```
interface TransactionOperations {
   void deposit(double amount);
   void withdraw(double amount);
   void transfer(Account toAccount, double amount);
}

abstract class Account implements TransactionOperations {
   protected String accountNumber;
   protected String accountHolder;
   protected double balance;

   public Account(String accountNumber, String accountHolder, double initialDeposit) {
      this.accountNumber = accountNumber;
      this.accountHolder = accountHolder;
      this.balance = initialDeposit;
   }

   public synchronized void deposit(double amount) {
      balance += amount;
      System.out.println(Thread.currentThread().getName() + " deposited " + amount + ". New
Balance: " + balance);
   }

   public synchronized void withdraw(double amount) {
      if (balance >= amount) {
         balance -= amount;
         System.out.println(Thread.currentThread().getName() + " withdrew " + amount + ". New
Balance: " + balance);
      } else {          System.out.println("Insufficient funds for " + Thread.currentThread().getName());
      }
   }
```

```
    public void transfer(Account toAccount, double amount) {
        synchronized (this) {
            if (balance >= amount) {
                this.withdraw(amount);
                toAccount.deposit(amount);
                System.out.println(Thread.currentThread().getName() + " transferred " + amount + " to " +
toAccount.accountNumber);
            } else {
                System.out.println("Transfer failed due to insufficient funds for " +
Thread.currentThread().getName());
            }
        }
    }

    public abstract void displayAccountDetails();
}

class SavingsAccount extends Account {
    private double interestRate;

    public SavingsAccount(String accountNumber, String accountHolder, double initialDeposit, double
interestRate) {
        super(accountNumber, accountHolder, initialDeposit);
        this.interestRate = interestRate;    }

    @Override
    public void displayAccountDetails() {
        System.out.println("Savings Account [Account Number: " + accountNumber + ", Holder: " +
accountHolder +
                ", Balance: " + balance + ", Interest Rate: " + interestRate + "%]");
    }
}
```

```
class CurrentAccount extends Account {
    private double overdraftLimit;


    public CurrentAccount(String accountNumber, String accountHolder, double initialDeposit, double
overdraftLimit) {
        super(accountNumber, accountHolder, initialDeposit);
        this.overdraftLimit = overdraftLimit;
    }


    @Override
    public synchronized void withdraw(double amount) {
        if (balance + overdraftLimit >= amount) {
            balance -= amount;
            System.out.println(Thread.currentThread().getName() + " withdrew " + amount + ". New
Balance: " + balance);
        } else {
            System.out.println("Overdraft limit exceeded for " +Thread.currentThread().getName());
        }
    }


    @Override
    public void displayAccountDetails() {
        System.out.println("Current Account [Account Number: " + accountNumber + ", Holder: " +
accountHolder +
                ", Balance: " + balance + ", Overdraft Limit: " + overdraftLimit + "]");
    }
}


class TransactionTask implements Runnable {
    private TransactionOperations operation;
    private String type;
    private double amount;
    private Account toAccount;
    public TransactionTask(TransactionOperations operation, String type, double amount) {
        this.operation = operation;
        this.type = type;
        this.amount = amount;
    }

    public TransactionTask(TransactionOperations operation, String type, double amount, Account
toAccount) {
        this(operation, type, amount);
        this.toAccount = toAccount;
    }

    @Override
    public void run() {
        switch (type.toLowerCase()) {
            case "deposit":
                operation.deposit(amount);
                break;
            case "withdraw":
                operation.withdraw(amount);
                break;
            case "transfer":
                if (operation instanceof Account && toAccount != null) {
                    ((Account) operation).transfer(toAccount, amount);
                }
                break;
            default:                System.out.println("Invalid transaction type");
        }
    }
```

```java
}

public class BankManagementSystem {
  public static void main(String[] args) {

    SavingsAccount savings = new SavingsAccount("SA123", "Alice", 1000.0, 5.0);
    CurrentAccount current = new CurrentAccount("CA456", "Bob", 2000.0, 500.0);

    savings.displayAccountDetails();
    current.displayAccountDetails();


    Thread t1 = new Thread(new TransactionTask(savings, "deposit", 500.0), "Thread-1");
    Thread t2 = new Thread(new TransactionTask(current, "withdraw", 1500.0), "Thread-2");
    Thread t3 = new Thread(new TransactionTask(savings, "transfer", 200.0, current), "Thread-3");
    Thread t4 = new Thread(new TransactionTask(current, "deposit", 300.0), "Thread-4");


    t1.start();
    t2.start();
    t3.start();
    t4.start();

    try {
      t1.join();
      t2.join();
      t3.join();
      t4.join();
    } catch (InterruptedException e) {
      e.printStackTrace();
    }
```

```
        System.out.println("\nFinal Account Details:");
        savings.displayAccountDetails();
        current.displayAccountDetails();
    }
}
```