OASIS INFOBYTE INTERNSHIP DOCUMENTATION

## INTERN NAME: K.AKSHAYA

## DOMAIN: JAVA DEVELOPMENT

## TASK NAME: ATM INTERFACE

## DURATION: 5th OCTOBER - 15th NOVEMBER

# Introduction

This document presents the ATM Interface project completed during my Java Development Internship at Oasis Infobyte.

The project demonstrates the implementation of a console-based ATM system with features such as account creation, login, deposit, withdrawal, transfer, and transaction history.

# Objective

To create an interactive ATM system in Java that allows multiple users to perform basic banking operations securely and efficiently.

# Tools and Technologies

**Language:** Java

**IDE:** VS Code

**Core Concepts Used:** Classes, Objects, HashMaps, ArrayLists, Scanner class, Loops, Conditional Statements

# Steps Performed

1.Designed **ATM** and **User** classes to manage account operations and user details.

2.Implemented **account creation** with username and PIN verification.

3.Built **login functionality** to authenticate users.

4.Added **ATM operations**: deposit, withdraw, check balance, transfer funds, and view transaction history.

5.Stored all **transaction history** in an ArrayList for each user.

6.Created a **menu-driven console interface** for easy user interaction.

# Code Implementation

import java.util.Scanner;

import java.util.HashMap;

import java.util.ArrayList;

class ATM {

private double balance;

private ArrayList<String> transactions;

public ATM() {

balance = 0.0;

transactions = new ArrayList<>();

}

public void deposit(double amount) {

if (amount <= 0) {

System.out.println("Invalid amount.");

return;

}

balance += amount;

transactions.add("Deposited: $" + amount);

System.out.println("Deposited $" + amount + " successfully.");

}

public void withdraw(double amount) {

if (amount <= 0) {

System.out.println("Invalid amount.");

} else if (amount > balance) {

System.out.println("Insufficient balance.");

} else {

balance -= amount;

transactions.add("Withdrew: $" + amount);

System.out.println("Withdrawn $" + amount + " successfully.");

}

}

public void transfer(double amount, ATM receiver) {

if (amount <= 0) {

System.out.println("Invalid amount.");

} else if (amount > balance) {

System.out.println("Insufficient balance to transfer.");

} else {

balance -= amount;

receiver.balance += amount;

transactions.add("Transferred $" + amount + " to another account.");

System.out.println("Transferred $" + amount + " successfully.");

}

}

public void showTransactions() {

if (transactions.isEmpty()) {

System.out.println("No transactions yet.");

} else {

System.out.println("Transaction History:");

for (String t : transactions) {

System.out.println(t);

}

}

}

public void checkBalance() {

System.out.println("Current Balance: $" + balance);

}

}

class User {

String username;

String pin;

ATM account;

public User(String username, String pin) {

this.username = username;

this.pin = pin;

this.account = new ATM();

}

}

public class ATMInterface {

private static final Scanner sc = new Scanner(System.in);

private static final HashMap<String, User> users = new HashMap<>();

public static void main(String[] args) {

System.out.println("Welcome to the Enhanced ATM Interface!");

boolean running = true;

while (running) {

System.out.println("\n1. Create Account");

System.out.println("2. Login");

System.out.println("3. Quit");

System.out.print("Enter choice: ");

int choice = sc.nextInt();

sc.nextLine(); // consume newline

switch (choice) {

case 1 -> createAccount();

case 2 -> login();

case 3 -> {

System.out.println("Goodbye!");

running = false;

}

default -> System.out.println("Invalid option!");

}

}

}

private static void createAccount() {

System.out.print("Enter a username: ");

String username = sc.nextLine();

if (users.containsKey(username)) {

System.out.println("Username already exists!");

return;

}

System.out.print("Set a 4-digit PIN: ");

String pin = sc.nextLine();

users.put(username, new User(username, pin));

System.out.println("Account created successfully!");

}

private static void login() {

System.out.print("Enter username: ");

String username = sc.nextLine();

if (!users.containsKey(username)) {

System.out.println("User not found!");

return;

}

System.out.print("Enter PIN: ");

String pin = sc.nextLine();

User user = users.get(username);

if (!user.pin.equals(pin)) {

System.out.println("Incorrect PIN!");

return;

}

System.out.println("Login successful! Welcome " + username);

userMenu(user);

}

private static void userMenu(User user) {

boolean loggedIn = true;

while (loggedIn) {

System.out.println("\n1. Check Balance");

System.out.println("2. Deposit");

System.out.println("3. Withdraw");

System.out.println("4. Transfer");

System.out.println("5. Transaction History");

System.out.println("6. Logout");

System.out.print("Enter choice: ");

int choice = sc.nextInt();

sc.nextLine(); // consume newline

switch (choice) {

case 1 -> user.account.checkBalance();

case 2 -> {

System.out.print("Enter amount to deposit: $");

double amt = sc.nextDouble();

user.account.deposit(amt);

}

case 3 -> {

System.out.print("Enter amount to withdraw: $");

double amt = sc.nextDouble();

user.account.withdraw(amt);

}

case 4 -> {

System.out.print("Enter recipient username: ");

String receiverName = sc.nextLine();

if (!users.containsKey(receiverName)) {

System.out.println("Recipient not found!");

break;

}

System.out.print("Enter amount to transfer: $");

double amt = sc.nextDouble();

user.account.transfer(amt, users.get(receiverName).account);

}

case 5 -> user.account.showTransactions();

case 6 -> {

System.out.println("Logged out successfully.");

loggedIn = false;

}

default -> System.out.println("Invalid option!");

}

}

}

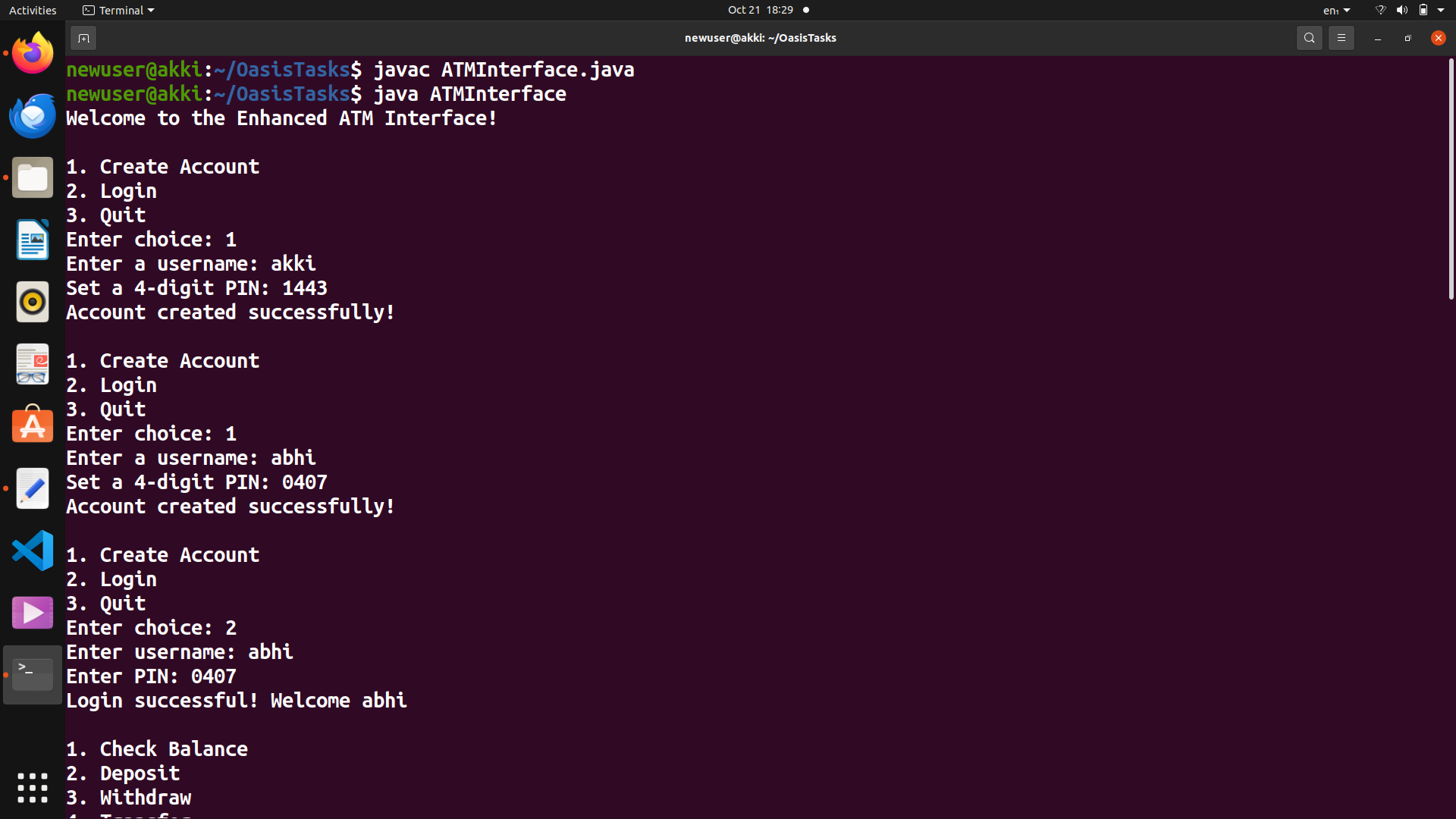
}

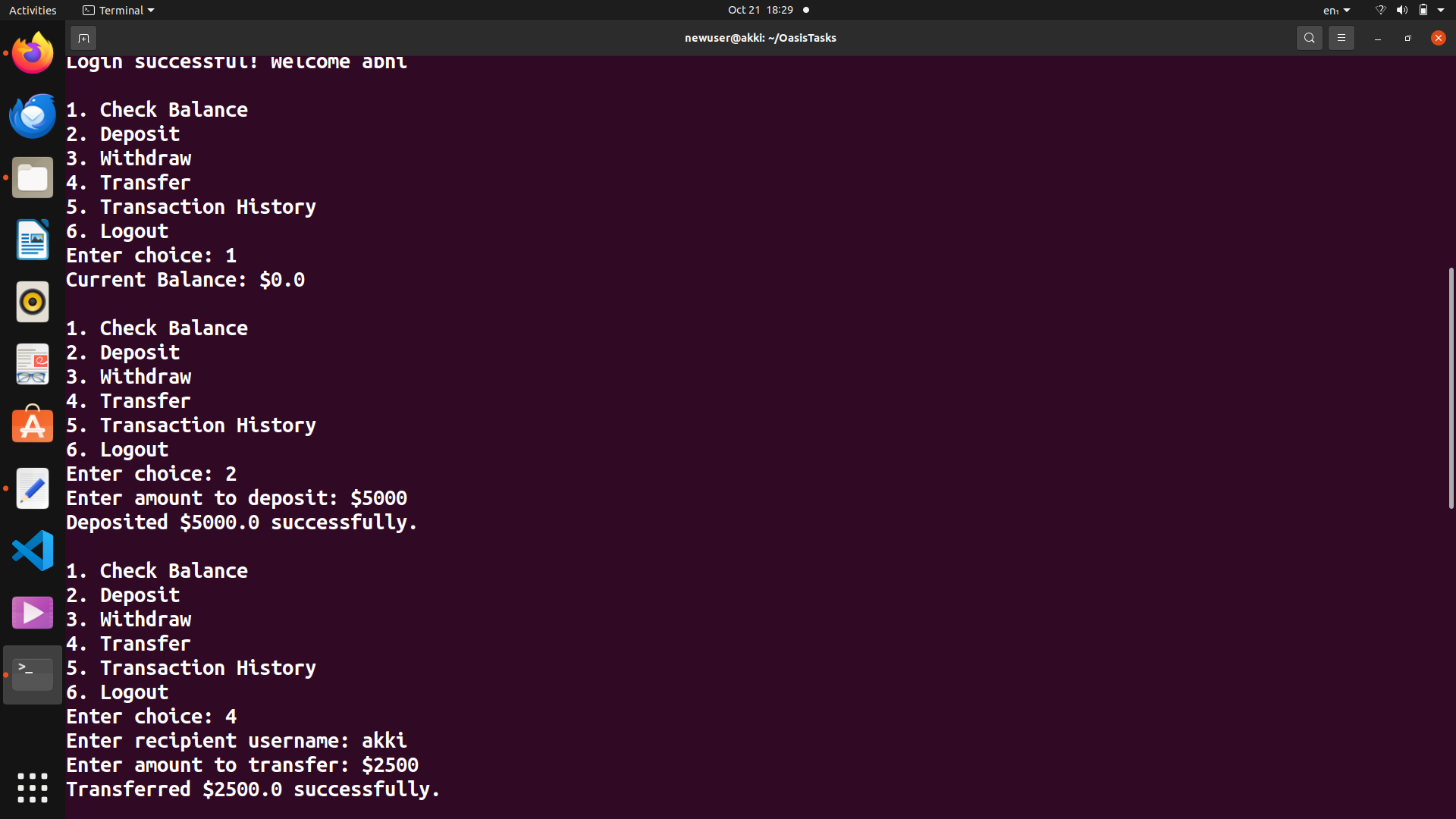
# Output

->Users can create accounts, login, and perform transactions.

->The system provides feedback for invalid inputs, insufficient balance, and transaction success messages.

->Transaction history logs all deposits, withdrawals, and transfers.





### **Conclusion**

This project enhanced my understanding of Java OOP concepts, data structures, and user input handling. I gained hands-on experience in building a functional console application simulating real-world banking operations.