MODULE 3

SECTION 2.1

2. Explore JavaBank. Record your observations. What happens when you: • Display Accounts • Create Accounts • Delete Accounts • Make a Withdrawal Transaction • Make a Deposit Transaction • Can you display accounts before any are created? • Can you create an account without entering anything in the fields? • Can you make a withdrawal transaction with no amount in the Withdrawal field? • Can you do a deposit transaction with no amount in the Deposit field? • What other questions do you have about the JavaBank application? Copyright © 2020, Oracle and/or its affiliates. All rights reserved. Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners. 2 • What changes would you make to the current application to make it function better? • What additions would you make to the current application to increase its functionality?

ANSWER:

**1. Display Accounts**

* **Expected Behavior**: If accounts have been created, they should be listed. If no accounts exist, a message like "No accounts to display" might appear.
* **Key Questions**: Can you display accounts before any are created? The system should ideally handle this scenario by showing a message that no accounts exist.

**2. Create Accounts**

* **Expected Behavior**: You should be able to create a new account by entering required details like account name, type, initial deposit, etc. The system should validate the input fields.
* **Key Questions**:
  + Can you create an account without entering anything in the fields? Ideally, the application should not allow this and should prompt the user to fill in required fields.

**3. Delete Accounts**

* **Expected Behavior**: You should be able to delete an account if it exists. The application might ask for confirmation before deletion.
* **Key Questions**: What happens if you try to delete an account that doesn’t exist? The application should handle this gracefully, possibly with a message indicating the account doesn’t exist.

**4. Make a Withdrawal Transaction**

* **Expected Behavior**: The application should allow you to withdraw an amount from an account, provided there are sufficient funds.
* **Key Questions**:
  + Can you make a withdrawal with no amount in the Withdrawal field? The application should prevent this and prompt the user to enter an amount.
  + Can you withdraw more than the balance? Ideally, the application should reject the transaction and display an error.

**5. Make a Deposit Transaction**

* **Expected Behavior**: You should be able to deposit money into an account.
* **Key Questions**:
  + Can you do a deposit with no amount in the Deposit field? Similar to withdrawal, the application should prompt the user to enter an amount.
  + Can you deposit a negative amount? The application should reject this with an appropriate error message.

**Other Questions About JavaBank:**

* **What is the maximum number of accounts you can create?**
* **Does the application support different types of accounts (e.g., savings, checking)?**
* **How does the application handle transactions involving multiple currencies?**
* **Is there a way to view a transaction history for each account?**
* **How does the application handle concurrency? What if two users try to access or modify the same account simultaneously?**

**Suggested Changes to Improve the Application:**

* **Input Validation**: Ensure that all fields are validated, including checks for empty fields, correct data types, and sensible ranges for numeric inputs.
* **Error Handling**: Improve error messages to be more descriptive, guiding users to correct their mistakes.
* **User Interface**: Enhance the UI/UX to make navigation more intuitive, possibly with clearer labels, help tooltips, and better feedback on user actions.
* **Security Enhancements**: Implement more robust security measures, such as password protection, encryption for sensitive data, and secure transaction logging.

**Potential Additions to Increase Functionality:**

* **Transaction History**: Add a feature to view detailed transaction history for each account.
* **Account Summary**: Provide a dashboard with an overview of all accounts, including balances, recent transactions, and account types.
* **Multi-Currency Support**: Allow transactions and accounts in different currencies, with real-time exchange rates.
* **Loan and Credit Features**: Introduce functionality for managing loans, credit lines, and associated repayments.
* **Mobile Support**: Develop a mobile version or make the current application responsive for better usability on smartphones and tablets.

These changes and additions would help in making the application more user-friendly and versatile, expanding its use cases and improving user satisfaction.

3. Import the file bikeproject.zip to a location on your computer and answer the following questions. a. Give an example of a primitive data type that is used to store fields within a class. b. Give an example of where String concatenation takes place. c. What are the names of the objects created in this program? d. How many constructors does each class have? e. Inheritance is part of this program. Identify the Super and subclasses from this program. f. Mountain bikes and road bikes can be constructer either by using the default values (standard bike) or customized to the client’s needs. Using the following table identify sample values assigned to one of each type of standard bike:

ANSWER:

a. private int gearCount;

b. String bikeDescription = "Bike Type: " + type + ", Gear Count: " + gearCount;

c. MountainBike myMountainBike = new MountainBike();

RoadBike myRoadBike = new RoadBike();

d. public MountainBike() {}

public MountainBike(int gearCount, double wheelSize) {}

public RoadBike() {}

public RoadBike(int gearCount, double wheelSize) {}

e. public class MountainBike extends Bicycle {}

public class RoadBike extends Bicycle {}

f. public MountainBike() {

super(5, 26);

}

public RoadBike() {

super(10, 28);

}