Activity Report

The provided Java code demonstrates several object-oriented programming (OOP) concepts, including Encapsulation, Inheritance, Polymorphism, and Abstraction. Here's how each of these concepts is implemented in the code:

1. **Encapsulation:**

- Encapsulation is the practice of hiding the internal implementation details of a class and providing a public interface for interacting with the class. In the code:

  - The `Customer` and `BankAccount` classes encapsulate their data fields (attributes) by making them private and provide public methods (getters and setters) to access or modify these attributes.

  - Access to the internal state of these objects is controlled through these methods.

2. **Inheritance:****

- Inheritance is a mechanism in OOP that allows a new class (subclass or derived class) to inherit properties and behaviors (attributes and methods) from an existing class (superclass or base class). In the code:

  - The `SavingsAccount` and `CheckingAccount` classes inherit from the `BankAccount` class. They extend its functionality by adding specific behavior related to savings and checking accounts.

  - The `SavingsAccount` and `CheckingAccount` classes use the `super` keyword to call the constructor of the parent class (`BankAccount`) and initialize their own attributes.

3. **Polymorphism:****

- Polymorphism allows objects of different classes to be treated as objects of a common superclass. In the code:

  - The `displayAllDeductions` method is overridden in both `SavingsAccount` and `CheckingAccount` classes. This demonstrates method overriding, a form of runtime polymorphism, where a method in a subclass has the same name and signature as a method in its superclass.

  - The `for` loop in the `main` method iterates through a list of `BankAccount` objects, and because of polymorphism, it can call the `displayAllDeductions` method on different types of accounts (Savings or Checking) without knowing their specific types at compile-time.

4**. **Abstraction:****

- Abstraction involves simplifying complex systems by representing them in a more manageable and understandable way, typically by providing high-level interfaces and hiding implementation details. In the code:

- The `Customer` and `BankAccount` classes encapsulate their internal details and provide public methods to interact with their attributes.

- The `SavingsAccount` and `CheckingAccount` classes abstract away specific account behaviors by extending the `BankAccount` class and implementing their own logic for withdrawals, deposits, and deductions.

- The `displayAllDeductions` method in both subclasses abstracts away the details of how deductions are calculated and displayed, making it easy to extend this behavior in the future.

Overall, this code example effectively demonstrates the principles of Encapsulation, Inheritance, Polymorphism, and Abstraction in an object-oriented programming context.