

```
import java.util.ArrayList;
import java.util.List;

// The Account interface with required banking methods
interface Account {
    void deposit(double amount);
    void withdraw(double amount);
    double calculateInterest();
    void viewBalance();
}

// SavingsAccount implements Account and has its unique method to add interest
class SavingsAccount implements Account {
    private String accountNumber;
    private double balance;
    private double interestRate; // e.g., 0.05 for 5%

    public SavingsAccount(String accountNumber, double balance, double interestRate) {
        this.accountNumber = accountNumber;
        this.balance = balance;
        this.interestRate = interestRate;
    }

    // Deposit money into the savings account
    public void deposit(double amount) {
        if (amount > 0) {
            balance += amount;
            System.out.println("Deposited $" + amount + " in Savings Account " + accountNumber);
        } else {
            System.out.println("Deposit amount must be positive.");
        }
    }

    // Withdraw money from the savings account
    public void withdraw(double amount) {
        if (amount > 0 && amount <= balance) {
            balance -= amount;
            System.out.println("Withdrew $" + amount + " from Savings Account " + accountNumber);
        } else {
            System.out.println("Insufficient balance or invalid amount for Savings Account " + accountNumber);
        }
    }
}
```

```
}

// Calculate interest based on the current balance
public double calculateInterest() {
    double interest = balance * interestRate;
    System.out.println("Calculated interest for Savings Account " + accountNumber + " is: $" + interest);
    return interest;
}

// Unique method for SavingsAccount: add the calculated interest to the balance
public void addInterest() {
    double interest = calculateInterest();
    balance += interest;
    System.out.println("Interest added. New balance for Savings Account " + accountNumber + " is: $" + balance);
}

// View the current balance
public void viewBalance() {
    System.out.println("Savings Account " + accountNumber + " balance: $" + balance);
}

public String getAccountNumber() {
    return accountNumber;
}
}

// CurrentAccount implements Account and includes a method to check overdraft usage
class CurrentAccount implements Account {
    private String accountNumber;
    private double balance;
    private double overdraftLimit;

    public CurrentAccount(String accountNumber, double balance, double overdraftLimit) {
        this.accountNumber = accountNumber;
        this.balance = balance;
        this.overdraftLimit = overdraftLimit;
    }

    // Deposit money into the current account
    public void deposit(double amount) {
        if (amount > 0) {
```

```
        balance += amount;
        System.out.println("Deposited $" + amount + " in Current Account " + accountNumber);
    } else {
        System.out.println("Deposit amount must be positive.");
    }
}

// Withdraw money from the current account (allows using overdraft)
public void withdraw(double amount) {
    if (amount > 0 && (balance + overdraftLimit) >= amount) {
        balance -= amount;
        System.out.println("Withdrew $" + amount + " from Current Account " + accountNumber);
        if (balance < 0) {
            System.out.println("Overdraft used: $" + (-balance));
        }
    } else {
        System.out.println("Insufficient funds (including overdraft) for Current Account " + accountNumber);
    }
}

// For a CurrentAccount, interest is typically not accrued; return 0.
public double calculateInterest() {
    System.out.println("No interest calculated for Current Account " + accountNumber);
    return 0.0;
}

// Unique method for CurrentAccount: check if overdraft is being used
public void checkOverdraft() {
    if (balance < 0) {
        System.out.println("Current Account " + accountNumber + " is using an overdraft of $" + (-balance));
    } else {
        System.out.println("Current Account " + accountNumber + " is not using any overdraft.");
    }
}

// View the current balance
public void viewBalance() {
    System.out.println("Current Account " + accountNumber + " balance: $" + balance);
}

public String getAccountNumber() {
```

```
        return accountNumber;
    }
}

// The Bank class holds a list of accounts and methods for managing them
class Bank {
    private List<Account> accounts;

    public Bank() {
        accounts = new ArrayList<>();
    }

    // Add an account to the bank
    public void addAccount(Account account) {
        accounts.add(account);
        System.out.println("Account added successfully.");
    }

    // Remove an account from the bank
    public void removeAccount(Account account) {
        if (accounts.remove(account)) {
            System.out.println("Account removed successfully.");
        } else {
            System.out.println("Account not found.");
        }
    }

    // List all accounts with their current balances
    public void listAccounts() {
        System.out.println("\nListing all accounts:");
        for (Account account : accounts) {
            account.viewBalance();
        }
    }
}

// Main class to demonstrate the banking system functionality
public class BankingSystem {
    public static void main(String[] args) {
        Bank bank = new Bank();
    }
}
```

```
// Create sample accounts
SavingsAccount savings1 = new SavingsAccount("S001", 1000.0, 0.05);
CurrentAccount current1 = new CurrentAccount("C001", 500.0, 200.0);

// Add accounts to the bank
bank.addAccount(savings1);
bank.addAccount(current1);

// Perform transactions on the SavingsAccount
System.out.println("\n--- Savings Account Transactions ---");
savings1.deposit(200);
savings1.withdraw(50);
savings1.addInterest(); // Unique method to add interest
savings1.viewBalance();

// Perform transactions on the CurrentAccount
System.out.println("\n--- Current Account Transactions ---");
current1.deposit(300);
current1.withdraw(900); // This may utilize the overdraft facility
current1.checkOverdraft(); // Unique method to check overdraft usage
current1.viewBalance();


// List all accounts in the bank
bank.listAccounts();
}
}
```

```
PS C:\Users\sakmm\Desktop\java> cd "c:\Users\sakmm\Desktop\java\" ; if ($?) { javac BankingSystem.java } ; if ($?) { java BankingSystem }
Account added successfully.
Account added successfully.

--- Savings Account Transactions ---
Deposited $200.0 in Savings Account S001
Withdrew $50.0 from Savings Account S001
Calculated interest for Savings Account S001 is: $57.5
Interest added. New balance for Savings Account S001 is: $1207.5
Savings Account S001 balance: $1207.5

--- Current Account Transactions ---
Deposited $300.0 in Current Account C001
Withdrew $900.0 from Current Account C001
Overdraft used: $100.0
Current Account C001 is using an overdraft of $100.0
Current Account C001 balance: $-100.0

Listing all accounts:
Savings Account S001 balance: $1207.5
Current Account C001 balance: $-100.0
PS C:\Users\sakmm\Desktop\java>
```

 Code Code

```
// Define the Animal interface with three methods
interface Animal {
    void eat();
    void sleep();
    void run();
}

// Lion class implementing Animal interface
class Lion implements Animal {
    @Override
    public void eat() {
        System.out.println("Lion hunts and eats its prey.");
    }

    @Override
    public void sleep() {
        System.out.println("Lion sleeps for about 20 hours a day.");
    }

    @Override
    public void run() {
        System.out.println("Lion runs at a speed of around 50 mph.");
    }
}

// Tiger class implementing Animal interface
class Tiger implements Animal {
    @Override
    public void eat() {
        System.out.println("Tiger eats large animals like deer and boar.");
    }

    @Override
    public void sleep() {
        System.out.println("Tiger sleeps in secluded areas of the jungle.");
    }

    @Override
```

```
    public void run() {
        System.out.println("Tiger runs stealthily to ambush its prey.");
    }
}

// Deer class implementing Animal interface
class Deer implements Animal {
    @Override
    public void eat() {
        System.out.println("Deer grazes on grass, leaves, and fruits.");
    }

    @Override
    public void sleep() {
        System.out.println("Deer takes short naps to remain alert to predators.");
    }

    @Override
    public void run() {
        System.out.println("Deer runs swiftly to escape danger.");
    }
}

// Main class to test the behavior of each animal
public class AnimalTest {
    public static void main(String[] args) {
        Animal lion = new Lion();
        Animal tiger = new Tiger();
        Animal deer = new Deer();

        System.out.println("Lion Behavior:");
        lion.eat();
        lion.sleep();
        lion.run();

        System.out.println("\nTiger Behavior:");
        tiger.eat();
        tiger.sleep();
        tiger.run();
    }
}
```



```
        System.out.println("\nDeer Behavior:");  
        deer.eat();  
        deer.sleep();  
        deer.run();  
    }  
}
```

```
PS C:\Users\sakmm\Desktop\java> cd "c:\Users\sakmm\Desktop\java\" ; if ($?) { javac AnimalTest.java } ; if ($?) { java AnimalTest }
```

Lion Behavior:

Lion hunts and eats its prey.

Lion sleeps for about 20 hours a day.

Lion runs at a speed of around 50 mph.

Tiger Behavior:

Tiger eats large animals like deer and boar.

Tiger sleeps in secluded areas of the jungle.

Tiger runs stealthily to ambush its prey.

Deer Behavior:

Deer grazes on grass, leaves, and fruits.

Deer takes short naps to remain alert to predators.

Deer runs swiftly to escape danger.

```
PS C:\Users\sakmm\Desktop\java>
```