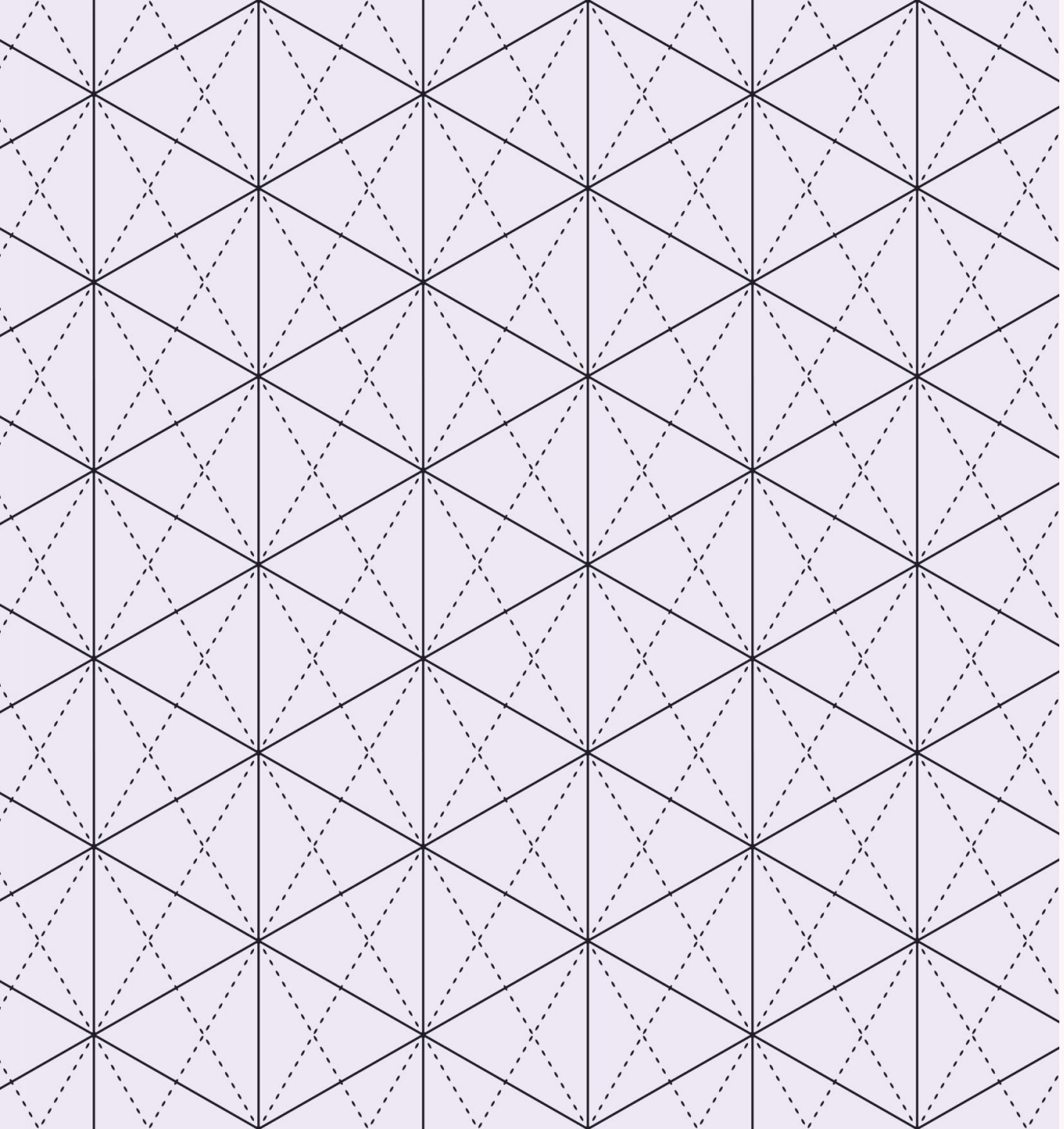


Sahel Assadi

Data Structures and Object-Oriented Programming

Online Banking System

Outline/Table of Contents

Project Description

- The Online Banking System is an online application in which customers can view account details, such as account type, balance, and transactions. Customers can also review chequing and savings accounts transactions, including deposit and withdrawal amounts with corresponding dates. As for employees, they get the right to open and close customer accounts, view their transactions and check their account balance.

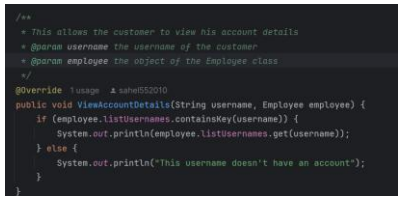
Program features and Screenshots

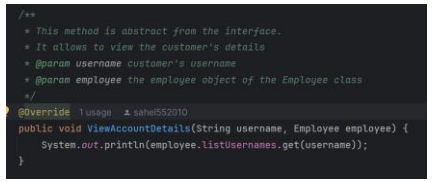
- The project consists of two hierarchies : the Customer and the Employee

```
public class Employee
```

```
public class Customer
```

Program features and Screenshots (con't)





- The project has one user-defined interface with one abstract method in it. That interface (CustomerDetails) has a void ViewAccountDetails (with parameters String username and Employee employee) that enables for both the Customer and the Employee to view the customer's details.

```java
/**
 * This method allows to view Transaction history
 * @param bankAccount the object of the BankAccount class
 */
public void TransactionHistory(BankAccount bankAccount) {  6 usages  2 overrides  sahel552010
    Collections.sort(bankAccount.transaction);
    System.out.println(bankAccount.transaction);
}
```

Program features and Screenshots (con't)

- There is one runtime-polymorphism in the project. The method that is used for polymorphism is the void TransactionHistory. The one for the BankAccount class is used to view the transaction of the bank account as a whole. For its subclasses (Chequing and Savings), it is used to view the transactions but especially from those type of accounts.

```java
/**
 * exports openCustomer and closedCustomer file paths
 */
public static void export() {  2 usages  sahel552010
    writeCustomers(OPEN_CUSTOMER_FILE_PATH, openedCustomer);
    writeCustomers(CLOSED_CUSTOMER_FILE_PATH, closedCustomer);
}

/**
 * This method writes to the csv file
 * @param path the file path
 * @param customers the customers list
 */
private static void writeCustomers(String path, Collection<Customer> customers) {  2 usages  sahel552010
    File file = new File(path);
    try (FileWriter fw = new FileWriter(file)) {
        for (Customer customer : customers) {
            fw.write( str: customer.getName() + ",");
            fw.write( str: customer.getUsername() + ",");
            fw.write( str: customer.getBankAccount() + ",");
            fw.write( str: customer.getSin() + ",");
            fw.write( str: customer.getDateOfBirth() + ",");
            fw.write( str: customer.getEmailAddress() + ",");
            fw.write(customer.getCreditScore());
        }
    } catch (IOException e) {
        throw new RuntimeException(e);
    }
/**
 * this method reads the customers from the csv file
 * @param path the file path
 * @return the customers list
 */
private static List<Customer> readCustomer(String path) {  2 usages  sahel552010
    List<Customer> customers = new ArrayList<>();

    File file = new File(path);
    int nextId;
    try (Scanner scanner = new Scanner(file)) {
        while (scanner.hasNextLine()) {
            String line = scanner.nextLine();
            String[] parts = line.split( regex: ",");
            String name = parts[0];
            String username = parts[1];
            BankAccount bankAccount = new BankAccount();
            int sin = Integer.parseInt(parts[3]);
            String dateOfBirth = parts[4];
            String emailAddress = parts[5];
            int creditScore = Integer.parseInt(parts[6]);


            Customer customer = new Customer(name, username, bankAccount, sin, dateOfBirth, emailAddress, creditScore);
            customers.add(customer);
        }

    } catch (FileNotFoundException e) {
        throw new RuntimeException(e);
    }
    return customers;
}
```

Program features and Screenshots (con't)

- The text I/O is used to read and write to file. In this project, the reading is used to read in the file the customers (their account details) that have already been written. As for the writing, it is used to write the customers details into the file(when a new one opens from the employee). It can also remove the customer (when an existing one closes).

Program features and Screenshots (con't)

```java
@Override  sahel552010
public int compareTo(Transaction o) {
    return Double.compare(this.amount, o.amount);
}
```

```java
@Override  sahel552010
public int compare(Customer o1, Customer o2) {
    return (o1.name.compareTo(o2.name)) * 10000
            + o1.creditScore - o2.creditScore;
}
```

- The Comparable is implemented in the Transaction class. It is used to sort the transactions from the smallest amount to the highest amount that has been transacted. As for the comparator, it is implemented in the customer class. It is used to sort the customers details first by name, and then secondly by credit score. Also, Unit testing wasn't used because all methods were void (as said in the requirement, if applicable).

Challenges

- I have faced some challenges throughout the development of the project. I had some difficulties with the text I/O as of how to do the reading and the writing. Also, I had a hard time finding some more classes and options for the Employee class. I also want to mention that one of my biggest difficulties was to think properly and use my time wisely for the project. I admit that the project could not look perfect, but at least I have tried my best respecting the requirements. I think what I can improve for my next project would be to use a good time management and to practice and understand a lot more. Taking the right time to do the project.

Learning outcomes

- Aside the challenges, I have also learned some amazing things from the project. I understood how to properly use the classes and when to use the inheritance (for example, BankAccount cannot be a subclass of Customer class, but it can be a field member of the Customer class, since Customer has a BankAccount). I also understood more about the use of Text I/O, especially for this project. I now know how to use GitHub, which will help me next time for my future studies in programming. This course has definitly helped me learn more about programming.