Polymorphism

This zip contains a series of Java examples demonstrating fundamental object-oriented programming (OOP) concepts: Method Overloading and Runtime Polymorphism.

Structure

- **MethodOverloading**
  - `MethodOverloading.java`: This class showcases **method overloading**, where multiple methods with the same name can exist in a single class, differentiated by their parameter lists.

- **RuntimePolymorphism**
  - `Account.java`: The parent class for all bank accounts. It provides common methods like `deposit()` and `withdraw()`.
  - `CheckingAccount.java`: A subclass of `Account` with specific withdrawal logic (e.g., a transaction fee).
  - `SavingsAccount.java`: A subclass of `Account` with specific deposit logic (e.g., adding a bonus or interest).
  - `AccountDriver.java`: The main class for the bank account example. It demonstrates runtime polymorphism by using an array of `Account` objects to manage both checking and savings accounts.

- **Practice**
  - `Circle.java`: A class representing a circle. It extends `Shape.java`.
  - `Rectangle.java`: A class representing a rectangle. It also extends `Shape.java`.
  - `Shape.java`: **This is an exercise for the student.** This class should be the parent class for `Circle` and `Rectangle` and should define a method that its subclasses must override (e.g., `calculateArea()`).
  - `ShapeDriver.java`: The main class for the shapes example. It should demonstrate runtime polymorphism by creating instances of `Circle` and `Rectangle` and treating them as `Shape` objects.

How to Run

To run any of the examples, you must first compile the Java files and then execute the main class from your terminal.

Compile:

javac <FileName>.java

(e.g., `javac MethodOverloading.java`)

Run:

java <ClassName>

(e.g., `java MethodOverloading.MethodOverloading`)