Lab 13: Event Handling: Flashcard Study Tool

Jiahao Zhu & Tien Dat Nguyen

December 3, 2025

1 Overview

This lab focuses on developing a simple Flashcard Study Tool using Java Swing. The goal is to reinforce the concepts of **Event Handling** and **File I/O**. The final program loads English–French flashcards from a text file and allows the user to interact with them using mouse clicks and keyboard input.

The following core concepts were demonstrated:

- Reading structured data using Java File I/O
- Handling Swing MouseEvent for flipping the flashcard
- Handling Swing KeyEvent to navigate between cards
- Constructing a GUI using JFrame, JPanel, and JLabel

2 Exercise

Q1: **Flashcard Class:** Create a class Flashcard which stores the English word (front) and the French translation (back).

```
private final String front; private final String
back;
```

Q2: **FlashcardLoader:** Create a class FlashcardLoader to implement file reading to load vocabulary pairs from flashcards_en_fr.txt. Each line in the file is formatted as:

apple=la pomme

The loader should:

- Read data using BufferedReader.
- Split each line on "=".
- Trim extra whitespace.
- Return a list of Flashcards.

```
public static List<Flashcard> loadFromFile(String fileName) throws IOException {}
```

Q3: **FlashcardPanel:** Create a FlashcardPanel class to implement user interaction through event handling. FlashcardsPanel should extend JPanel.

- Mouse click → flip between English/French
  1. Flip card between English (front) and French (back)
  2. Uses a MouseAdaptor to simplify implementation
- LEFT/RIGHT keys → navigate cards
  1. LEFT: Previous card 2. RIGHT: Next card

```
addMouseListener(new MouseAdapter(){});
addKeyListener(new KeyAdapter(){});
```

Q4: **FlashcardApp:** Create a FlashcardApp to integrate all previous components into a running graphical application.

- Create the main window (JFrame)
- Load the cards in the Panel object
- Setup the layout, size, and window properties

3 Tips

Here are some notes and tips on Event Handling (MouseEvent, KeyEvent):

1. **MouseEvent Example:**

```
panel.addMouseListener(new MouseAdapter() {
    // When you click the mouse, the program runs.
    @Override
    public void mouseClicked(MouseEvent e) {
        System.out.println("Mouse clicked!"); }
});
```

2. **KeyEvent Example:**

- Detect the button when you press

```
panel.addKeyListener(new KeyAdapter() {
    // When you press a key, the program detects which one.
    @Override
    public void keyPressed(KeyEvent e) {
        System.out.println("Key pressed: " + e.getKeyCode()); }
});
```

- Detect LEFT/RIGHT Arrow Keys

```
panel.addKeyListener(new KeyAdapter() {
    // Shows how to respond differently depending on the key pressed.
    @Override public void keyPressed(KeyEvent e) { if (e.getKeyCode() == KeyEvent.VK_LEFT)
    System.out.println("LEFT!"); if (e.getKeyCode() == KeyEvent.VK_RIGHT)
    System.out.println("RIGHT!");
    }
});
```

---

4 Event Handling Learning Resources

Below are recommended short tutorials for students to learn MouseEvent and KeyEvent:

- **MouseEvent API Documentation:** https://docs.oracle.com/javase/8/docs/api/java/awt/event/MouseEvent.html
- **KeyEvent API Documentation:** https://docs.oracle.com/javase/8/docs/api/java/awt/event/KeyEvent.html

If you want to include an image, you can do so as follows, see Figure 1.

5 Work Distribution

Include a brief overview of how the team contributed to the assignment.

Jiahao Zhu:

- Do the outline of this lab and research what to do for this lab
- Do the instruction part and do video demo
- Do Junit test

Tien Dat Nguyen:

- Do the code and fixing bug
- Do the AI reflection part

6 AI Reflection

You need to note briefly how you have used any AI tool in the development process, and add a small reflection on how it did / didn't help you learn/review the subject matter needed for the video demo. There isn't really a right or wrong answer here, it's just your opinions / experiences.

In this assignment, we employed an AI tool as an assistant for coding and revision, rather than as a mechanism that merely generated the complete solution. Primarily, we consulted it to elucidate Java concepts that were unclear to us, such as various methods for reading text files or the proper implementation of MouseListener and KeyListener within Swing frameworks. Additionally, we sought its recommendations for a coherent structure of our classes, including Flashcard, FlashcardLoader, FlashcardPanel, and FlashcardApp. Typically, we developed our own initial version before requesting the AI to evaluate it, identify issues, or propose a more refined design, integrating only those elements that we fully comprehended through manual adaptation.

This approach facilitated a more targeted review of the module's material. Rather than navigating extensive documentation to recall minor syntactical details, we posed specific inquiries and received prompt clarifications, thereby allocating more time to contemplate object-oriented design, separation of concerns, and event-handling, the core learning objectives of the assignment. It also compelled a closer examination of error handling and edge cases in file input/output operations, as the AI frequently prompted consideration of scenarios involving invalid lines or absent files.

Nevertheless, our utilization of AI revealed certain risks. On occasion, the recommendations proved unnecessarily intricate for the assignment's level or deviated from the provided skeleton, necessitating filtration and verification against the brief. Furthermore, excessive reliance on generated code diminished our comprehension and complicated debugging efforts. In summary, the AI tool proved beneficial for learning and revision when employed as a tutor and code reviewer, yet it did not supplant the necessity of authoring, testing, and deliberating upon the solution.

Figure 1: Caption