Credit Name: Chapter 8

Assignment Name: Account, PersonalAcct, BusinessAcct

Describe the errors you've encountered while working on this assignment. What caused the error and how do you overcome the error?

1. Missing Method Implementation

```
public abstract class Account {
    public abstract void withdraw(double amount);
}

public class PersonalAcct extends Account {

}
```

```
public class PersonalAcct extends Account {
    @Override
    public void withdraw(double amount) {
        if (balance - amount < 100) {
            balance -= 2;
        }
        balance -= amount;
    }
}
```

Error: The PersonalAcct and BusinessAcct subclasses did not implement the withdraw() method from the abstract Account class.

Cause: The withdraw() method in Account was declared as abstract, and all subclasses must implement it.

Fix: Added the withdraw() method in both PersonalAcct and BusinessAcct classes with specific rules for penalties based on balance thresholds.

2. Incorrect Penalty Logic

```
if (balance < 100) {
    balance -= 2;
}
balance -= amount;
```

```
if (balance - amount < 100) {
    balance -= 2;
}
balance -= amount;
```

Error: In PersonalAcct, the penalty for falling below $100 was applied even if the balance didn't actually fall below $100 after withdrawal.

Cause: The penalty was deducted before validating if the balance fell below the threshold.

Fix: Updated the logic to apply the penalty only if the balance actually falls below $100:

3. Misusing super() in Subclass Constructor

```
public class PersonalAcct extends Account {
    public PersonalAcct(double balance) {
        this.balance = balance;
    }
}
```

```
public class PersonalAcct extends Account {
    public PersonalAcct(double balance) {
        super(balance);
    }
}
```

Error: The constructors for PersonalAcct and BusinessAcct did not call the superclass constructor, causing a compilation error.

Cause: The Account class constructor required a balance parameter, and the subclasses didn't pass it.

Fix: Added super(balance) in the constructors of both subclasses:

public PersonalAcct(double balance) {

    super(balance);

4. Input Validation for Deposit and Withdrawal

```
public void deposit(double amount) {
    balance += amount;
```

```java
public void deposit(double amount) {
    if (amount < 0) {
        System.out.println("Amount cannot be negative.");
        return;
    }
    balance += amount;
}
```

Error: The program allowed negative amounts for deposits and withdrawals, causing incorrect balance calculations.

Cause: No validation was in place to check for negative input values.

Fix: Added a check to ensure deposit and withdrawal amounts are positive:

5. Scanner Buffer Issue

```java
int choice = input.nextInt();
String action = input.next();
```

```java
int choice = input.nextInt();
input.nextLine();
String action = input.next();
```

Error: Switching between nextInt() and nextLine() caused the program to skip user input.

Cause: nextInt() left a newline character in the input buffer, which was read by nextLine().

Fix: Added a input.nextLine() after nextInt() to clear the buffer:

int choice = input.nextInt();

input.nextLine(); // Clear buffer