Homework #3

CS 4120/5220 – Adv Appl Devlpmnt in Java

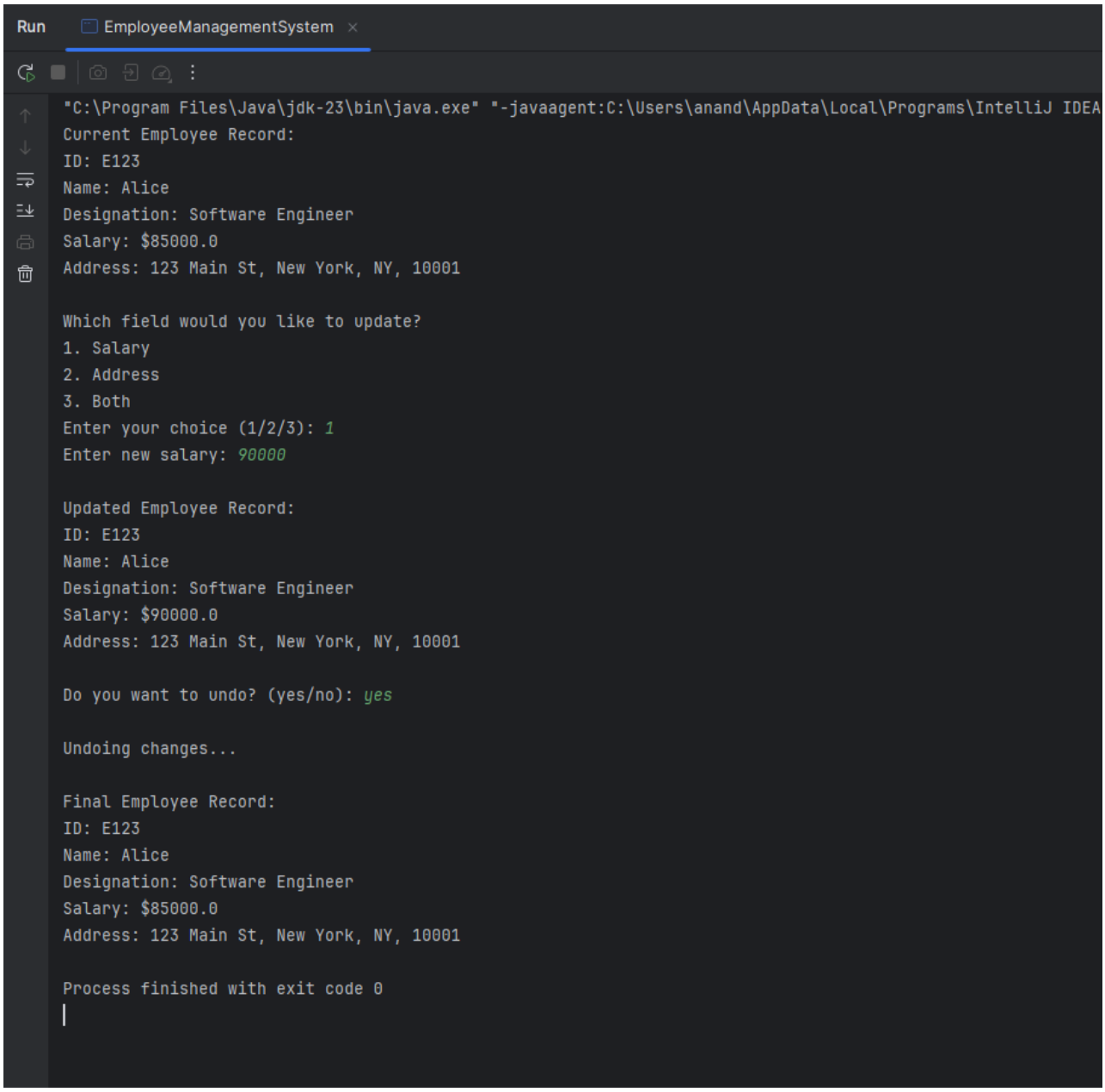
Fall 2025

100 Points
Due: 10/03/2025, 11:59 PM

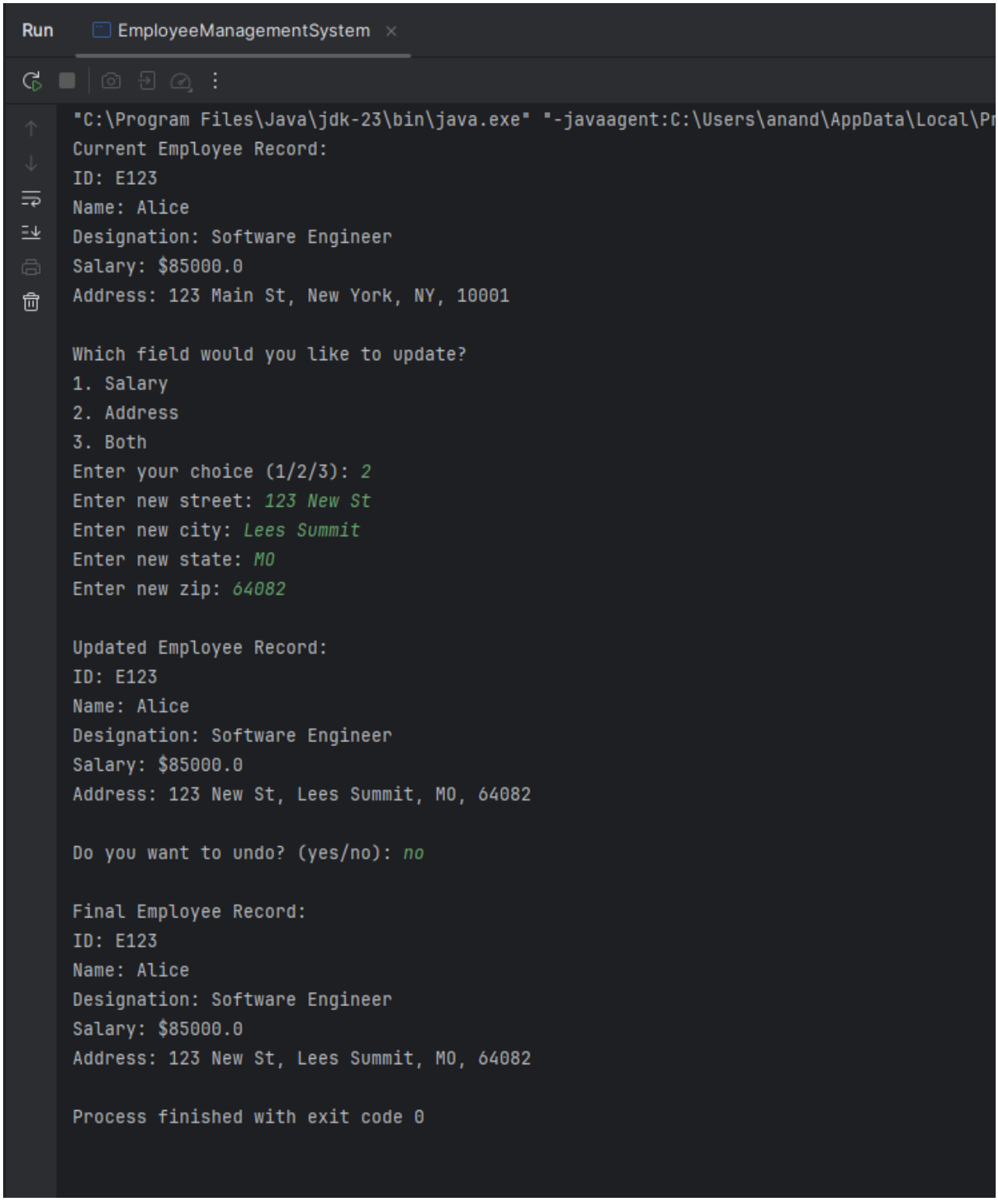
Instructions: Submit all required project files, including source code, as a single compressed archive (ZIP file). Ensure that your project is well-organized, with appropriate folder structures for source files, resources, and any required dependencies. Include a README file if necessary to explain how to run the program. Do not copy and paste code from external sources. Clearly comment on your code to explain important logic. Convert any written explanations, analysis, or required screenshots into a single PDF file and include it in the ZIP archive. Upload the ZIP file to Brightspace before the due date and time.

1. Create a program to simulate Employee Record Management with Undo Feature. **(35 points)**
    a. Create an Address class with the following attributes:
        i. street (String)
        ii. city (String)
        iii. state (String)
        iv. zip (String)
    b. Create an Employee class with the following attributes.
        i. id (String)
        ii. name (String)
        iii. designation (String)
        iv. salary (double)
        v. address (Address)
    c. Implement deep cloning so that changes to an employee's profile can be undone, including address updates.
    d. Implementation in main():
        i. Display the current employee record.
        ii. Allow the user to edit employee details by choice, including salary and address.
        iii. Before applying changes, store a deep clone of the original record as backup.
        iv. If the user chooses to undo, restore the last saved version.

Sample execution output:

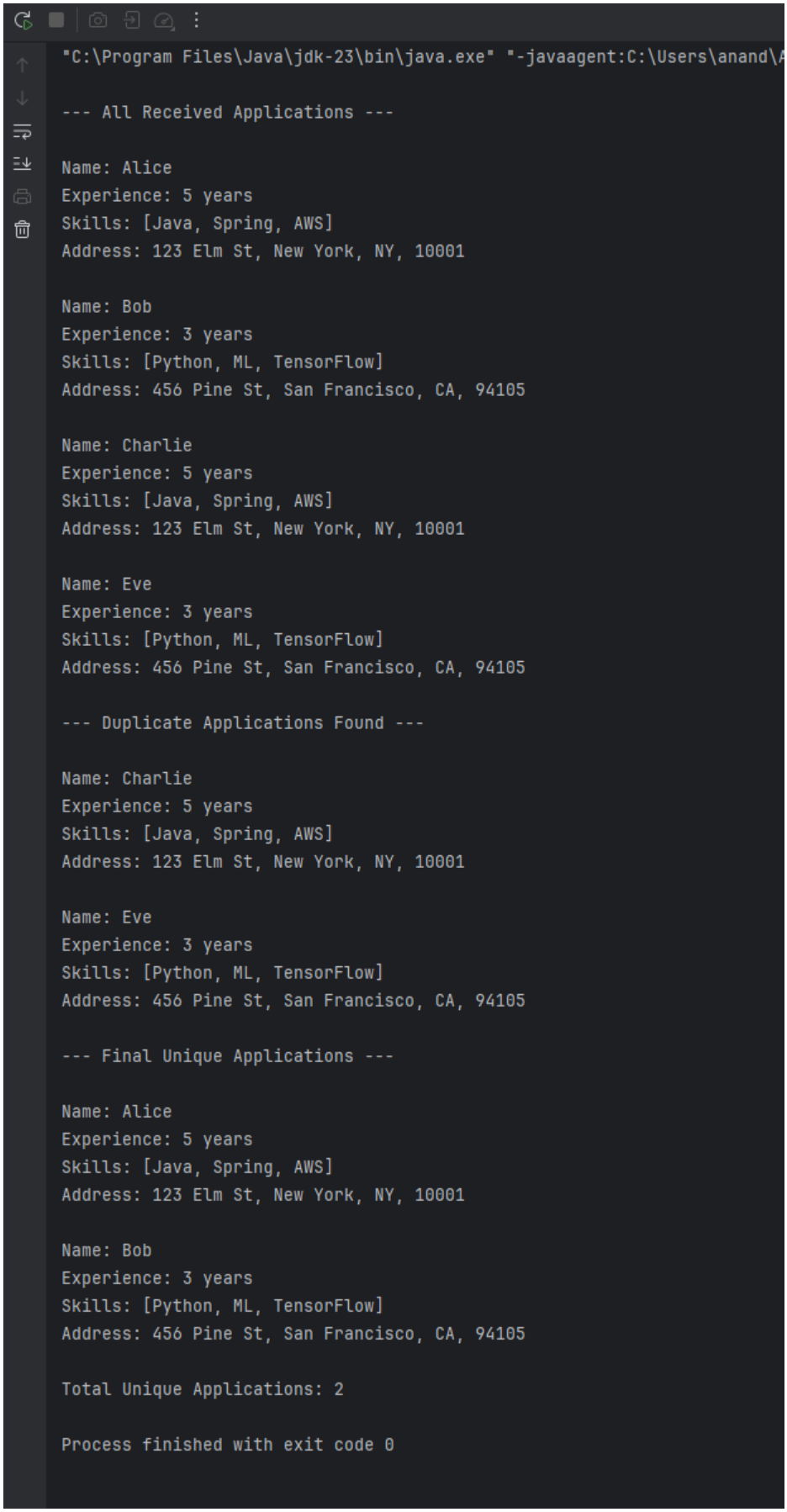


```
"C:\Program Files\Java\jdk-23\bin\java.exe" "-javaagent:C:\Users\anand\AppData\Loc
Current Employee Record:
ID: E123
Name: Alice
Designation: Software Engineer
Salary: $85000.0
Address: 123 Main St, New York, NY, 10001

Which field would you like to update?
1. Salary
2. Address
3. Both
Enter your choice (1/2/3): 1
Enter new salary: 90000

Updated Employee Record:
ID: E123
Name: Alice
Designation: Software Engineer
Salary: $90000.0
Address: 123 Main St, New York, NY, 10001

Do you want to undo? (yes/no): no

Final Employee Record:
ID: E123
Name: Alice
Designation: Software Engineer
Salary: $90000.0
Address: 123 Main St, New York, NY, 10001

Process finished with exit code 0
```


Run
EmployeeManagementSystem
"C:\Program Files\Java\jdk-23\bin\java.exe" "-javaagent:C:\Users\anand\AppData\Local\Pr
Current Employee Record:
ID: E123
Name: Alice
Designation: Software Engineer
Salary: $85000.0
Address: 123 Main St, New York, NY, 10001
Which field would you like to update?
1. Salary
2. Address
3. Both
Enter your choice (1/2/3): 2
Enter new street: 123 New St
Enter new city: Lees Summit
Enter new state: MO
Enter new zip: 64082
Updated Employee Record:
ID: E123
Name: Alice
Designation: Software Engineer
Salary: $85000.0
Address: 123 New St, Lees Summit, MO, 64082
Do you want to undo? (yes/no): no
Final Employee Record:
ID: E123
Name: Alice
Designation: Software Engineer
Salary: $85000.0
Address: 123 New St, Lees Summit, MO, 64082
Process finished with exit code 0

2. Create a Resume Filtering System identifying similar applications and prevent redundant submissions: **(35 points)**
    a. Create an Address class with the following attributes:
        i. street (String)
        ii. city (String)
        iii. state (String)
        iv. zip (String)
    b. Create a Resume class with the following attributes:
        i. candidateName (String)
        ii. yearsOfExperience (int)
        iii. skills (ArrayList<String>)
    c. Override the following methods:
        i. toString() – Displays resume details in a readable format.
        ii. equals() – Two resumes are equal if they have the same years of experience, skills, and address (ignore candidates name).
    d. Implementation in main():
        i. Store multiple resumes in ArrayList.
        ii. Display all the resumes.
        iii. Check for duplicates and display duplicate applications.
        iv. Print a warning if an application is already submitted.
        v. Display final unique applications.

Sample execution output:


"C:\Program Files\Java\jdk-23\bin\java.exe" "-javaagent:C:\Users\anand\A
--- All Received Applications ---
Name: Alice
Experience: 5 years
Skills: [Java, Spring, AWS]
Address: 123 Elm St, New York, NY, 10001
Name: Bob
Experience: 3 years
Skills: [Python, ML, TensorFlow]
Address: 456 Pine St, San Francisco, CA, 94105
Name: Charlie
Experience: 5 years
Skills: [Java, Spring, AWS]
Address: 123 Elm St, New York, NY, 10001
Name: Eve
Experience: 3 years
Skills: [Python, ML, TensorFlow]
Address: 456 Pine St, San Francisco, CA, 94105
--- Duplicate Applications Found ---
Name: Charlie
Experience: 5 years
Skills: [Java, Spring, AWS]
Address: 123 Elm St, New York, NY, 10001
Name: Eve
Experience: 3 years
Skills: [Python, ML, TensorFlow]
Address: 456 Pine St, San Francisco, CA, 94105
--- Final Unique Applications ---
Name: Alice
Experience: 5 years
Skills: [Java, Spring, AWS]
Address: 123 Elm St, New York, NY, 10001
Name: Bob
Experience: 3 years
Skills: [Python, ML, TensorFlow]
Address: 456 Pine St, San Francisco, CA, 94105
Total Unique Applications: 2
Process finished with exit code 0

3. Implement a Generic Min-Max finder that determines the minimum and maximum elements from an array of any comparable type. **(30 points)**
    a. Create a generic class MinMaxFinder<T extends Comparable<T>> with:
        i. A constructor that takes an array of elements.
        ii. A method T findMin() that returns the smallest element.
        iii. A method T findMax() that return the largest element.
    b. Demonstrate the MinMaxFinder<T> class in main():
        i. Create instances with different data types:
        ii. Integer Array → [5, 2, 9, 1, 7]
        iii. Double Array → [3.5, 7.2, 1.8, 9.0]
        iv. String Array → ["Apple", "Mango", "Banana", "Cherry"] (Lexicographically sorted)
        v. Find and display the min and max values for each dataset.

Sample Output: