Java Programming 2-1: Working with Pre-Written Code Practice Activities

Vocabulary Section

1. **A method that can modify an object**: **Mutator Method** (often referred to as a "setter").
2. **A method that can access the contents of an object but does not modify that object**: **Accessor Method** (often referred to as a "getter").
3. **Object that can store multiple object types and can grow and shrink dynamically as required**: **ArrayList**.
4. **The process where one object acquires the properties of another**: **Inheritance**.
5. **Allows you to check the quality of the code for a class independent of the rest of the program code**: **Unit Testing**.

JavaBank Exploration

- **Display Accounts**: If no accounts exist, it likely shows an empty list or a message indicating no accounts are present.
- **Create Accounts**: This would allow you to add new accounts. The system might prompt for details like account number, holder name, etc.
- **Delete Accounts**: This should remove an account from the system. It might ask for confirmation before deletion.
- **Make a Withdrawal Transaction**: Deducts a specified amount from an account, possibly after verifying that the balance is sufficient.
- **Make a Deposit Transaction**: Adds a specified amount to an account balance.

For the specific questions:

- **Display accounts before creation**: Typically, it should show no accounts or an error message.
- **Create an account without entering anything**: The system should ideally prevent this and prompt for necessary information.
- **Withdraw with no amount entered**: It should either display an error or prompt the user to enter an amount.
- **Deposit with no amount entered**: Similarly, it should display an error or prompt for an amount.

Bike Project Questions

1.

Primitive Data Type: For example, int might be used for storing the number of gears in a bike class.

2.
3.

String Concatenation Example: This could occur in a method that builds a description string for a bike, e.g., "Bike: " + bikeName.

4.
5.

Objects in the Program: Look for new keyword instances, such as MountainBike, RoadBike, etc.

6.
7.

Number of Constructors: Check each class to see how many constructors are defined. Typically, you might find a default constructor and others with parameters.

8.
9.

Super and Subclasses: Identify the base class (e.g., Bike) and the derived classes (e.g., MountainBike, RoadBike).

10.

Sample values for standard bikes:

11.

- **Mountain Bike (MB)**: handleBars: Bull Horn, tyres: RockShox XC32, tyreWidth: 20
- **Road Bike (RB)**: handleBars: Drop, tyres: Type, tyreWidth: 20

Calculator Program

1. **Import the Project**: Follow the steps to import the Calculator.jar into Eclipse.
2. **Run and Investigate**: Once run, see what basic operations the Calculator supports (likely addition, division, etc.).
3. **Enhance Functionality**:
    - **Add Multiplication and Subtraction**: Update the code to include buttons for multiplication and subtraction.
    - **Test Functionality**: Ensure all operations work as expected.
    - **Export as Runnable JAR**: Follow the steps to export your updated calculator so it can run independently.

Suggested Changes and Additions

- **Improvements**: Depending on what you find in JavaBank or the Calculator, you might suggest user input validation, more user-friendly interfaces, or additional features like transaction history.
- **New Features**: Consider adding features like loan processing in JavaBank or scientific functions in the Calculator.