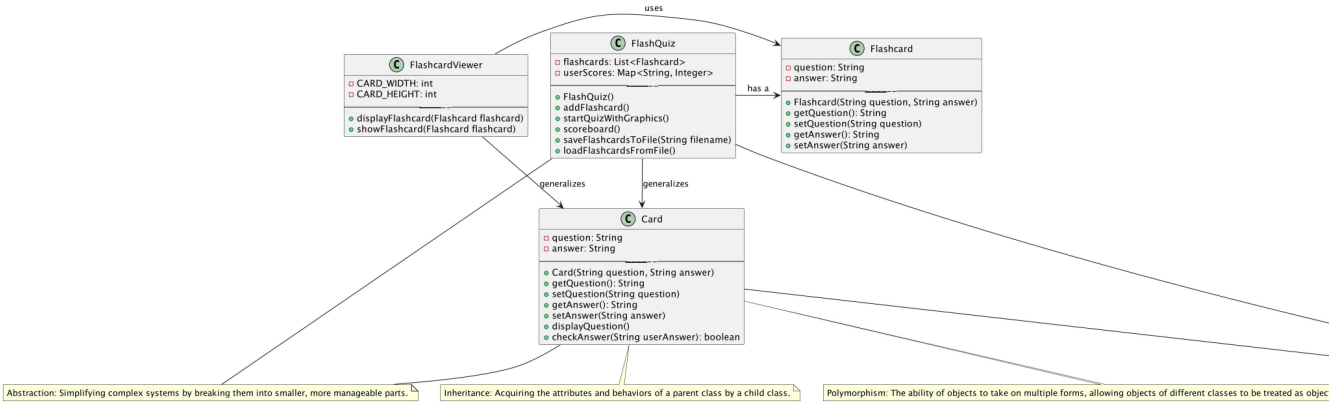
DOCUMENTATION

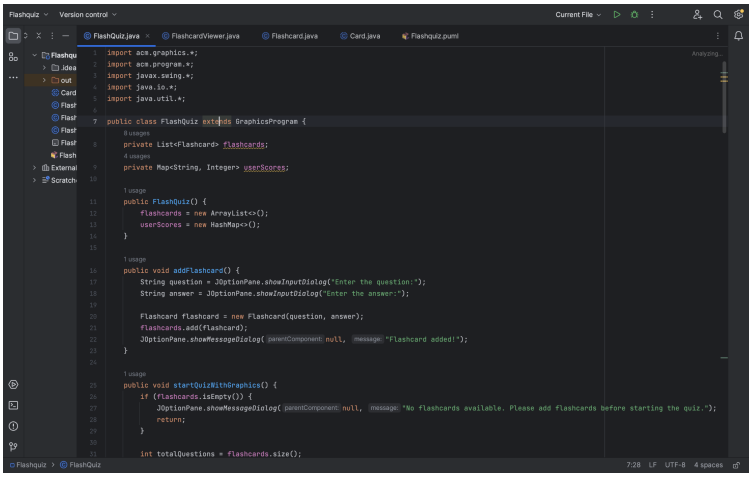
- **a) Include UML Diagrams (with explanations)**



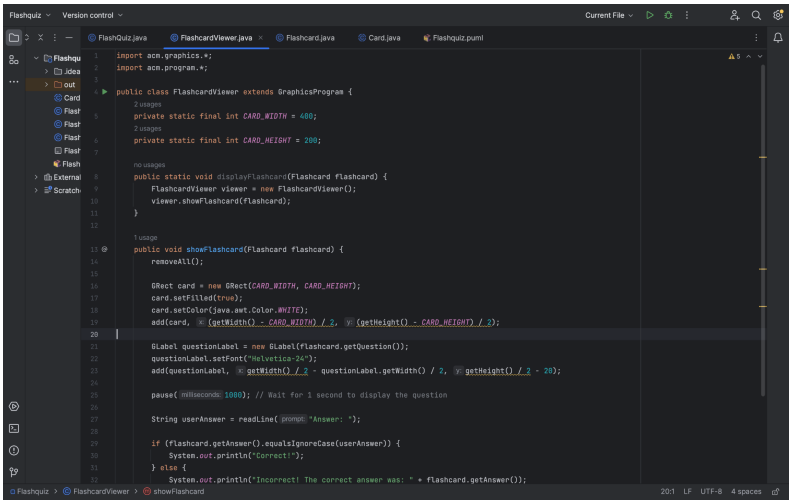
➢ Explanation: The UML diagram depicts the class structure and relationships of the FlashQuiz program, comprising four classes: FlashQuiz, Flashcard, FlashcardViewer, and Card. The FlashQuiz class manages flashcards and user scores, while the Flashcard class represents individual flashcards with questions and answers. FlashcardViewer displays flashcards in a graphical interface, and Card offers an alternative flashcard representation. The four pillars of object-oriented programming further guide the design. Abstraction simplifies complexity by breaking systems into manageable parts; encapsulation bundles data and methods within classes, providing data protection. Inheritance allows child classes to inherit attributes and methods from parents, promoting code reuse, while polymorphism enables objects of different classes to be treated as a common superclass. These principles ensure the program's structure, reusability, and maintainability, making FlashQuiz a potent educational tool. Through a well-crafted flashcard-based quiz system and interactive learning experience, FlashQuiz empowers students to enhance knowledge retention and self-improvement, fostering a lifelong passion for learning.

- **b) Screenshot (with explanations)**

➢



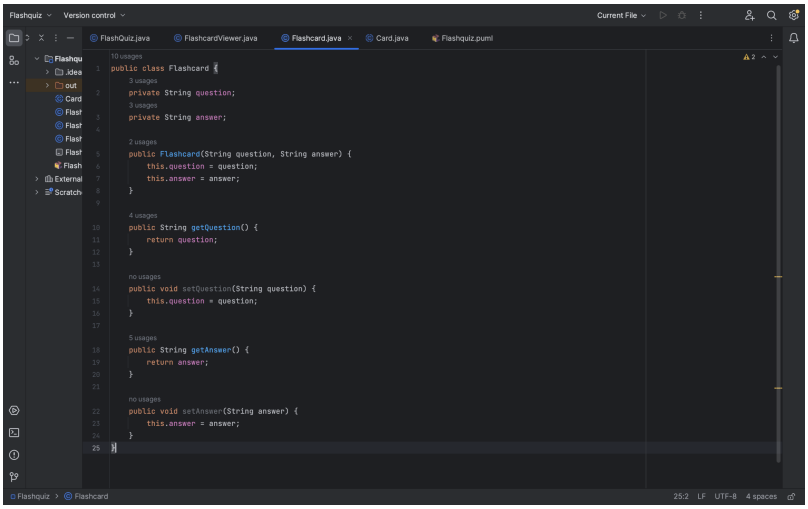
➢ Explanation: The main code



```
import acm.graphics.*;
import acm.program.*;

public class FlashcardViewer extends GraphicsProgram {
    private static final int CARD_WIDTH = 400;
    private static final int CARD_HEIGHT = 200;

    public static void displayFlashcard(Flashcard flashcard) {
        FlashcardViewer viewer = new FlashcardViewer();
        viewer.showFlashcard(flashcard);
    }

    public void showFlashcard(Flashcard flashcard) {
        removeAll();

        GRect card = new GRect(CARD_WIDTH, CARD_HEIGHT);
        card.setFilled(true);
        card.setColor(java.awt.Color.WHITE);
        add(card, (getWidth() - CARD_WIDTH) / 2, (getHeight() - CARD_HEIGHT) / 2);

        GLabel questionLabel = new GLabel(flashcard.getQuestion());
        questionLabel.setFont("Helvetica-24");
        add(questionLabel, getWidth() / 2 - questionLabel.getWidth() / 2, getHeight() / 2 - 20);

        pause(1000); // Wait for 1 second to display the question

        String userAnswer = readLine("Answer: ");

        if (flashcard.getAnswer().equalsIgnoreCase(userAnswer)) {
            System.out.println("Correct!");
        } else {
            System.out.println("Incorrect! The correct answer was: " + flashcard.getAnswer());
        }
```

➢

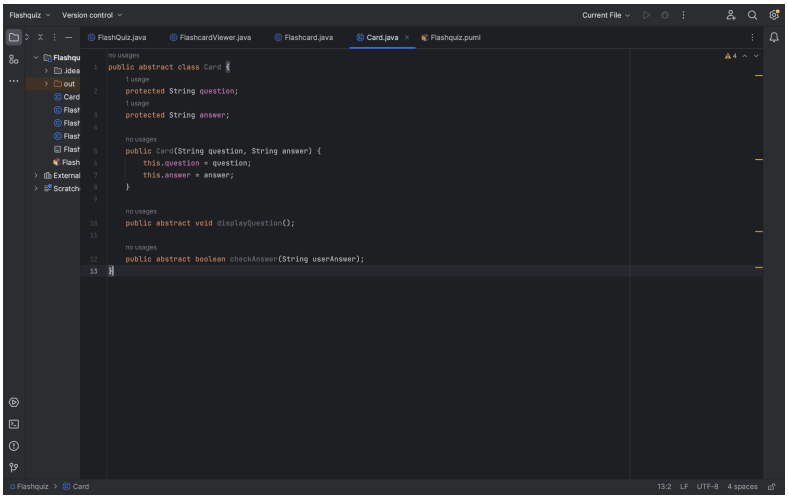
➢ Explanation: The code that makes acm graphics



```
public class Flashcard {
    private String question;
    private String answer;

    public Flashcard(String question, String answer) {
        this.question = question;
        this.answer = answer;
    }

    public String getQuestion() {
        return question;
    }

    public void setQuestion(String question) {
        this.question = question;
    }

    public String getAnswer() {
        return answer;
    }

    public void setAnswer(String answer) {
        this.answer = answer;
    }
}
```

➢

➢ Explanation: the code that makes the flash card



```
public abstract class Card {
    protected String question;
    protected String answer;

    public Card(String question, String answer) {
        this.question = question;
        this.answer = answer;
    }

    public abstract void displayQuestion();

    public abstract boolean checkAnswer(String userAnswer);
}
```

➢

➢ Explanation: the code that makes the card

❖ **c) Contribution per member**

➢ Andrei Baltazar

        - Coded Flashquiz.java
        - Coded FlashcardViewer.java
        - PUML of the code
        - Project Proposal (Parts III and IV)
        - Project Final Paper (Related Works, Conclusion/FutureFramework)
    - Hanz Bañas
        - Coded Flashquiz.java
        - Coded Flaschard.java
        - Coded Card.java
        - Project Proposal (Parts I and II)
        - Project Final Paper (Introduction, Implementation/OOP Aspects)
    - Jerico Bernasol
        - Managed the Github Account
        - Project Proposal (Part V)
        - Project Final Paper (Proposed Application, Walkthrough/Data/Results)

- **d) Explain the OOP aspects utilized in the project**

Aspects of object-oriented programming (OOP) have been incorporated into this flashcard quiz program to support code organization, reuse, and extension. The following are the key OOP elements used in the program, including the four pillars:

a. **Classes and Objects**: The program utilizes multiple classes such as Flashcard, Card, FlashcardViewer, and FlashQuiz. These classes represent the building blocks of the application, and objects of these classes interact to implement the desired functionalities.

b. **Encapsulation**: The Flashcard class encapsulates the question and answer data as private fields and provides getter and setter methods to access and modify this data. The use of private access modifiers ensures that the internal state of the Flashcard objects is well-controlled.

c. **Inheritance**: The Card class serves as an abstract superclass, defining common methods like displayQuestion() and checkAnswer(). The Flashcard class extends Card, inheriting the abstract methods and providing its specific implementations.

d. **Polymorphism**: The Flashcard class demonstrates polymorphism by implementing the abstract methods declared in the Card class. This allows the program to treat Flashcard objects as Card objects, facilitating a more generalized and flexible design.

e. **Abstraction**: The Card class is an example of abstraction. It provides a blueprint for different card types while hiding the specific implementation details. Concrete subclasses (e.g., Flashcard) implement the abstract methods to define their unique behavior.

f. **Composition**: The FlashQuiz class utilizes composition to store multiple Flashcard objects in the flashcards list and user scores in the userScores map. This composition allows for better organization and handling of the components.

Overall, the program successfully applies key OOP concepts, creating a flashcard quiz application that is organized, scalable, and user-friendly. The successful implementation of these OOP characteristics assures that the software can be easily extended and altered for future enhancements or new features.

- **e) At least ten references**
  - ➢