

SKILL DEVELOPMENT

Project Report



DLithe Consultancy Services Pvt. Ltd.

Project Report Assessment

Student Name: Vaishnavi k kulkarni

USN No: 2JR23CS117

Assignment: Java

Organization: DLithe Consultancy Services Pvt. Ltd.

Supervisor's Name: Archana SM

Submitted to

Signature of Training Supervisor

Signature of Students

Date:

Date:

Table of Contents

Introduction	4
Background	5
Sentence Capitalizer Program	6
Training Experience	12
Key Learnings	13
Challenges	15
Applications	16
Conclusion	17

Introduction:

A **flashcard game** is a simple yet powerful educational tool designed to enhance learning and memorization. It is widely used by students and educators for practicing vocabulary, learning definitions, solving math problems, and preparing for exams. The core concept revolves around presenting a question or prompt to the user, who then tries to recall the correct answer before flipping the card to check their response. This process helps reinforce memory through active recall and spaced repetition.

Building a flashcard game using **Java** provides an excellent opportunity to understand and apply fundamental concepts of object-oriented programming (OOP). Java, being one of the most widely-used programming languages, is ideal for this kind of project due to its clarity, structure, and strong support for GUI development and file handling.

In a basic implementation of a flashcard game in Java, the system stores a collection of flashcards, each containing a question and its corresponding answer. The user interacts with the game by viewing one flashcard at a time, attempting to answer it, and then viewing the correct answer. The game may also track the number of correct and incorrect answers to provide feedback and motivate improvement.

The game logic can be implemented using **core Java** through a console-based interface. In this version, Java classes such as Scanner, ArrayList, and File can be used to manage user input, store flashcards dynamically, and read/write data to files. The basic flow includes creating a list of flashcards, randomly shuffling them, presenting them one at a time, accepting user responses, and giving appropriate feedback.

For a more advanced version, **Java Swing** or **JavaFX** can be used to create a graphical user interface (GUI). This makes the game more interactive and visually appealing. Users can click buttons to flip cards, input answers in text fields, and see real-time score updates. JavaFX, in particular, offers modern UI components, animation capabilities, and better support for multimedia, making it suitable for enhancing the user experience.

Background:

Flashcards have long been recognized as one of the most effective learning tools for promoting memorization and reinforcing knowledge. They are simple, compact cards that present a question, term, or concept on one side and the corresponding answer or explanation on the other. This format encourages **active recall**—a learning technique where individuals try to remember a piece of information without immediately seeing the answer, which significantly improves memory retention.

The origin of flashcards can be traced back to the 19th century when educators began experimenting with various tools to enhance learning efficiency. Although not known by the term “flashcard” at the time, similar methods were employed in classrooms to drill vocabulary and arithmetic. The term "flashcard" itself came into more common use in the 20th century as printed educational materials became widespread.

Flashcards are especially popular in language learning, where they are used to memorize vocabulary, grammar rules, and verb conjugations. However, their application extends well beyond language acquisition. Flashcards are now used in diverse fields including science (e.g., periodic table elements), history (e.g., important dates and figures), mathematics (e.g., formulae and equations), and standardized test preparation (e.g., SAT, GRE, MCAT).

This program is often introduced in beginner to intermediate Java programming courses as it covers multiple core concepts in a compact and meaningful exercise. It encourages learners to think about string handling, edge cases (like multiple spaces or punctuation), and the importance of readable and maintainable code.

Overall, the Sentence Capitalizer Program is not just a utility tool but also an educational example that bridges theoretical understanding with practical coding skills. It reinforces fundamental programming techniques while providing a useful real-world application of text processing in Java.

Flashcard Program Requirement Collection:

Project Overview: The "flashcard" project is to design and develop an interactive flashcard program using Java that enables users to create, manage, and study flashcards in a structured and engaging way. The system is aimed at helping learners memorize facts, vocabulary, or concepts efficiently through active recall and repetition.

Functional Requirements:

- These are the features the flashcard program must support.

1.1 Flashcard Management

- The system should allow users to **create** new flashcards.
- Users should be able to **edit** existing flashcards (update questions or answers).
- Users can **delete** flashcards from the set.
- The program should allow the import/export of flashcards from/to a file (e.g., .txt, .csv, or JSON).

1.2 Study Mode

- The system should **display flashcards one at a time**.
- Users can choose to see the **question side first** and input their answer.
- The system will reveal the correct answer after user input.
- Users should receive **feedback** (e.g., correct or incorrect).
- The flashcards should be presented in **random order** (optional shuffle feature).

1.3 Score Tracking and Feedback

- Track the number of **correct and incorrect answers**.
- Show a **summary** at the end of a session (e.g., total attempted, correct, percentage score).
- Optionally, maintain a **high score or progress history**.

Non-Functional Requirements:

- These describe how the system should behave and perform.

2.1 Usability

- The user interface should be **simple and intuitive** (console-based or GUI).

- Instructions should be clear and easy to follow.

2.2 Portability

- The program should run on any system with **Java installed**.
- No external dependencies (or minimal).

2.3 Performance

- The program should handle **hundreds of flashcards** without performance degradation.
- The program must be implemented in Java.
- It should run in a command-line or terminal environment.
- The code must be clean and readable, with meaningful method names.
- Proper use of Java's built-in classes such as Scanner and StringBuilder.

System Requirements:

- Operating System: Windows 7/8/10/11, macOS, or any Linux distribution
- Java Development Kit (JDK) version 8 or higher
- Integrated Development Environment (IDE) like IntelliJ IDEA, Eclipse, or NetBeans (optional but recommended)
- Any text editor or IDE (e.g., IntelliJ IDEA, Eclipse, VS Code).
- Support for file handling using Java's built-in I/O libraries

User Requirements:

- The user wants an easy-to-use flashcard application for learning and memorization.
- The user should be able to run the program without requiring complex setup

Assumptions and Constraints:

- Users have Java installed and are familiar with basic desktop or console applications.
- Flashcards will be created, saved, and loaded locally using plain text formats (e.g., .txt, .csv).
- application must be developed in Java and run on Windows, macOS, or Linux systems .

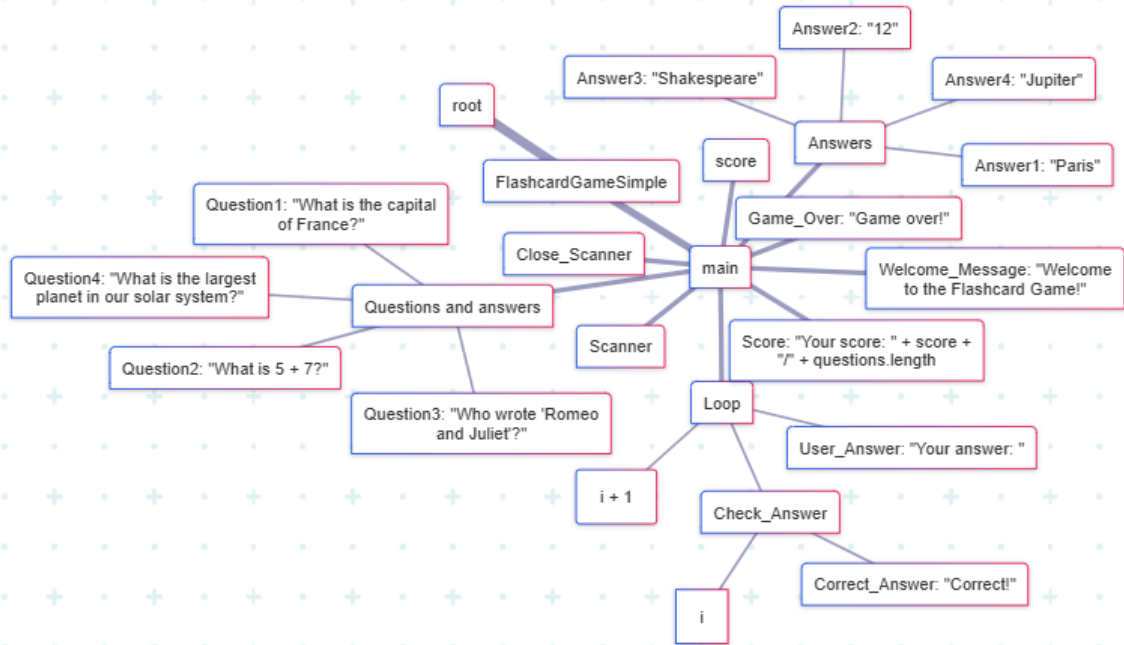
Input/Output Examples:

Input	Output
"What is the capital of Japan?"	"Tokyo"
" What is 2 + 2? "	"4"

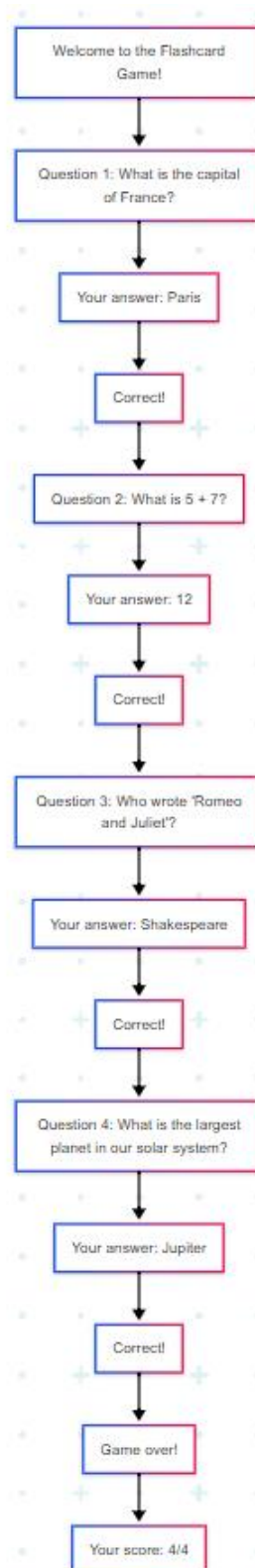
Program Features Summary:

Feature	Description
Create Flashcards	Users can add new question-answer pairs easily.
Edit and Delete	Modify or remove existing flashcards
Answer Feedback	Provides immediate feedback on correctness
Score Tracking	Keeps track of correct and incorrect answers.

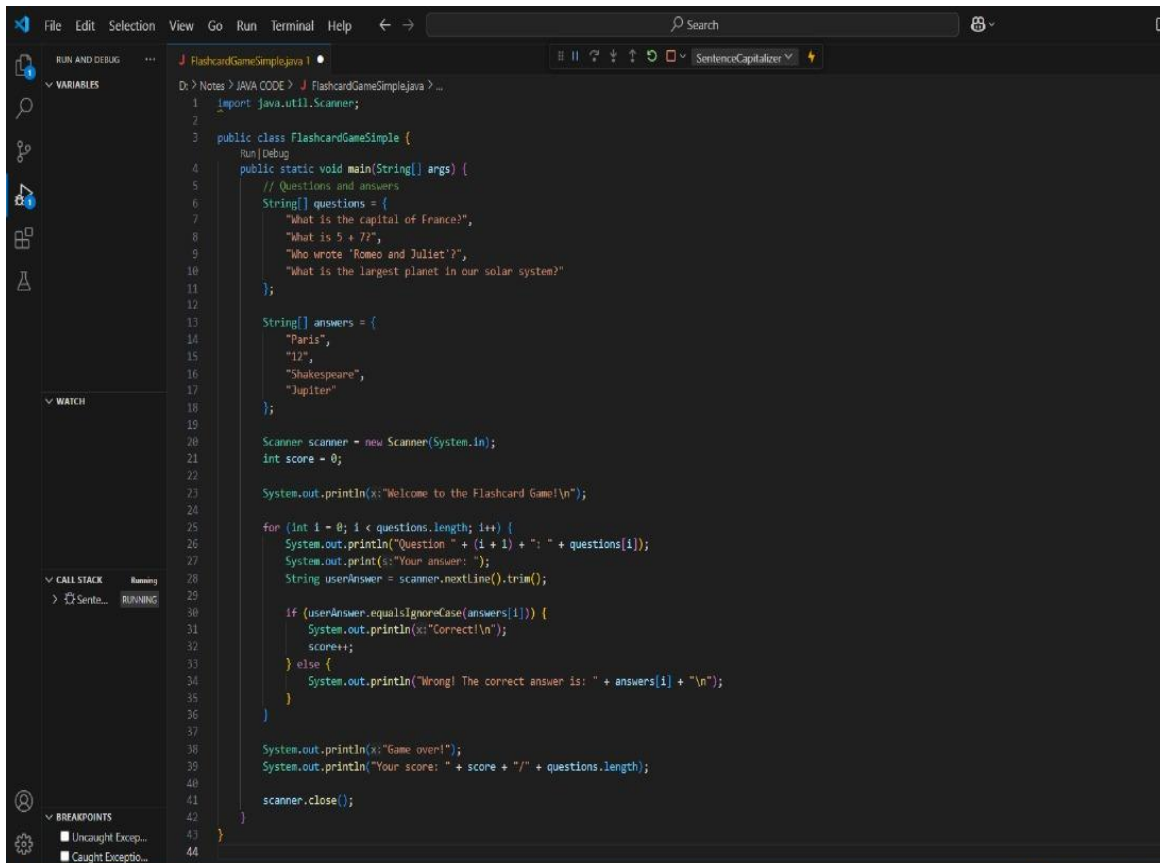
Mind Map:



Flowchart:



Project Development Images:



The screenshot shows an IDE with a dark theme. The main editor displays a Java file named `FlashcardGameSimple.java`. The code is as follows:

```
1 import java.util.Scanner;
2
3 public class FlashcardGameSimple {
4     // Questions and answers
5     public static void main(String[] args) {
6         String[] questions = {
7             "What is the capital of France?",
8             "What is 5 + 7?",
9             "Who wrote 'Romeo and Juliet'?",
10            "What is the largest planet in our solar system?"
11        };
12
13        String[] answers = {
14            "Paris",
15            "12",
16            "Shakespeare",
17            "Jupiter"
18        };
19
20        Scanner scanner = new Scanner(System.in);
21        int score = 0;
22
23        System.out.println("Welcome to the Flashcard Game!\n");
24
25        for (int i = 0; i < questions.length; i++) {
26            System.out.println("Question " + (i + 1) + ": " + questions[i]);
27            System.out.print("Your answer: ");
28            String userAnswer = scanner.nextLine().trim();
29
30            if (userAnswer.equalsIgnoreCase(answers[i])) {
31                System.out.println("Correct!\n");
32                score++;
33            } else {
34                System.out.println("Wrong! The correct answer is: " + answers[i] + "\n");
35            }
36        }
37
38        System.out.println("Game over!");
39        System.out.println("Your score: " + score + "/" + questions.length);
40
41        scanner.close();
42    }
43 }
44
```

The IDE interface includes a menu bar (File, Edit, Selection, View, Go, Run, Terminal, Help), a search bar, and a sidebar with panels for VARIABLES, WATCH, CALL STACK (showing `Sent...`), and BREAKPOINTS. The status bar at the bottom indicates the file is in the `Run/Debug` mode.

Technologies used:



Output:

```
PS C:\Users\Vaishnavi> & 'C:\Program Files\Java\jdk-24\bin\java.exe' '-agentlib:jdwp=transport=dt_socket,server=n,suspend=y,address=localhost:59384' '--enable-preview' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\Vaishnavi\AppData\Local\Temp\vscodesas_5a68d\jdt_us\jdt.ls-java-project\bin' 'FlashcardGameSimple'
Welcome to the Flashcard Game!

Question 1: What is the capital of France?
Your answer: paris
Correct!

Question 2: What is 5 + 7?
Your answer: 12
Correct!

Question 3: Who wrote 'Romeo and Juliet'?
Your answer: william shakespeare
Wrong! The correct answer is: Shakespeare

Question 4: What is the largest planet in our solar system?
Your answer: jupiter
Correct!

Game over!
Your score: 3/4
PS C:\Users\Vaishnavi>
```

Training Experience:

Using flashcards to learn Java is an effective way to build a strong foundation through active recall and spaced repetition. You start by creating flashcards that cover essential Java topics such as syntax, data types, control flow statements, object-oriented programming concepts, common methods, and exception handling. Each day, you review a small set of these flashcards, attempting to recall the answer before flipping the card to check your knowledge.

Cards you find difficult are marked for more frequent review, ensuring you focus on weaker areas. To reinforce your understanding, you complement flashcard study with practical coding exercises that apply the concepts you've reviewed, such as writing simple programs using loops or classes.

By consistently tracking your progress and gradually introducing more advanced topics, flashcard-based training helps you learn Java in manageable, focused sessions that strengthen both memory and coding skills over time.

Key Learnings:

☐ **Basic Input/Output**

- Use Scanner to take user input from the console.
- Use System.out.println and System.out.print to display questions and prompts.

☐ **Data Storage with Arrays**

- Store questions and answers using a **2D array** (or other collections).
- Access elements via indexes, e.g., flashcards[i][0] for questions, flashcards[i][1] for answers.

☐ **Looping Through Questions**

- Use a for loop to iterate over all flashcards.
- Repeat question display and input for each iteration.

☐ **String Comparison**

- Use equalsIgnoreCase() to compare user input with the correct answer ignoring case differences.

☐ **User Interaction Flow**

- Prompt the user for input and read their response.
- Provide immediate feedback (correct or wrong).
- Show the correct answer if the user is wrong.

☐ **Score Tracking**

- Maintain a score variable to count correct answers.
- Display the total score at the end of the game.

☐ **Program Structure**

- Organize code inside the main method for simplicity.
- Use arrays and simple control flow for basic game logic.

☐ **Basic Error Handling**

- Trim input using trim() to avoid mismatch due to spaces.

- Simple input validation (e.g., ignoring case) to improve user experience.
- Use **methods** to break down code into reusable parts (e.g., `askQuestion()`, `getUserAnswer()`).
- Improves readability, testing, and debugging.

Challenges:

1.Designing the Data Structure

- Choosing between arrays vs. ArrayList.
- Managing questions and answers separately vs. using a class like Flashcard.

2. Handling User Input Correctly

- Input may include leading/trailing spaces or different casing.
- Need to account for partial matches or multiple correct answers (e.g., synonyms).

3. Code Organization

- Writing everything in main() quickly becomes messy.
- Challenge: breaking the game into meaningful methods and classes (modular programming).

4. Managing Game State

- Keeping track of:
 - Which questions have been asked.
 - Correct vs. incorrect answers.
 - Score calculation and result display.

5. Adding File Input

- Reading flashcards from a file (.csv, .txt) can introduce issues:
 - File not found
 - Malformed lines
 - Parsing errors

6. Improving Game Experience

- Making the game interactive (e.g., multiple-choice, retry wrong questions).
- Handling edge cases like:
 - No flashcards loaded
 - Empty inputs
 - Duplicate questions

7. Error and Exception Handling

- Without proper try-catch blocks, the game can crash from unexpected input or file issues.
- Challenge: making error messages user-friendly.

8. Randomization and Fairness

- Shuffling questions with `Collections.shuffle()` while avoiding repetition.
- Ensuring game flow still makes sense with randomized order.

9. Scalability

- As more flashcards are added, memory and performance management might become an issue.
- Sorting, categorizing by topic or difficulty becomes harder to manage manually.

10. Adding a GUI (Swing or JavaFX)

- Moving from console to a graphical interface brings a steep learning curve:
 - Managing layout, user events, and display updates.
 - Handling asynchronous input/output.

Applications:

1.Educational Tools

- Study Aid for Students: Help students memorize vocabulary, formulas, definitions, and concepts.
- Language Learning: Flashcards can help learn new languages (e.g., English-Spanish translations).
- Exam Preparation: Great for standardized test prep (GRE, SAT, IELTS, etc.) using question/answer flashcards.

2. Interview and Job Prep

- Technical Interview Practice: Use flashcards to remember algorithms, data structures, and common coding patterns.
- Behavioral Questions Review: Practice answering STAR-based behavioral interview questions.

3. Corporate Training

- Employee Onboarding: Teach new hires company policies, tools, or procedures via gamified flashcards.
- Skill Refreshers: Continuous learning tools for revisiting important topics or tools (e.g., cybersecurity basics, product info).

4. Gamified Learning Platforms

- Add features like scores, levels, and leaderboards to make learning more engaging.
- Can be part of a larger edtech application or LMS (Learning Management System).

5. Quiz/Trivia Games

- Repurpose the flashcard logic into a quiz app, trivia game, or classroom review tool.
- Add features like multiplayer mode, timers, and power-ups.

6. Memory Training

- Used for cognitive skill development: training attention, retention, and recall using timed challenges.

7. Mobile or Desktop App

- Turn the game into a Java-based Android app using Android Studio.
- Or use JavaFX/Swing to create a desktop GUI flashcard app.

8. Portfolio Project / Resume Builder

- A well-designed flashcard game demonstrates:
 - OOP principles
 - File I/O
 - Data handling
 - GUI design (optional)
 - Real-world problem solving

- Can showcase your Java skills to potential employers.

9. Custom Practice Tools

- Build personalized study tools for specific subjects (e.g., Java concepts, history dates, anatomy).
- Allow others to create and load their own flashcard sets.

10. Open-Source Contribution

- Create and share the flashcard app on GitHub for others to contribute to and use.
- Helps build collaboration and software development experience.

Conclusion:

In conclusion, developing a flashcard game using Java provides a strong foundation in both core programming principles and practical software development skills. The project emphasizes the importance of key Java concepts such as arrays, loops, conditionals, object-oriented programming, and user input handling. By designing and implementing a system that displays questions, captures user responses, and provides immediate feedback, the developer gains hands-on experience with logic structuring and program flow control. Furthermore, the flashcard game can be extended in numerous meaningful ways, such as incorporating a class-based design using objects to represent individual flashcards, reading questions from external files, or implementing more advanced features like randomization, scoring, and categorization. These extensions provide opportunities to explore collections like ArrayList, use Java's exception handling mechanisms, and improve code modularity with reusable methods. Optional enhancements such as a graphical user interface (GUI) using JavaFX or Swing introduce GUI programming concepts, making the application more interactive and visually appealing. This project is not only an engaging way to reinforce knowledge through practice but also serves as a practical educational tool. It can be adapted for various learning purposes including vocabulary building, exam preparation, and even technical interview review. The flashcard game is also a valuable addition to a personal programming portfolio, demonstrating the developer's grasp of both fundamental and intermediate Java topics. Overall, this project blends functionality with educational value and highlights Java's capabilities in building simple yet effective interactive applications.

