



# TURQUIZ

QUIZ GAME



# GOALS

Our project focuses on developing an interactive application for teaching basic Turkish, designed to increase engagement and efficiency in the learning process. Problem Statement: Many students find traditional exam preparation methods boring and uninteresting. Solution: Our app uses G to allow you to select the topics and types of quizzes you need to prepare effectively.





# ABOUT US

Nargiza (Head) Responsibilities: Project planning and implementation, quality assurance.

Gulayim (Developer) Responsibilities: System architecture and design, Implementation.

Adina (Designer) Responsibilities: User interface design.



# PROBLEM AND SOLUTION

Formulation of the problem: Many students have difficulty with traditional exam preparation methods, finding them boring and uninteresting. There is a need for an interactive platform that enhances learning through game-based quizzes

Solution: Our solution is an exam preparation app that uses brainstorming to make exam preparation fun and effective for students.



# 1) FACTORY METHOD:

We used the Factory method to take questions for our quiz from files. The most important thing here is that we have a class in which we initialize, depending on external conditions, the class we need for the interface.

3 usages

```
private void runWindow(Integer type) throws IOException {  
    String path = "";  
    FXMLLoader fxmLoader = null;  
    PageController controller;  
    Question question=null;  
    switch (type) {  
        case (1):  
            path = "/com/example/myproject/page3Manual.fxml";  
            question = new QuestionManualAnswer( questionsFilePath: "questionsFile.txt", answersFilePath: "answersFile.txt",stageLevel);  
            break;  
        case (2):  
            path = "/com/example/myproject/page3True.fxml";  
            question = new QuestionTrueFalse( questionsFilePath: "questionsTFFile.txt", answersFilePath: "answersTFFile.txt", stageLevel);  
            break;  
        case (3):  
            path = "/com/example/myproject/page3variants.fxml";  
            question = new Question4Answer( questionsFilePath: "question4File.txt", answersFilePath: "answers4File.txt", stageLevel);  
            break;  
    }  
}
```

1) FACTORY METHOD: CHECK ANSWER AND NEXTQUESTION METHODS  
WHICH GIVES US QUESTIONS FOR OUR QUIZ AND CHECK THE RIGHT  
ANSWERS



## 2) FACADE:

It was used to "count and generate a result line.

We pass an instance of the question class into it, and it itself pulls out the necessary information.

```
public void setQuestions(Question questions) {  
    this.questions = questions;  
}
```

3 usages

```
public void showStatistic1() {  
    FasadeRezult result = new FasadeRezult();  
    String rez = result.result(questions);  
    score.setText(rez);  
}
```

```
public class FasadeRezult {  
    3 usages  
    public String result(Question question){  
        StringBuilder stBuilder = new StringBuilder("");  
        stBuilder.append(question.getCountRightAnswer())  
                .append("/")  
                .append(question.getCountAnswer());  
  
        return stBuilder.toString();  
    }  
}
```

2) IN OUR CASE, WE USED FACADE TO SHOW HOW MANY POINTS OR CORRECT ANSWERS WE GET. WE USED FACADE IN THE LAST PAGE IN OUR QUIZ - RESULT PAGE



### 3) STRATEGY:

Actually, you have the Question.java Interface and its implementations: Question4Answer.java, QuestionManualAnswer.java and QuestionTrueFalse.java. when we select one of 3 options on the screen (manual, true or 4varianta) - then, in fact, we use 2 development patterns at once. The most important thing here is that we have a class in which we initialize, depending on external conditions, the class we need for the interface. - this is one pattern. And the second is the very fact that we have one interface and several of its implementations.



4 usages 3 implementations

```
public void incrementRightAnswerCount();
```

1 usage 3 implementations

```
public Integer getCountRightAnswer();
```

1 usage 3 implementations

```
public Integer getCountAnswer();
```

2 usages 3 implementations

```
public Answer4 answersForQuestion();
```

3 usages 3 implementations

```
public Integer getLevel();
```

```
@Override
```

```
public void initLevel(Question question) {  
    this.questions = question;  
    textArea.setText(questions.nextQuestion());  
    rez = false;  
}
```

### 3) STRATEGY OF COUNTING ANSWERS AND GETTING LEVEL





THANKS FOR ATTENTION!

THE END