

Dokumentacja projektu

DRILL

Anna Baran, Kinga Kaczmarczyk

Aplikację budowałyśmy w paru krokach, gdzie każdy z nich jest opisany poniżej.

Milestone 1

Plik z pytaniami przykładowymi zamieściliśmy w resources. Wygląda on tak :

```
1. Stolica Polski to:  
a. Wieden  
b. Londyn  
>>> c. Warszawa  
d. Ateny  
  
2.* Na faladze Polski znajdują się kolory:  
>>> a. biały  
b. czarny  
>>> c. czerwony  
d. niebieski
```

* oznacza pytanie wielokrotnego wyboru, a >>> oznacza odpowiedź prawidłową.

Stworzyłyśmy taką początkową aplikację, dzięki której możemy przeprowadzić test. Okienka stworzone zostały za pomocą JavaFX.

W następnym milestone chciłybyśmy udoskonalić tą aplikację.

Milestone 2

W tym milestone zamierzamy zastosować wzorce projektowe takie jak Budowniczy oraz Strategia, aby nasza aplikacja była otwarta na rozbudowę, w szczególności na dodawanie nowych opcji przeprowadzania testu, opcji zliczania punktów oraz parsera.

Po konsultacjach zaimplementowałyśmy wzorzec Builder do tworzenia obiektów Question (w tym momencie znajduje się w tym samym pliku co klasa Question).

Dodałyśmy interface Iterator do klasy Loader, co usprawnia przemieszczanie się po kolejnych obiektach Question z listy.

Mamy także wzorzec Strategy w Counter i Loader. Dzięki temu możemy rozszerzać aplikację o kolejne opcje dotyczące przeprowadzania testu (daje to większą swobodę użytkownikowi jeśli chodzi o personalizację przeprowadzania Quizu).

```

classDiagram
    class App {
        +start(primaryStage: Stage)
        +main(args: String[])
    }
    class StartWindowController {
        -optionController: OptionWindowController
        -loader: FXMLLoader
        -loader1: FXMLLoader
        -optionParent: Parent
        +initialize(URL, ResourceBundle)
        -handleStartAction(ActionEvent)
        -handleReadyAction(ActionEvent)
        +handleOptionsButtonAction(ActionEvent)
        +handleFileChooserButton(ActionEvent)
        +start(Stage)
    }
    class OptionWindowController {
        -shuffleA: boolean
        -shuffleQ: boolean
        -replyQ: boolean
        +handleOkButtonAction(ActionEvent)
        +initialize(URL, ResourceBundle)
    }
    class Question {
        -questionText: String
        -answers: List<Answer>
        -isOneAnswer: boolean
        +getCorrectAnswers(): List<String>
    }
    class Answer {
        -answer: String
        -isCorrect: boolean
    }
    class QuizManager {
        -pointCounter: ICounter
        +checkAnswers(ArrayList<String>, Question)
        +equalAnswers(ArrayList<String>, Question): boolean
    }
    class IQuizManager {
        +checkAnswers(ArrayList<String>, Question)
    }
    class IParser {
        +readFile(): List<Question>
    }
    class ICounter {
        +addPoint()
        +subtractPoint()
    }
    class QuestionBuilder {
        -questionText: String
        -answers: List<Answer>
        -isOneAnswer: boolean
        -numberOfCorrectAnswers: int
        -isCompleteQuestion: boolean
        +buildQuestionText(String)
        +buildAnswer(String, boolean)
        +buildsOneAnswer(boolean)
        +build(): Question
    }
    class Loader {
        #currentQuestion: Question
        #questionPointer: int
        #questions: List<Question>
        +setQuestions(List<Question>)
        +getAnswers(): List<Answer>
    }
    class QuestionLoaderShuffler
    class QuestionLoader
    class PointCounter
    class PointCounterWithSubstracting
    class User {
        -nickName: String
        -points: int
    }

    App --> StartWindowController
    StartWindowController --> OptionWindowController
    StartWindowController --> Question
    StartWindowController --> QuizManager
    OptionWindowController --> QuizManager
    Question --> Answer : 1..*
    Question --> QuestionBuilder : 0..1
    QuestionBuilder --> Question
    QuestionBuilder --> IParser
    QuestionBuilder --> Loader
    IParser --> QuestionBuilder
    IParser --> Iterator
    Loader --> QuestionLoaderShuffler : 0..1
    Loader --> QuestionLoader : 0..1
    QuestionLoaderShuffler --> Question : 1..*
    QuestionLoader --> Question : 1..*
    QuizManager --> IQuizManager
    QuizManager --> ICounter
    ICounter --> PointCounter
    ICounter --> PointCounterWithSubstracting
    PointCounterWithSubstracting --> PointCounter
    User --> PointCounter
  
```

Milestone 3

- nowe okno wyświetlające statystyki gry (liczbę zdobytych punktów, ścieżkę do pliku, datę i godzinę odbycia testu),
- obsługę różnych trybów gry, które można wybrać w oknie opcji. Dodane opcje to: mieszanie pytań, mieszanie odpowiedzi i powtarzanie pytań jeżeli odpowiedź była błędna (lub częściowo dobra),
- obsługę różnych trybów naliczania punktów: odejmowanie punktu za błędną odpowiedź lub ułamek punktu za częściowo prawidłową odpowiedź,
- okna sygnalizujące użytkownikowi błędy (błąd parsowania, niewłaściwa ścieżka do pliku).

```

classDiagram
    class LogWindowController {
        +handleOkButtonAction(ActionEvent)
        +initialize(URL, ResourceBundle)
    }
    class StartWindowController {
        -optionController: OptionWindowController
        -loader: FXXMLLoader
        -optionParent: Parent
        +initialize(URL, ResourceBundle)
        +handleStartAction(ActionEvent)
        +handleReadyAction(ActionEvent)
        +handleOptionsButtonAction(ActionEvent)
        +handleFileChooserButton(ActionEvent)
        +start(Stage)
    }
    class ErrorWindowController {
        +setText(String)
        +initialize(URL, ResourceBundle)
    }
    class OptionWindowController {
        -shuffleAnswer: boolean
        -shuffleQuestions: boolean
        -replyQuestions: boolean
        -counterOption: CounterOption
        -group: ToggleGroup
        +handleOkButtonAction(ActionEvent)
        +handleStartAction(ActionEvent)
        +handleFileChooserAction(ActionEvent)
        +initialize(URL, ResourceBundle)
    }
    class MainWindowController {
        -questionLoader: Loader
        -manager: IQuizManager
        -group: ToggleGroup
        -questionNumber: int
        +initialize(URL, ResourceBundle)
        +handleCheckAction(ActionEvent)
        +handleNextAction(ActionEvent)
        +setPath(String)
        +setOptions(boolean, boolean, boolean, CounterOption)
        +handleYesAction()
        +handleNoAction()
        +nextQuestion()
        +getAnswer(): ArrayList<String>
    }
    class QuizManager {
        -pointCounter: ICounter
        -counterOption: CounterOption
        -equalAnswers(ArrayList<String>, Question): boolean
    }
    class QuizManagerWithPartialGoodAnswers {
        -pointCounter: ICounter
        -equalAnswers(ArrayList<String>, Question): boolean
    }
    class Answer {
        -answer: String
        -isCorrect: boolean
    }
    class Question {
        -questionText: String
        -answers: List<Answer>
        -isOneAnswer: boolean
        +getCorrectAnswers(): List<String>
    }
    class QuestionLoader {
        -currentQuestion: Question
        -questionPointer: int
        -questions: List<Question>
        -shufflingAnswers: boolean
        -shufflingQuestions: boolean
        -replayingQuestions: boolean
        +setQuestions(List<Question>)
        +setOptions(boolean, boolean, boolean)
        +getAnswers(): List<Answer>
        +wrongAnswer()
    }
    class IParser {
        +readFile(): List<Question>
    }
    class IQuizManager {
        +checkAnswers(ArrayList<String>, Question)
    }
    class ICounter {
        -score: double
        +addPoint(double)
        +subtractPoint(double)
    }
    class FileParser {
        -bufferedReader: BufferedReader
        -questions: List<Question>
        +addQuestionToTest(String): String
        +isnt(String): boolean
        +parseQuestion(String, QuestionBuilder)
        +parseAnswers(String, QuestionBuilder)
    }
    class IParser {
        +readFile(): List<Question>
    }
    class IQuizManager {
        +checkAnswers(ArrayList<String>, Question)
    }
    class ICounter {
        -score: double
        +addPoint(double)
        +subtractPoint(double)
    }
    class CounterOption {
        -NORMAL
        -SUBTRACTING
        -PARTIAL
    }
    class StatisticWriter {
        +writeToFile(double, String)
    }
    class StatisticReader {
        -buffer: BufferedReader
        -logs: ObservableList<String>
    }
    class PointCounter {
    }

    LogWindowController --> StartWindowController
    StartWindowController --> ErrorWindowController
    StartWindowController --> OptionWindowController
    OptionWindowController --> MainWindowController
    MainWindowController --> QuizManager
    QuizManager --> QuizManagerWithPartialGoodAnswers
    MainWindowController --> Question
    Question --> Answer
    Question --> QuestionLoader
    QuestionLoader --> IParser
    QuestionLoader --> IQuizManager
    QuestionLoader --> ICounter
    QuestionLoader --> CounterOption
    QuestionLoader --> StatisticWriter
    QuestionLoader --> StatisticReader
    QuestionLoader --> PointCounter
    IParser --> FileParser
    IQuizManager --> IQuizManager
    ICounter --> ICounter
    CounterOption --> CounterOption
    StatisticWriter --> StatisticWriter
    StatisticReader --> StatisticReader
    PointCounter --> PointCounter
  
```

Podsumowanie

Nasz system przeprowadzania testów jednokrotnego i wielokrotnego wyboru mogłybyśmy oczywiście rozszerzyć. Niestety, przez ograniczenia czasowe nie zdołaliśmy zawrzeć w projekcie wszystkich naszych pomysłów.