FR1**). This will ask the student for the number of questions to display (call this number **Q**). If this is greater than the number of questions in the bank, then it will be set to the question bank size.

Your task: Functional Requirement 7

FR7: A student may end the question session at any time, at which point some statistics are displayed:

- A score, representing the number or percentage of correct answers out of the total questions in the question bank.
- The number of questions not answered.
- The elapsed time between the start of the quiz and the end.

Example:

```
You took 1 minutes and 15 seconds.
Your score is: 3 / 3.
You didn’t answer 0 questions.
```

Your task: Functional Requirement 8, 9, 10

FR8: The quiz displays **Q** questions in a random order. It indicates the number of questions to be displayed. Only one question at a time is displayed and the question number is clearly indicated.

FR9: The student decides when they want to move to the next question or go back to a previous question.

FR10: A single choice question is displayed to the student in an appropriate textual format allowing the student to choose an answer depending on the question type. For a fill-the-blanks question the student adds the missing words or phrases for the blanks. The form this takes is up to you.

Your task: Non-Functional Requirements 1, 2, 3

NFR1:

- The question bank must be persistent and stored in a text-based database: i.e. text files.
- You may decide on the format that you wish to use.
- You must **NOT** use a relational database nor any binary format, *i.e.* it must be possible to view the files with a text editor.

NFR2: A text-based, menu-driven user interface is required.

NFR3: It must be possible to plug in new kinds of question in the future, hence the need for inheritance.

Example:

```
Menu

Enter your option:
1 – View all modules and question banks
2 – Create new module with question banks
3 – Create new module
4 – Create question bank
5 – Edit question bank
6 – Save changes
7 – Quit
```

Marking scheme

Quality of the documentation: design as described, diagrams, grammar, spelling, etc.	Does the documentation follow instructions in points 2(a) to 2(d)?	20%
Implementation: Question bank management	This covers FR1, FR3, FR4, FR5	5%
Implementation: Creating new questions	This covers FR2: FR2a, FR2b	5%
Implementation: Running the quiz	This covers FR6, FR7, FR8, FR9, FR10	5%
Implementation: Persistence and UI	This covers NFR1, NFR2, NFR3	5%
General implementation and design	Other functions required are implemented correctly. Is the code of high quality, e.g. object oriented and following principles such as high cohesion and low coupling, good identifier names, indentation, small methods, commenting, error checking and exceptions, reading and writing a file, use of ArrayLists (or similar), packages?	20%
Implementation: use of inheritance	Has inheritance been used effectively?	10%
Testing	Does the documentation follow the instructions in point 2(e)?	15%
Evaluation	Does the documentation follow instructions in point 2(f)?	5%
Creativity & innovation	I am looking for applications that show flair, creativity or innovation. This addresses the Appendix AA 80% to100% assessment criteria that state: "and will more than completely fulfill the functional requirements." A good possibility here is to provide a graphical user interface for controlling the program: user controls. Another possibility would be to add further cryptographic algorithms or code breaking where the key is not known. Ask me if you are not sure.	10%

* For details see **Main-individual-assignment-brief.pdf** provided in the Blackboard

Marking grid

Marking grid

		Maximum marks
1	**Documentation on design and general**	**20**
1.1	General: Cover page	1
1.2	General: Contents page	1
1.3	General: Introduction of the assignment	2
1.4	Design: Use case diagram	3
1.5	Design: Class diagram	5
1.6	Design: Textual class descriptions	5
1.7	Design: Pseudo-code	2
1.8	General: Spelling/grammar	1
2	**Implementation: Question bank management**	**5**
2.1	FR1: Create question bank	2
2.2	FR3: List / remove questions	1
2.3	FR4: List banks for module	1
2.4	FR5: Delete bank when empty	1
3	**Implementation: Creating new questions**	**5**
3.1	FR2a: Single choice questions	3
3.2	FR2b: Fill the blanks questions	2
4	**Implementation: Running the quiz**	**5**
4.1	FR6: Attempt quiz	1
4.2	FR7: End question session/statistics	1
4.3	FR8: Quiz displays Q questions random	1
4.4	FR9: Student moving to next question	1
4.5	FR10: Question displayed in correct form	1
5	**Implementation: Persistence and UI**	**5**
5.1	NFR1: Persistence	2
5.2	NFR2: Program menu	1
5.3	NFR3: Plugging new questions	2
6	**General implementation and design**	**20**
6.1	Code quality: Applying OOP principles	10
6.2	Code quality: Appropriate identifier names	1
6.3	Code quality: Proper indentation	1
6.4	Code quality: Small methods	2
6.5	Code quality: Comments	2
6.6	Code quality: Error checking	2
6.7	Code quality: Package use	2
7	**Use of Inheritance**	**10**
7.1	The "is a" test works	4
7.2	Correct placement of fields	2
7.3	Correct placement of methods / overriding	2
7.4	Correct use of polymorphism	2
8	**Testing**	**15**
8.1	Test table	8
8.2	Appropriate screenshots	4
8.3	Discussion of testing	2
8.4	Shown running from OS command prompt window or terminal	1
9	**Evaluation**	**5**
9.1	What difficulties you faced?	1
9.2	What remains to be done?	1
9.3	What did you learn?	1
9.4	What marks should be awarded and why?	1
9.5	Brief description of creativity and innovation.	1
	Creativity and innovation	**10**
	Total	**100**

* For full details see **Main-individual-assignment-marking-grid.pdf** provided in the Blackboard

Hand-in

- Upload a single zip file (NOT rar or anything else: a 2% penalty will be applied if the wrong format is used) to Blackboard that contains:
  - Your IntelliJ project
  - A **PDF (NOT doc/docx or anything else)** document (2000-word (+/- 10%)) that contains description of the project.
    - 2% (two percent) points will be deducted if the wrong format is used
- You should name the submitted zip file as **YourEmailID-main-assignment-2023-24.zip**
  - Example: **far8-main-assignment-2023-24.zip**. Otherwise, 2% penalty will be applied if the wrong name of the zip file is provided.

* For details see Main-individual-assignment-brief.pdf provided in the Blackboard

Hand-in

- The PDF file will contain
  - A front cover page that includes your name, email, document date, and the title of this assignment.
  - Page 2 (two) must provide a ***content page***.
  - Page 3 (three) must start with an ***Introduction*** that says what is to follow and briefly what was achieved. This should also contain a UML use-case diagram that presents the requirements.
  - A section called ***Design*** that has a class diagram, and a textual description of each class and its relationships. I will be looking for well-designed classes, method names and attribute names. Include a pseudo-code example that shows implementation of one of your most complex algorithms.
  - A section called ***Testing***. This section should have a test table as described in class (see example provided with this assignment). This section should also contain screenshots of your program running (a screen shot for every function implemented). It must also have a discussion of test data used and the results of running the tests. **Show your program running from the command-line.**
  - A section called ***Evaluation***. Describe how you went about solving the assignment: what was difficult, what remains to be done, what did you learn, and what mark do you think you should be awarded and why. Have a sub-section for each of these.
  - This word count excludes the cover page and titles of diagrams and tables.

* For details see **Main-individual-assignment-brief.pdf** provided in the Blackboard

Important dates

- **Handed out:** Monday 11th March 2024

- **Hand-in:** Wednesday 8th May 2024, 1 pm via Blackboard upload [PROVISIONAL]

- **Feedback:** Friday 31st May 2024

* For details see **Main-individual-assignment-brief.pdf** provided in the Blackboard

IMPORTANT: Unacceptable Academic Practice (UAP)

- All reports and codes will be checked for plagiarism, copying and collusion

More on UAP at:

https://www.aber.ac.uk/en/academic-registry/handbook/regulations/uap/

Questions?

far8@aber.ac.uk

1872 PRIFYSGOL ABERYSTWYTH UNIVERSITY