MSCS-632: Assignment 4: Lab Report: Implementing Control Structures

---

Sandesh Pokharel

University of the Cumberlands

MSCS-632 Advanced Programming Languages

Dr. Vanessa Cooper

March 2, 2025

Introduction

Managing employee schedules efficiently is a crucial aspect of any organization. This project implements a command-line Employee Shift Scheduler in two different programming languages: Java and Python. The scheduler ensures proper shift assignment while considering employee preferences and company requirements.

The key objectives of this project are:

- Collecting employee names and their preferred shifts for each day.
- Assigning valid shifts while ensuring no employee works more than one shift per day and a maximum of five days per week.
- Guaranteeing at least two employees per shift.
- Handling conflicts where preferred shifts are unavailable.
- Displaying the final weekly schedule in a structured format.

Programming Languages Chosen

For this assignment, Java and Python were chosen as the two programming languages due to their distinct paradigms:

- Java: A statically typed, object-oriented language with strong enforcement of structured programming.
- Python: A dynamically typed, high-level language known for its simplicity and flexibility.

These two languages contrast in terms of syntax, memory management, and execution model, providing a good comparison of control structures.

Implementation Overview

The application follows the following logic:

Input Collection

- The program prompts the user to enter the number of employees (at least 12 employees for complete shift coverage).
- Each employee is asked for their preferred shifts for each day.
- Employees can either select a shift manually or leave it blank, in which case the program randomly assigns an available shift.

Scheduling Logic

- Validation: Ensures that no employee works more than one shift per day and a maximum of five days per week.
- Conflict Resolution: If an employee selects a shift that is full, they are asked to choose a different shift.
- Random Assignment: If an employee doesn’t provide a preference or if no shifts are available, the system randomly assigns a shift.

Final Output

- After assigning shifts, the program displays the final schedule for the week in a structured format.
- The Java and Python versions produce identical outputs, ensuring consistency across implementations.

Key Features

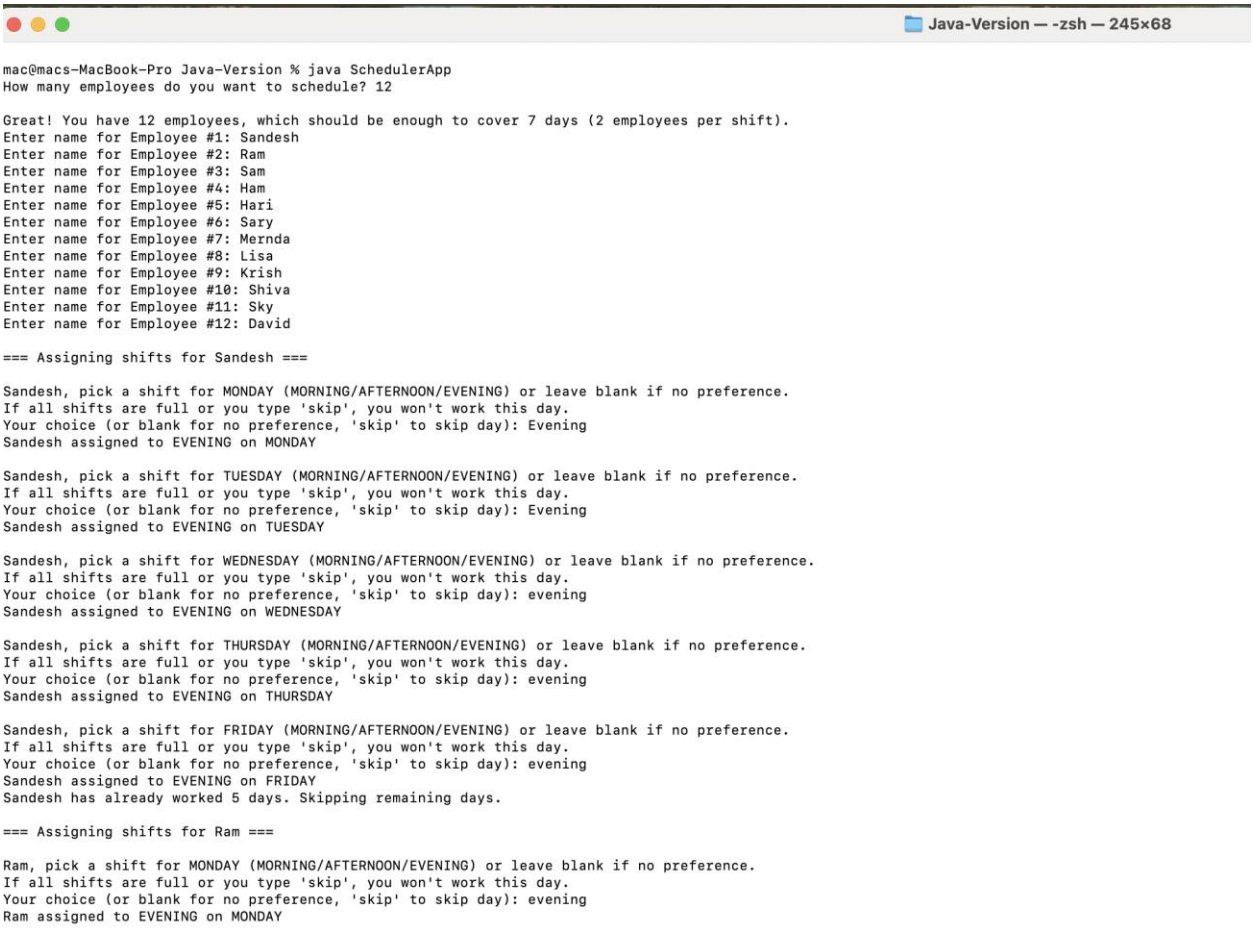
Features	Java Implementation	Python Implementation
Command-Line Interaction	✅ Yes	✅ Yes
Shift Assignment	✅ Yes	✅ Yes
Conflict Resolution	✅ Yes	✅ Yes
Random Shift Assignment	✅ Yes	✅ Yes
Minimum Employees Check	✅ Yes	✅ Yes
Structured Output	✅ Yes	✅ Yes

Both implementations provide the same logic, ensuring consistent functionality across both programming paradigms.

Screenshots

Below are screenshots showing the final schedule output from both the Java and Python implementations.

📌 Java Implementation Output Screenshot

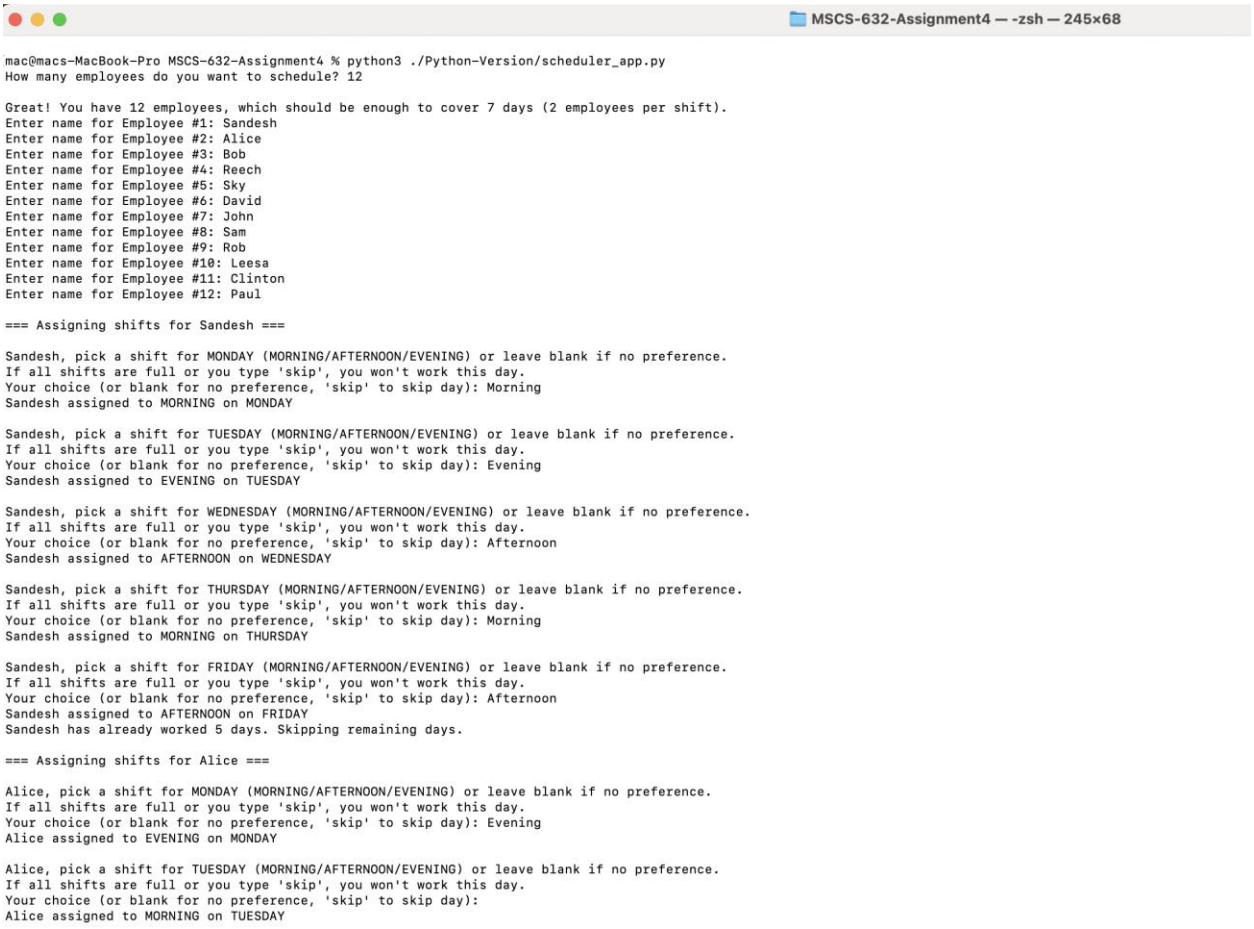

Java-Version — -zsh — 245×68
mac@macs-MacBook-Pro Java-Version % java SchedulerApp
How many employees do you want to schedule? 12
Great! You have 12 employees, which should be enough to cover 7 days (2 employees per shift).
Enter name for Employee #1: Sandesh
Enter name for Employee #2: Ram
Enter name for Employee #3: Sam
Enter name for Employee #4: Ham
Enter name for Employee #5: Hari
Enter name for Employee #6: Sary
Enter name for Employee #7: Mernda
Enter name for Employee #8: Lisa
Enter name for Employee #9: Krish
Enter name for Employee #10: Shiva
Enter name for Employee #11: Sky
Enter name for Employee #12: David
=== Assigning shifts for Sandesh ===
Sandesh, pick a shift for MONDAY (MORNING/AFTERNOON/EVENING) or leave blank if no preference.
If all shifts are full or you type 'skip', you won't work this day.
Your choice (or blank for no preference, 'skip' to skip day): Evening
Sandesh assigned to EVENING on MONDAY
Sandesh, pick a shift for TUESDAY (MORNING/AFTERNOON/EVENING) or leave blank if no preference.
If all shifts are full or you type 'skip', you won't work this day.
Your choice (or blank for no preference, 'skip' to skip day): Evening
Sandesh assigned to EVENING on TUESDAY
Sandesh, pick a shift for WEDNESDAY (MORNING/AFTERNOON/EVENING) or leave blank if no preference.
If all shifts are full or you type 'skip', you won't work this day.
Your choice (or blank for no preference, 'skip' to skip day): evening
Sandesh assigned to EVENING on WEDNESDAY
Sandesh, pick a shift for THURSDAY (MORNING/AFTERNOON/EVENING) or leave blank if no preference.
If all shifts are full or you type 'skip', you won't work this day.
Your choice (or blank for no preference, 'skip' to skip day): evening
Sandesh assigned to EVENING on THURSDAY
Sandesh, pick a shift for FRIDAY (MORNING/AFTERNOON/EVENING) or leave blank if no preference.
If all shifts are full or you type 'skip', you won't work this day.
Your choice (or blank for no preference, 'skip' to skip day): evening
Sandesh assigned to EVENING on FRIDAY
Sandesh has already worked 5 days. Skipping remaining days.
=== Assigning shifts for Ram ===
Ram, pick a shift for MONDAY (MORNING/AFTERNOON/EVENING) or leave blank if no preference.
If all shifts are full or you type 'skip', you won't work this day.
Your choice (or blank for no preference, 'skip' to skip day): evening
Ram assigned to EVENING on MONDAY

```
===== FINAL WEEKLY SCHEDULE =====

--- MONDAY ---
  MORNING: Sam Sary
  AFTERNOON: Ham Hari
  EVENING: Sandesh Ram

--- TUESDAY ---
  MORNING: Sam Sary
  AFTERNOON: Ham Hari
  EVENING: Sandesh Ram

--- WEDNESDAY ---
  MORNING: Hari Sary
  AFTERNOON: Sam Ham
  EVENING: Sandesh Ram

--- THURSDAY ---
  MORNING: Hari Sary
  AFTERNOON: Sam Ham
  EVENING: Sandesh Ram

--- FRIDAY ---
  MORNING: Sam Hari
  AFTERNOON: Ham Sary
  EVENING: Sandesh Ram

--- SATURDAY ---
  MORNING: Lisa Krish
  AFTERNOON: Mernda Shiva
  EVENING: Sky David

--- SUNDAY ---
  MORNING: Lisa Sky
  AFTERNOON: Mernda David
  EVENING: Krish Shiva
mac@macs-MacBook-Pro Java-Version %
```

In case shift is full:

```
=== Assigning shifts for Sam ===

Sam, pick a shift for MONDAY (MORNING/AFTERNOON/EVENING) or leave blank if no preference.
If all shifts are full or you type 'skip', you won't work this day.
Your choice (or blank for no preference, 'skip' to skip day): evening
That shift is already full. Pick another or skip.
Your choice (or blank for no preference, 'skip' to skip day): Morning
Sam assigned to MORNING on MONDAY
```

📌 Python Implementation Output Screenshot



```
===== FINAL WEEKLY SCHEDULE =====

--- MONDAY ---
  MORNING: Sandesh Reech
  AFTERNOON: Sky David
  EVENING: Alice Bob

--- TUESDAY ---
  MORNING: Alice Bob
  AFTERNOON: Sky David
  EVENING: Sandesh Reech

--- WEDNESDAY ---
  MORNING: Reech Sky
  AFTERNOON: Sandesh David
  EVENING: Alice Bob

--- THURSDAY ---
  MORNING: Sandesh Alice
  AFTERNOON: Bob David
  EVENING: Reech Sky

--- FRIDAY ---
  MORNING: Reech David
  AFTERNOON: Sandesh Sky
  EVENING: Alice Bob

--- SATURDAY ---
  MORNING: Sam Clinton
  AFTERNOON: Rob Paul
  EVENING: John Leesa

--- SUNDAY ---
  MORNING: John Paul
  AFTERNOON: Rob Leesa
  EVENING: Sam Clinton
mac@macs-MacBook-Pro MSCS-632-Assignment4 %
mac@macs-MacBook-Pro MSCS-632-Assignment4 %
```

Source Code Repository

The complete source code for both implementations is available in a public GitHub repository.

📌 GitHub Repository Link:

🔗 **https://github.com/sanspokharel26677/MSCS-632-Assignment4/tree/main**

Conclusion

This project successfully demonstrates control structures and logic flow in two different programming languages. The Java implementation showcases structured programming with object-oriented principles, while the Python implementation leverages dynamic typing and concise syntax. Both versions adhere to the same scheduling logic, ensuring correctness and reliability.

This assignment effectively highlights how different programming paradigms can implement the same logic in different ways, reinforcing the importance of control structures in software development.

References

- Python Official Documentation: https://docs.python.org/3/
- Java Official Documentation: https://docs.oracle.com/en/java/