

SET 2

1. Design a class BankAccount with properties accountNumber and balance, and methods deposit() and withdraw(). Extend this class with subclasses SavingsAccount and CheckingAccount. Implement specific rules such as minimum balance requirements and interest calculation for savings accounts.

Program:

```
class BankAccount {  
    public String accountNumber;  
    public double balance;  
    public BankAccount(String accountNumber, double initialBalance) {  
        this.accountNumber = accountNumber;  
        this.balance = initialBalance;  
    }  
    public String getAccountNumber() {  
        return accountNumber;  
    }  
    public double getBalance() {  
        return balance;  
    }  
    public void deposit(double amount) {  
        if (amount > 0) {  
            balance += amount;  
            System.out.println("Deposited: " + amount);  
        } else {  
            System.out.println("Invalid deposit amount");  
        }  
    }  
    public boolean withdraw(double amount) {  
        if (amount > 0 && balance >= amount) {
```

```

        balance -= amount;

        System.out.println("Withdrew: " + amount);

        return true;
    } else {

        System.out.println("Invalid withdrawal amount or insufficient balance");

        return false;
    }
}

}

class SavingsAccount extends BankAccount {

    private static final double minbal = 1000.0;

    private static final double INTEREST_RATE = 0.02;

    public SavingsAccount(String accountNumber, double initialBalance) {

        super(accountNumber, initialBalance);
    }

    @Override

    public boolean withdraw(double amount) {

        if (balance - amount >= minbal) {

            return super.withdraw(amount);

        } else {

            System.out.println("Withdrawal would breach minimum balance of " + minbal);

            return false;

        }

    }

    public void addInterest() {

        double interest = balance * INTEREST_RATE;

        deposit(interest);

        System.out.println("Interest added: " + interest);

    }
}

```

```

}

class CheckingAccount extends BankAccount {

    private static final double limit = 500.0;

    public CheckingAccount(String accountNumber, double initialBalance) {

        super(accountNumber, initialBalance);

    }

    @Override

    public boolean withdraw(double amount) {

        if (balance - amount >= -limit) {

            return super.withdraw(amount);

        } else {

            System.out.println("Withdrawal exceeds overdraft limit of "+limit);

            return false;

        }

    }

}

public class Main {

    public static void main(String[] args) {

        SavingsAccount savings = new SavingsAccount("SA12345", 2000.0);

        CheckingAccount checking = new CheckingAccount("CA12345", 1000.0);

        savings.deposit(2000.0);

        checking.deposit(500.0);

        savings.withdraw(500.0);

        checking.withdraw(1000.0);

        savings.addInterest();

        System.out.println("Savings balance: " + savings.getBalance());

        System.out.println("Checking balance: " + checking.getBalance());

    }

}

```

Output:

```
java -cp /tmp/8iiysLx51w/Main
Deposited: 2000.0
Deposited: 500.0
Withdrew: 500.0
Withdrew: 1000.0
Deposited: 70.0
Interest added: 70.0
Savings balance: 3570.0
Checking balance: 500.0

=== Code Execution Successful ===
```

2. Create a base class **GameCharacter** with properties **name**, **health**, and **level**. Extend this class with subclasses **Warrior**, **Mage**, and **Archer**. Implement methods such as **attack()** and **defend()** differently for each subclass, showcasing polymorphism through inheritance.

Program:

```
abstract class GameCharacter {
    public String name;
    public int health;
    public int level;

    public GameCharacter(String name, int health, int level) {
        this.name = name;
        this.health = health;
        this.level = level;
    }

    public String getName() {
        return name;
    }
}
```

```
}
```

```
public int getHealth() {  
    return health;  
}
```

```
public int getLevel() {  
    return level;  
}
```

```
public void setHealth(int health) {  
    this.health = health;  
}
```

```
public void levelUp() {  
    this.level++;  
}
```

```
public abstract void attack();
```

```
public abstract void defend();  
}
```

```
class Warrior extends GameCharacter {
```

```
    public Warrior(String name, int health, int level) {  
        super(name, health, level);  
    }
```

```
    public void attack() {  
        System.out.println(name + " swings a sword for " + (level * 10) + " damage!");  
    }
```

```

    public void defend() {
        System.out.println(name + " blocks with a shield, reducing damage by " + (level * 5) +
".");
    }
}

class Mage extends GameCharacter {

    public Mage(String name, int health, int level) {
        super(name, health, level);
    }

    public void attack() {
        System.out.println(name + " casts a fireball for " + (level * 12) + " damage!");
    }

    public void defend() {
        System.out.println(name + " conjures a magical barrier, reducing damage by " + (level *
4) + ".");
    }
}

class Archer extends GameCharacter {

    public Archer(String name, int health, int level) {
        super(name, health, level);
    }

    public void attack() {
        System.out.println(name + " shoots an arrow for " + (level * 8) + " damage!");
    }

    public void defend() {
        System.out.println(name + " dodges the attack, reducing damage by " + (level * 3) + ".");
    }
}

```

```

public class GameTest {

    public static void main(String[] args) {

        GameCharacter warrior = new Warrior("rohith", 120, 10);

        GameCharacter mage = new Mage("rao", 50, 6);

        GameCharacter archer = new Archer("deepa", 45, 9);


        warrior.attack();

        warrior.defend();


        mage.attack();

        mage.defend();


        archer.attack();

        archer.defend();

    }

}

```

Output:

```

java -cp /tmp/ArxdG12bYj/GameTest
rohith swings a sword for 100 damage!
rohith blocks with a shield, reducing damage by 50.
rao casts a fireball for 72 damage!
rao conjures a magical barrier, reducing damage by 24.
deepa shoots an arrow for 72 damage!
deepa dodges the attack, reducing damage by 27.

=== Code Execution Successful ===

```

3. Design a class Product with properties productId, name, and price. Extend this class with

subclasses Electronics and Clothing. Implement methods to calculate discounts based on membership status for electronics and seasonal sales for clothing.

Program:

```
class Product {  
    public String productId;  
    public String name;  
    public double price;  
  
    public Product(String productId, String name, double price) {  
        this.productId = productId;  
        this.name = name;  
        this.price = price;  
    }  
  
    public double getPrice() {  
        return price;  
    }  
}  
  
class Electronics extends Product {  
    public Electronics(String productId, String name, double price) {  
        super(productId, name, price);  
    }  
  
    public double calculateDiscount(boolean isMember) {  
        return isMember ? getPrice() * 0.90 : getPrice();  
    }  
}  
  
class Clothing extends Product {  
    public Clothing(String productId, String name, double price) {  
        super(productId, name, price);  
    }  
}
```



```

    public double calculateDiscount(boolean isSeasonalSale) {
        return isSeasonalSale ? getPrice() * 0.80 : getPrice();
    }
}

public class Main {

    public static void main(String[] args) {

        Electronics laptop = new Electronics("E123", "Laptop", 1000);

        Clothing tshirt = new Clothing("C123", "T-Shirt", 50);

        System.out.println("Electronics (Laptop) Price for Member: " +
laptop.calculateDiscount(true));

        System.out.println("Electronics (Laptop) Price for Non-Member: " +
laptop.calculateDiscount(false));

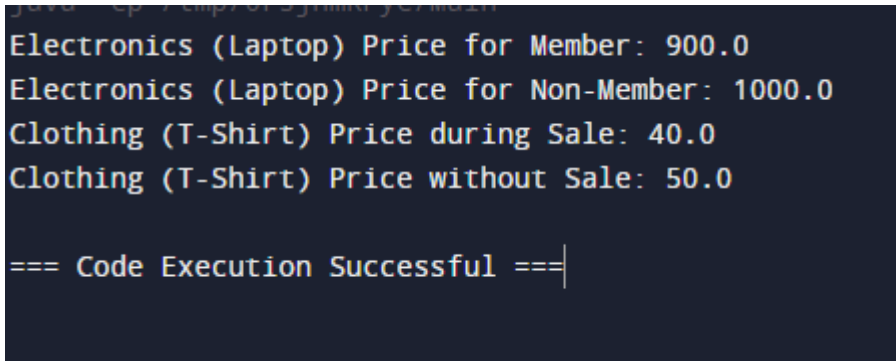
        System.out.println("Clothing (T-Shirt) Price during Sale: " +
tshirt.calculateDiscount(true));

        System.out.println("Clothing (T-Shirt) Price without Sale: " +
tshirt.calculateDiscount(false));

    }
}

```

Output:



```

Electronics (Laptop) Price for Member: 900.0
Electronics (Laptop) Price for Non-Member: 1000.0
Clothing (T-Shirt) Price during Sale: 40.0
Clothing (T-Shirt) Price without Sale: 50.0

=== Code Execution Successful ===

```

4. Design a class LibraryItem with properties title, author, and year. Extend this class with subclasses Book and DVD. Implement methods for checking in and out items, and display detailed information for each item type.

Program:

```
abstract class LibraryItem {  
    String title;  
    String author;  
    int year;  
    boolean isCheckedOut;  
  
    public LibraryItem(String title, String author, int year) {  
        this.title = title;  
        this.author = author;  
        this.year = year;  
        this.isCheckedOut = false;  
    }  
  
    public void checkOut() {  
        if (!isCheckedOut) {  
            isCheckedOut = true;  
            System.out.println(title + " has been checked out.");  
        } else {  
            System.out.println(title + " is already checked out.");  
        }  
    }  
  
    public void checkIn() {  
        if (isCheckedOut) {  
            isCheckedOut = false;  
            System.out.println(title + " has been checked in.");  
        } else {  
            System.out.println(title + " is already checked in.");  
        }  
    }  
}
```

```
}
```

```
abstract void displayInfo();
```

```
}
```

```
class Book extends LibraryItem {
```

```
    public Book(String title, String author, int year) {
```

```
        super(title, author, year);
```

```
    }
```

```
    void displayInfo() {
```

```
        System.out.println("Book Title: " + title);
```

```
        System.out.println("Author: " + author);
```

```
        System.out.println("Year: " + year);
```

```
        System.out.println("Checked Out: " + (isCheckedOut ? "Yes" : "No"));
```

```
    }
```

```
}
```

```
class DVD extends LibraryItem {
```

```
    public DVD(String title, String author, int year) {
```

```
        super(title, author, year);
```

```
    }
```

```
    void displayInfo() {
```

```
        System.out.println("DVD Title: " + title);
```

```
        System.out.println("Director: " + author); // Assuming author represents the director for
```

```
        System.out.println("Year: " + year);
```

```
        System.out.println("Checked Out: " + (isCheckedOut ? "Yes" : "No"));
```

```
    }
```

```
}
```

```
public class Main {
```

```
    public static void main(String[] args) {
```

```
        LibraryItem book = new Book("1984", "George Orwell", 1949);
```

```
        LibraryItem dvd = new DVD("Inception", "Christopher Nolan", 2010);
```

```
        System.out.println("Book Information:");
```

```
        book.displayInfo();
```

```
        System.out.println("\nDVD Information:");
```

```
        dvd.displayInfo();
```

```
        System.out.println("\nChecking out the book:");
```

```
        book.checkOut();
```

```
        book.displayInfo();
```

```
        System.out.println("\nChecking in the book:");
```

```
        book.checkIn();
```

```
        book.displayInfo();
```

```
        System.out.println("\nChecking out the DVD:");
```

```
        dvd.checkOut();
```

```
        dvd.displayInfo();
```

```
        System.out.println("\nChecking in the DVD:");
```

```
        dvd.checkIn();
```

```
        dvd.displayInfo();
```

```
    }
```

```
}
```

Output:

```
java -cp ./lib/* org.javacore.ch07.m02n
Book Information:
Book Title: 1984
Author: George Orwell
Year: 1949
Checked Out: No

DVD Information:
DVD Title: Inception
Director: Christopher Nolan
Year: 2010
Checked Out: No

Checking out the book:
1984 has been checked out.
Book Title: 1984
Author: George Orwell
Year: 1949

=== Code Exited With Errors ===
```