Main Assignment - CS12320
Individual Assignment: Question bank

Nikola Nikolov

nin6@aber.ac.uk

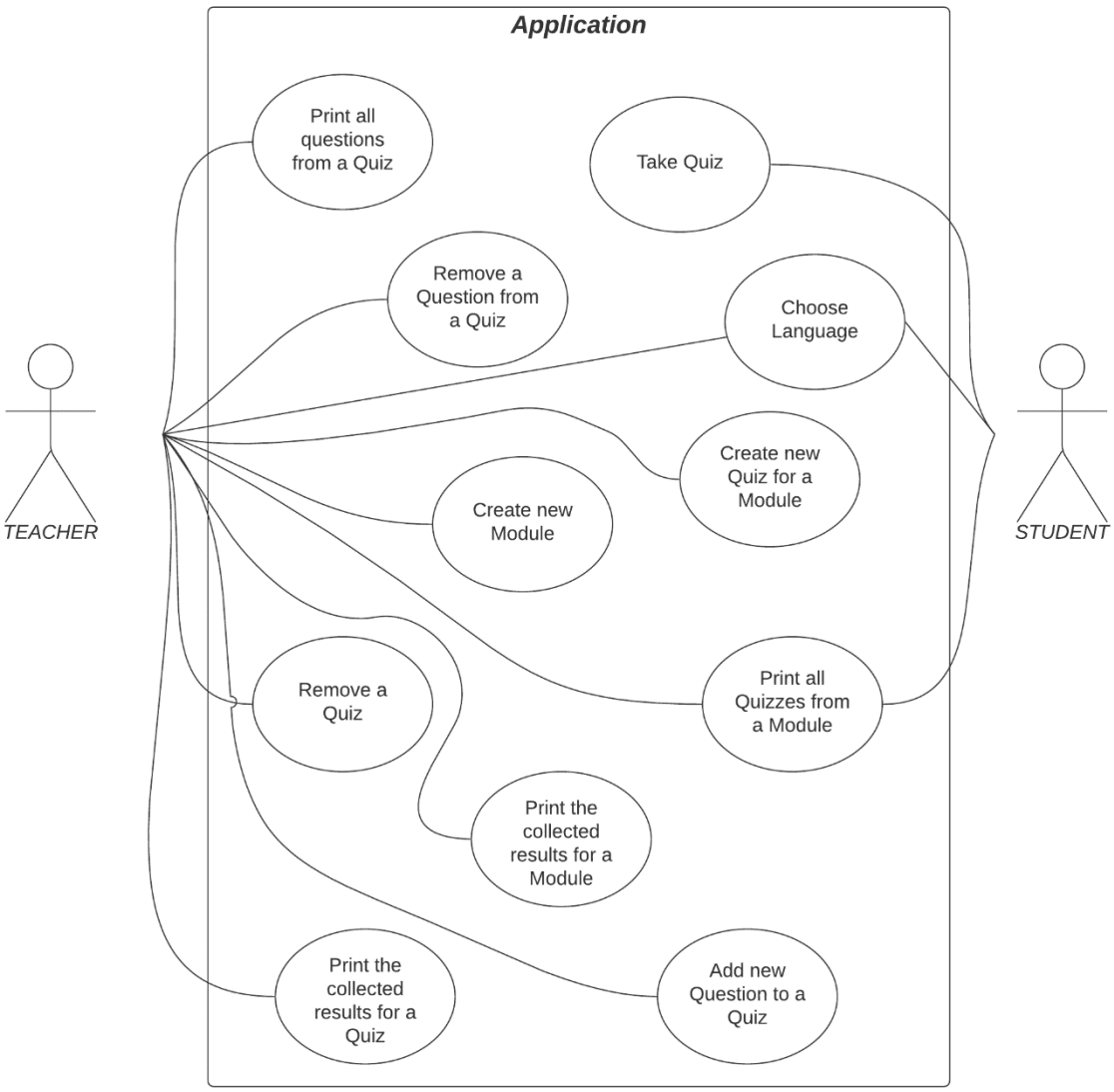
May 5, 2021

Contents

Introduction

That is a report on what I have done to complete my assignment. Brief explaining of the functionality of the prototype program I had to make is, a program that allows to teachers to create quizzes of every module that they want, while the students can do the quizzes and see their results. In this report I will explain how I solved the problems that I faced making the program. Also, the way I write the code and why I did it as I did. There is going to be a part that will be focus on the testing of the program and the thinks I learned while I was doing the program.

To help you with the understanding of my code I will put an UML use case diagram here in the beginning. On the UML use case diagram, you can see what the different kind of users can do and what they cannot.



After I finished the functional and non-functional requirement I deside to add a bit more flair jest to be sure that the customer will be satisfied (the lecture will give me more points). And by flair I meant that I make the Application in way that can store the results of every student that have done the quiz and the teacher will have the chance to see a sample statistic of the results only for a specific quiz or for the whole module. Also, I have added one more type of question, match-type of question (there are two columns, and you have to connect every line with the correct one on the other column).

Design

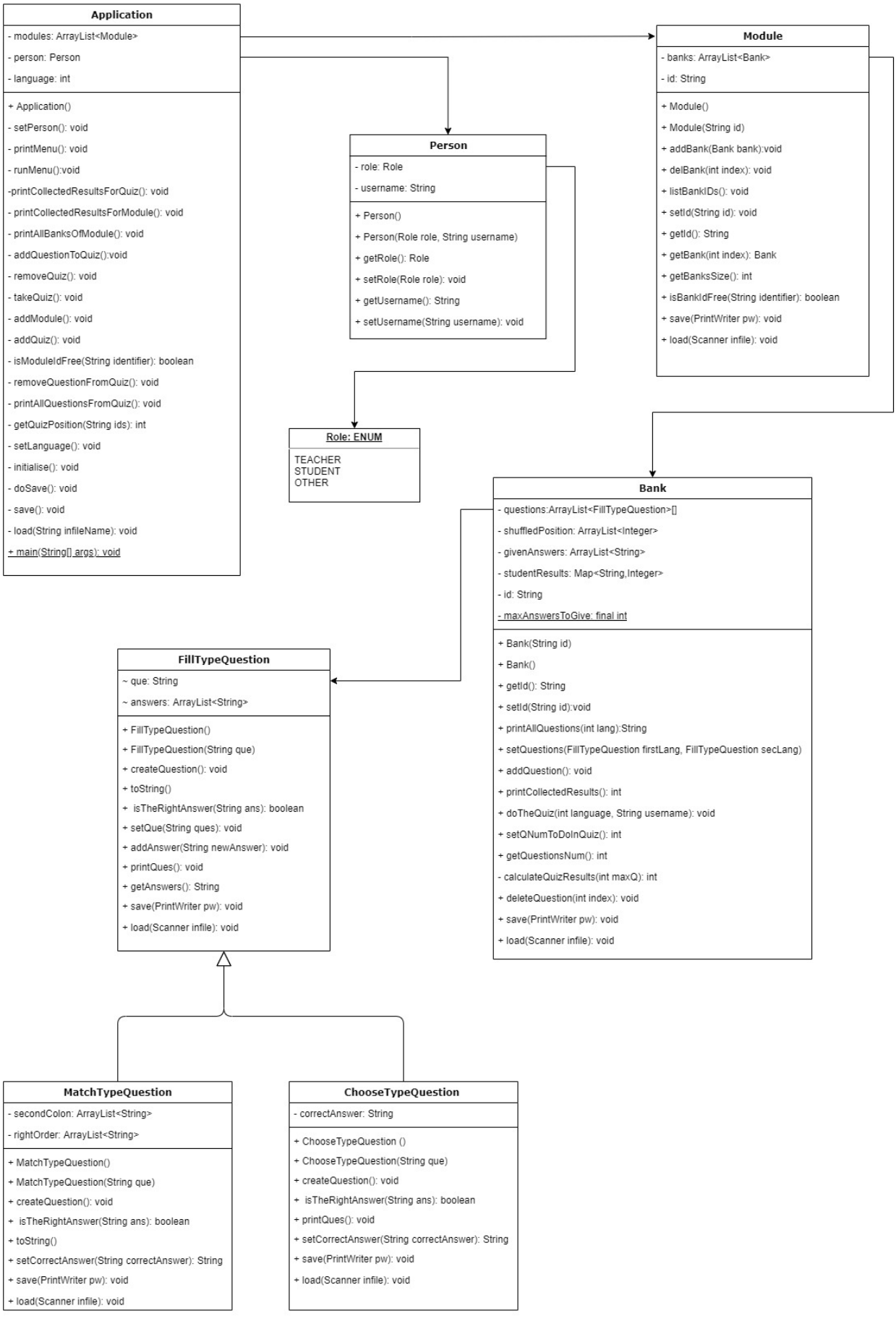
In this section we will look at the whey I created and connected the classes. That will be followed by explanation on every class individually and the functions that contains.

You can see the class diagram on the following page. However, I will put a SVG copy in the zip file just in case it is not readable enough because of the zooming.

At the beginning I was not sure what will be the best way to use inheritance for my question classes. I thought to create a super class called Question and all types of question to be inherited by it. But I realized that no matter how I do it I will need to override almost every method like createQuestion, isTheRightAnswer, printQues, even the save method, because all of them had to treat the received information differently and to print different thing. So I decide not to create a Question class which is going to have just to variables in it and unusable methods, Instead I made the Fill-TypeQuestion the super class because that class have the two variables that every type of question will need. I thought that creating that class will be unnecessary and I make me type more code for nothing, but still, I am not sure in that decision. Please let me know if the other way was better and why if so.

All type of questions are inherited by my Fill-TypeQuestion class which is connected to the Bank class. The Bank class can every type of questions stored in the array called “questions”. “questions” is an array of ArrayList, it is that why so it can store as much language as needed. For now, it is set to two as the requirement asked, but it can easily de set to what every number just with little corrections in the Bank class.

The Module class have an ArrayList of Bank, in same way as Application class have ArrayList of Modules. The Application class is the class that make the corresponding with the use in way so the use can choose which method to activate. The Application class have variable of type Person too. The person class is small but is maid so the program to have identifier like username and the role of the person. When the security is ad that class can expand and more useful. I decide that the Enum class to choose the role is a good Idea just because in future updates is likely to have more roles like directors, admins, or others.


Application
- modules: ArrayList<Module>
- person: Person
- language: int
+ Application()
- setPerson(): void
- printMenu(): void
- runMenu():void
-printCollectedResultsForQuiz(): void
- printCollectedResultsForModule(): void
- printAllBanksOfModule(): void
- addQuestionToQuiz():void
- removeQuiz(): void
- takeQuiz(): void
- addModule(): void
- addQuiz(): void
- isModuleIdFree(String identifier): boolean
- removeQuestionFromQuiz(): void
- printAllQuestionsFromQuiz(): void
- getQuizPosition(String ids): int
- setLanguage(): void
- initialise(): void
- doSave(): void
- save(): void
- load(String infileName): void
+ main(String[] args): void
Person
- role: Role
- username: String
+ Person()
+ Person(Role role, String username)
+ getRole(): Role
+ setRole(Role role): void
+ getUsername(): String
+ setUsername(String username): void
Module
- banks: ArrayList<Bank>
- id: String
+ Module()
+ Module(String id)
+ addBank(Bank bank):void
+ delBank(int index): void
+ listBankIDs(): void
+ setId(String id): void
+ getId(): String
+ getBank(int index): Bank
+ getBanksSize(): int
+ isBankIdFree(String identifier): boolean
+ save(PrintWriter pw): void
+ load(Scanner infile): void
Role: ENUM
TEACHER
STUDENT
OTHER
Bank
- questions:ArrayList<FillTypeQuestion>[]
- shuffledPosition: ArrayList<Integer>
- givenAnswers: ArrayList<String>
- studentResults: Map<String,Integer>
- id: String
- maxAnswersToGive: final int
+ Bank(String id)
+ Bank()
+ getId(): String
+ setId(String id):void
+ printAllQuestions(int lang):String
+ setQuestions(FillTypeQuestion firstLang, FillTypeQuestion secLang)
+ addQuestion(): void
+ printCollectedResults(): int
+ doTheQuiz(int language, String username): void
+ setQNumToDoInQuiz(): int
+ getQuestionsNum(): int
- calculateQuizResults(int maxQ): int
+ deleteQuestion(int index): void
+ save(PrintWriter pw): void
+ load(Scanner infile): void
FillTypeQuestion
~ que: String
~ answers: ArrayList<String>
+ FillTypeQuestion()
+ FillTypeQuestion(String que)
+ createQuestion(): void
+ toString()
+ isTheRightAnswer(String ans): boolean
+ setQue(String ques): void
+ addAnswer(String newAnswer): void
+ printQues(): void
+ getAnswers(): String
+ save(PrintWriter pw): void
+ load(Scanner infile): void
MatchTypeQuestion
- secondColon: ArrayList<String>
- rightOrder: ArrayList<String>
+ MatchTypeQuestion()
+ MatchTypeQuestion(String que)
+ createQuestion(): void
+ isTheRightAnswer(String ans): boolean
+ toString()
+ setCorrectAnswer(String correctAnswer): String
+ save(PrintWriter pw): void
+ load(Scanner infile): void
ChooseTypeQuestion
- correctAnswer: String
+ ChooseTypeQuestion ()
+ ChooseTypeQuestion(String que)
+ createQuestion(): void
+ isTheRightAnswer(String ans): boolean
+ printQues(): void
+ setCorrectAnswer(String correctAnswer): String
+ save(PrintWriter pw): void
+ load(Scanner infile): void

Questions

The question classes are not that different from each other. They have pretty much the same methods, but the methods work in different way. For example, print method for Fill-type questions must print only the sentence while Choose-type of question must print the answers as well.

In this classes an interesting method is the is theRightAnswer. That method receives information in a string, but every class treats the information in different way. For example, if the question is of choose to type the method will take the first char (it must be just one) and will cast the letter into a number in way that A will be equal to 0, B to 1 and so on. After that, it will check if on the place is the correct answer. But if the question is of fill type the same function will just loop through the answer and if there is a match it will return true.

Bank

The most complex method in this class is doTheQuiz. After initializing everything that will be needed in the class the setQNumToDoInQuiz method which ask the use how many questions will do from that quiz. After that, every time we check if the responds of the user are the same as the special characters that will move to next or previous question or submit the work. If the respond is not match for none of these it is going to be save as an answer for the question. When the user submits its work the calculateQuizResults is called. That method counts how many of the question ware left without and answer and print the num at the end. Meanwhile, it counts how many of the answers are correct by calling theRightAnswer method from the respective class. At the end prints the result in percentage and save the result and the username of the use.

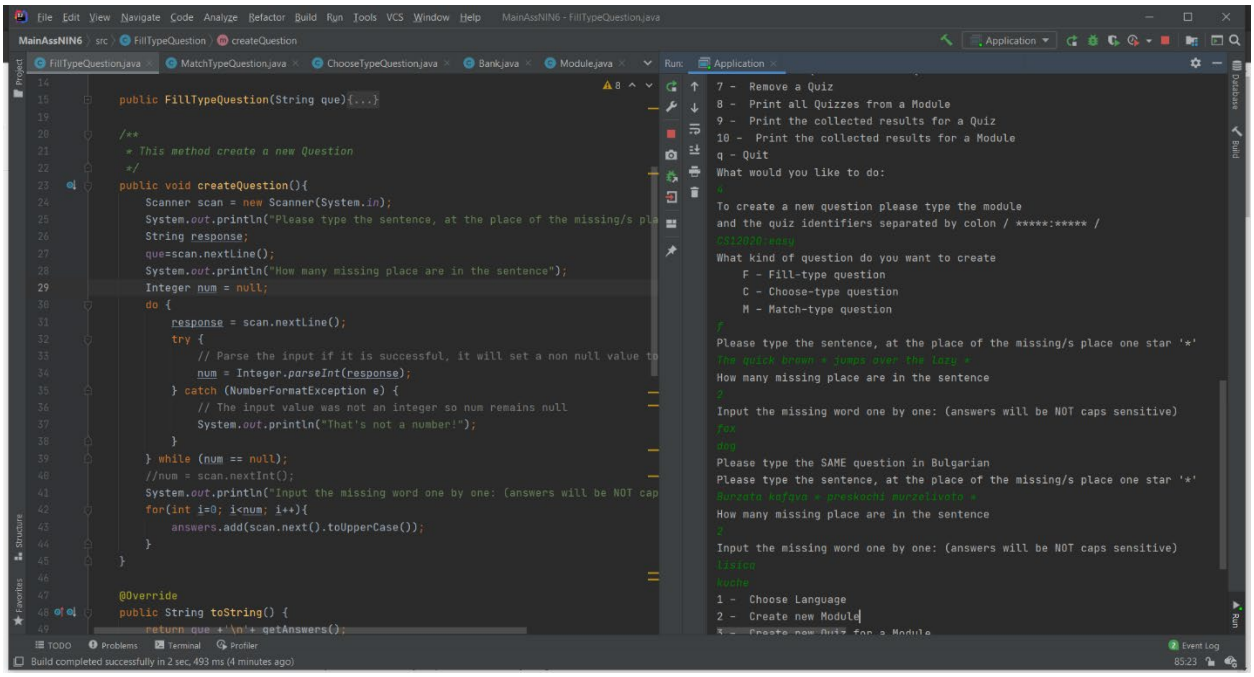
Application

The other worth explaining class is Application. The RunMenu method allows the user to use the methods that are purpose for them. Every use is separate in different method in way that they can be easy to understand.

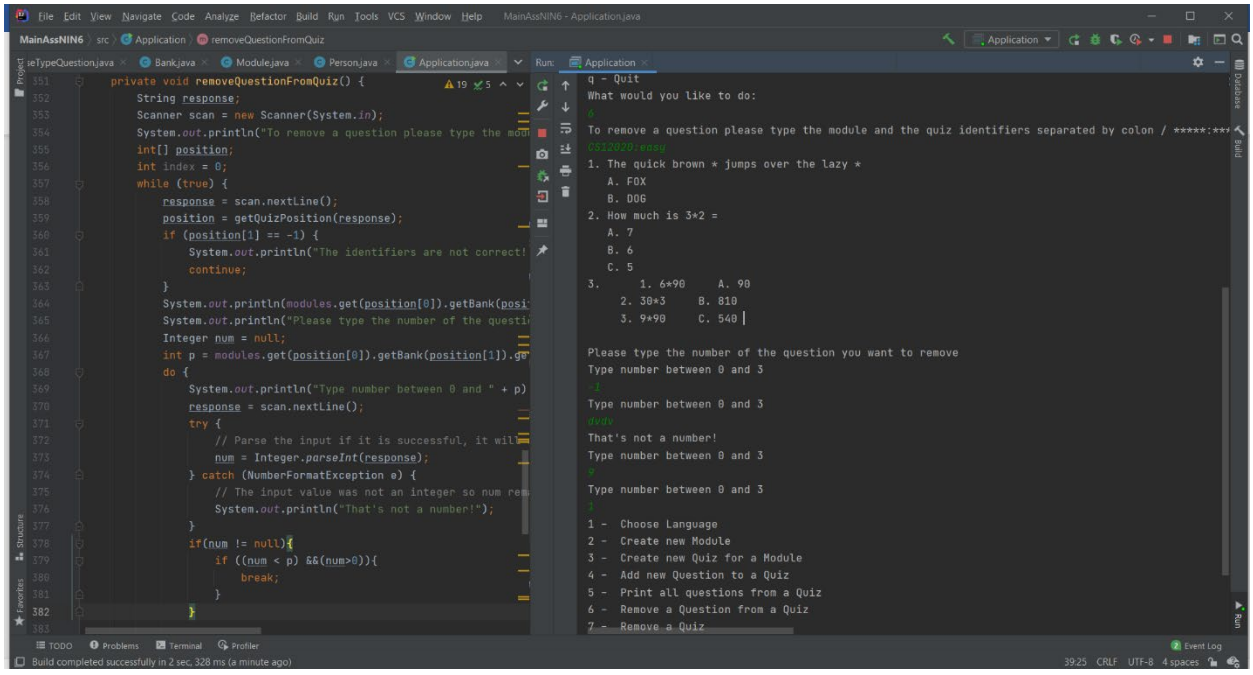
Testing

ID	Requirement	Description	Inputs	Expected outputs	Pass/ Fail	Comments
A1.1	FR1	Every module have unique ID	Module's 2 ID to be same as the id of the first one	Return error message and ask for new id because the id is already used	P	
A1.2	FR2	Add a question to a bank	Input the que = "How toll am I?" And answers to choose from	The question and the answers to be saved properly	P	
A1.3	FR3	Removing Question	IDs of the Module and The Bank, and the number of the question	To remove just the question on that number in this Bank	P	
			Wrong Module ID	To ask for another ID	P	
			Number that is not in the range	To ask for new number	P	
A1.4	FR4	List all the question banks	The ID of the Module	To print the questions	P	
			Wrong Module ID	To ask for another ID	P	
A1.5	FR5	Delete a question bank	Empty Bank	To be deleted	P	
			Full Bank	NOT to be deleted and message to say that the bank has to be empty	P	
A1.6	FR6	Take a quiz	Correct response	No errors	P	
			Incorrect response	Asking for correct input	P	
A1.7	FR7	End the quiz at any time	Try to end it	End it	P	

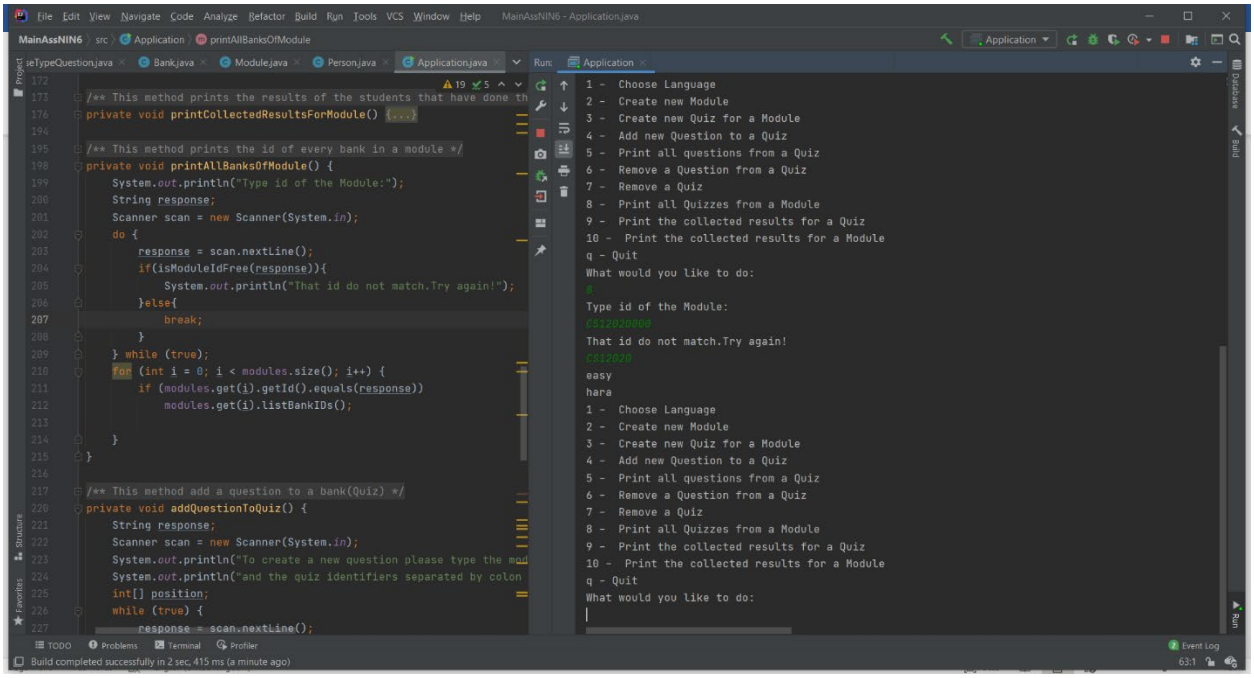
A1.8	FR8	The quiz displays Q questions in a random order.	Enter number in range	Print the number of questions	P	
			Enter bigger number that the questions	Print the max of questions	P	
			Enter negative number	Ask for new number	P	
A1.9	FR9	Student decides when they want to move to other question	Try to move to the next question	To print the next question	P	
			Try to move to the previous question	To print the previous question	P	
			Try to go over the limits (use next q on the last one)	Print a message that that cannot be done	P	
A1.10	FR10	question is displayed in an appropriate textual	--	question is displayed in an appropriate textual	P	
A2.1	NFR1	question bank must be persistent and stored in a text-based database	-	-	P	
A2.2	NFR2	text-based, menu-driven user interface	-	text-based, menu-driven user interface	P	
A2.3	NFR3	Two languages are supported	Choose language	Print the questions is chosen language	P	
A2.4	NFR4	possible to plug in new kinds of question	-	-	P	But little correction on the bank class will be necessary



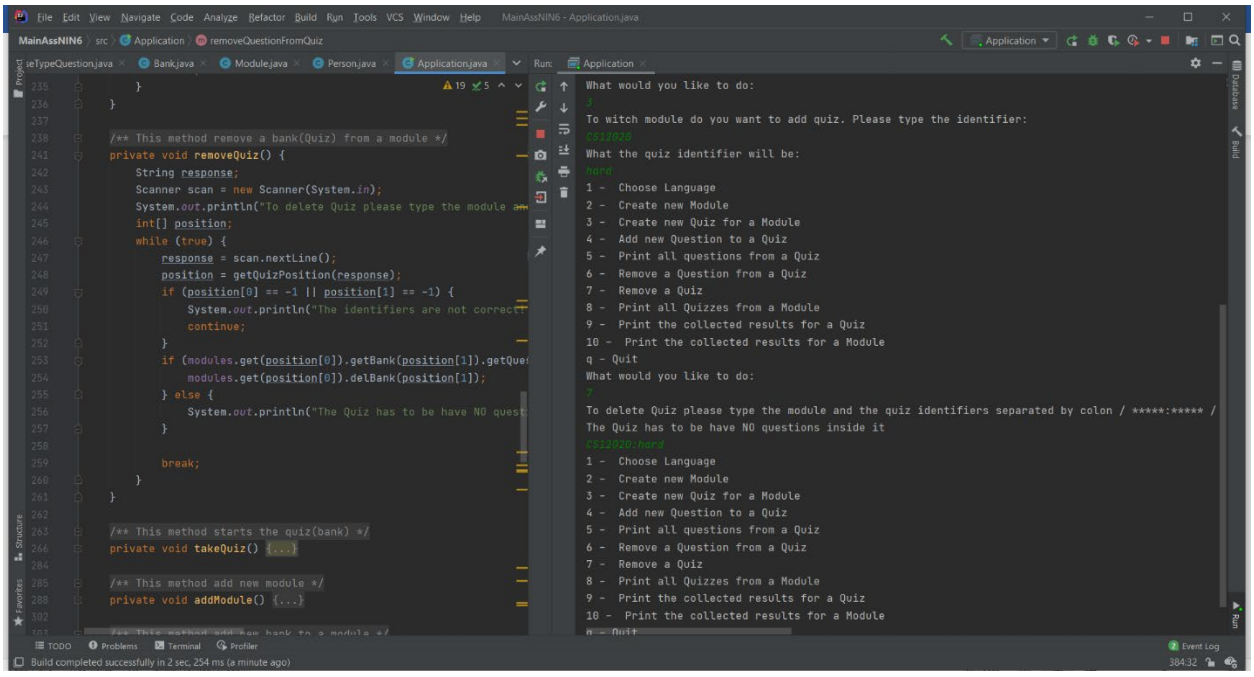
A1.2 I have input question.



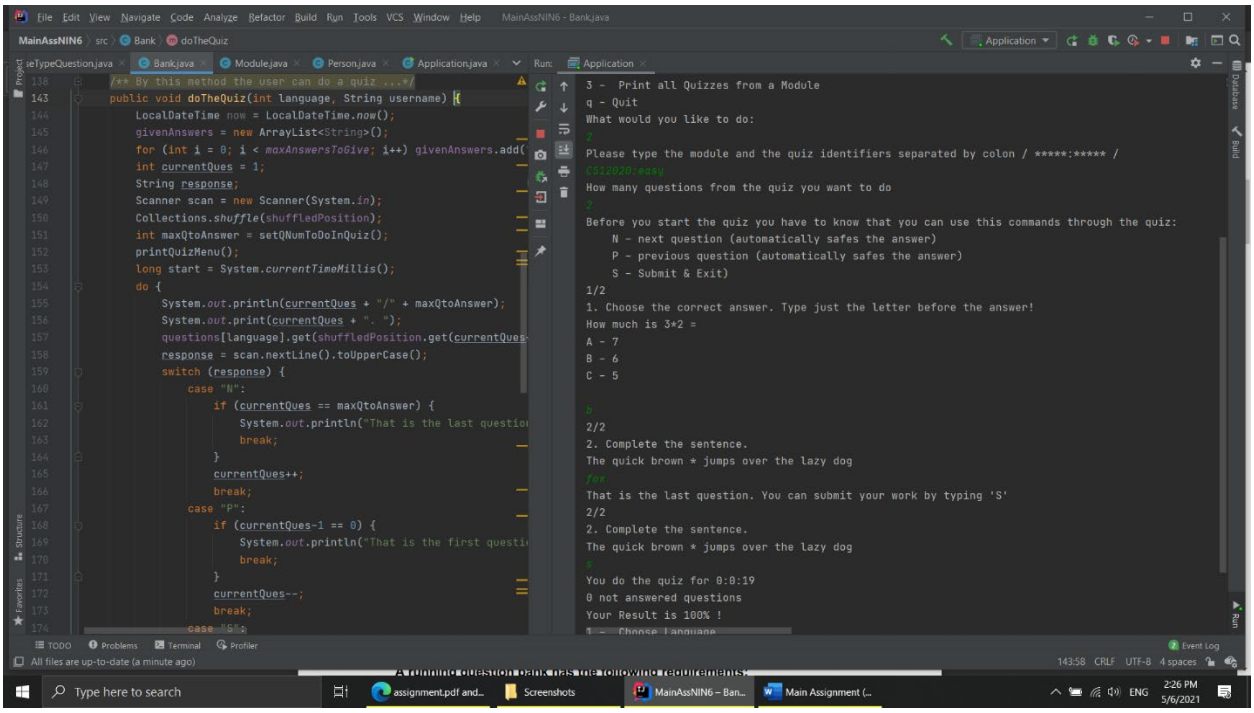
A1.3 Removing question - Trying to put incorrect numbers.



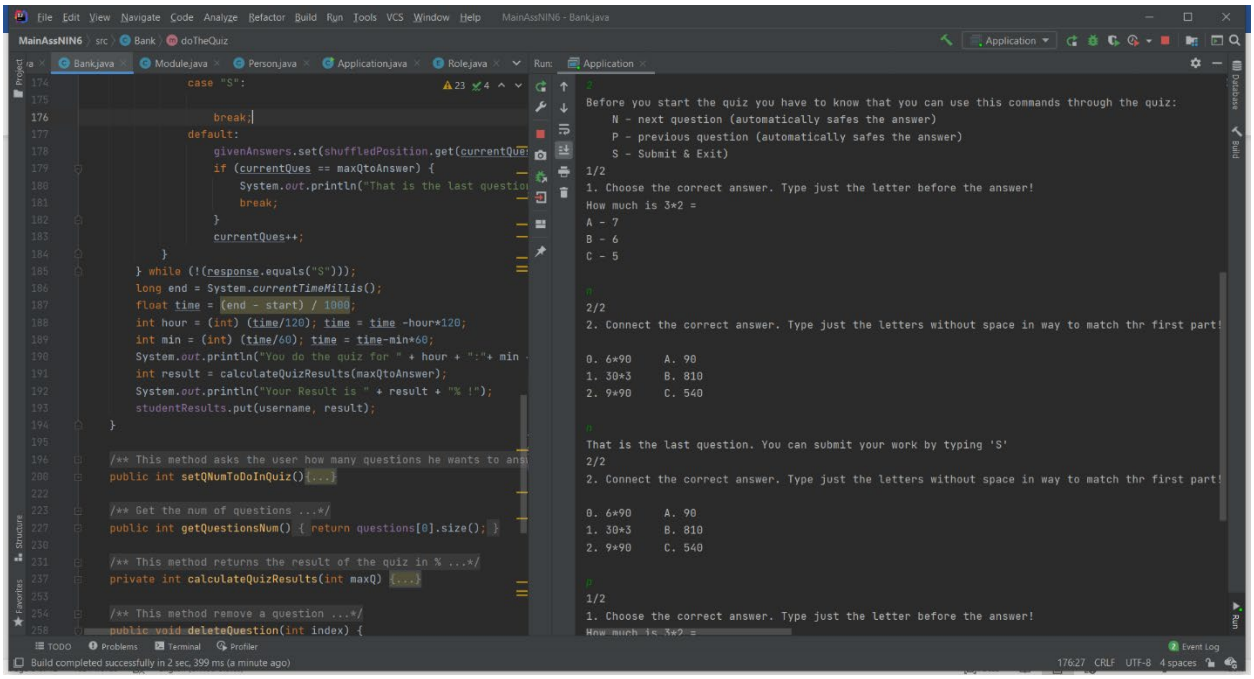
A1.4 List all banks - Trying to put incorrect id.



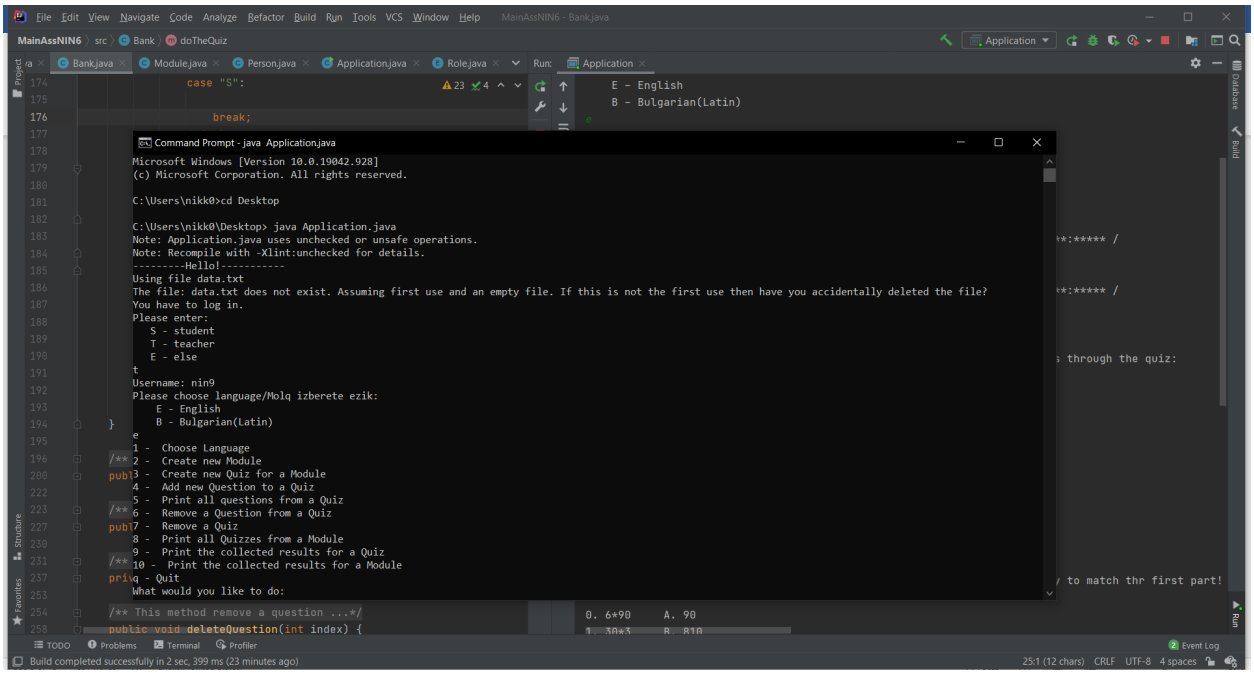
A1.5 Remove an empty bank



A1.6/ A1.7/ A1.8 Ending the Quiz and seen the result



A1.9 Moving to next and previous question



Starting the program from the command prompt.

Evaluation

I started by reading the assignment two or three times so I can memorize most of it. I left it for two days and try to imagine have everything will be connected and will how it will work. While I was working some ideas pop up and carefully collected them. After that period, I read the assignment two more times to be sure that I am not missing anything. I crated a class diagram in my textbook. That diagram was far from perfect but good enough to help me with the implementation of the program. I start from the question classes. The bank and the module followed shortly after that. At this point I have been made the classes with gutters and seters and some printing methods as well. Then I implement the Application class. First, I made the print menu class and the runMenu method. I put comments that explain what every swift case must do. Starting from the easiest one I cleated a private method in the Application class that called the right methods from the other classes. While I was expanding that class, I wrote some methods in the other classes in way that every function works properly.

There were some problems that was a bit difficult for me, but I googled them and read why and how other programs made a solution. That really helped me to expand my knowledge and to finish the assessment.

I decide for flair to add one more type of question and storing student results to monitor progress. I cannot assume have much my mark will be, because I tried me best and that is 90% on my knowledge. I am sure that there is better ways to do this program and the people that know this ways will mark my solution low. After all I hope that I receive 60-70%.