

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.

Main.java	Output
<pre>1- class BankAccount { 2- protected String accountNumber; 3- protected double balance; 4- public BankAccount(String accountNumber, double balance) { 5- this.accountNumber = accountNumber; 6- this.balance = balance; 7- } 8- public void deposit(double amount) { 9- balance += amount; 10- System.out.println("Deposited: \$" + amount); 11- } 12- public void withdraw(double amount) { 13- if (balance >= amount) { 14- balance -= amount; 15- System.out.println("Withdrawn: \$" + amount); 16- } else { 17- System.out.println("Insufficient funds"); 18- } 19- } 20- } 21- class SavingsAccount extends BankAccount { 22- private double interestRate; 23- public SavingsAccount(String accountNumber, double balance, double interestRate) { 24- super(accountNumber, balance); 25- this.interestRate = interestRate; 26- } 27- public void addInterest() { 28- double interest = balance * interestRate / 100; 29- balance += interest; 30- System.out.println("Interest added: \$" + interest); 31- } 32- }</pre>	<pre>java -cp /tmp/J0kZrUswfZ/Main Deposited: \$500.0 Interest added: \$37.5 Withdrawn: \$200.0 Deposited: \$300.0 Withdrawn: \$2500.0 === Code Execution Successful ===</pre>

Main.java	Output
<pre>29- balance += interest; 30- System.out.println("Interest added: \$" + interest); 31- } 32- } 33- class CheckingAccount extends BankAccount { 34- private double overdraftLimit; 35- public CheckingAccount(String accountNumber, double balance, double overdraftLimit) { 36- super(accountNumber, balance); 37- this.overdraftLimit = overdraftLimit; 38- } 39- public void withdraw(double amount) { 40- if (balance + overdraftLimit >= amount) { 41- balance -= amount; 42- System.out.println("Withdrawn: \$" + amount); 43- } else { 44- System.out.println("Exceeded overdraft limit"); 45- } 46- } 47- } 48- public class Main { 49- public static void main(String[] args) { 50- SavingsAccount savings = new SavingsAccount("SAV123", 1000.0, 2.5); 51- savings.deposit(500.0); 52- savings.addInterest(); 53- savings.withdraw(200.0); 54- CheckingAccount checking = new CheckingAccount("CHK456", 2000.0, 500.0); 55- checking.deposit(300.0); 56- checking.withdraw(2500.0); 57- } 58- } 59- }</pre>	<pre>java -cp /tmp/J0kZrUswfZ/Main Deposited: \$500.0 Interest added: \$37.5 Withdrawn: \$200.0 Deposited: \$300.0 Withdrawn: \$2500.0 === Code Execution Successful ===</pre>

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.

Main.java	Output
<pre>1- class GameCharacter { 2- String name; 3- int health; 4- int level; 5- public GameCharacter(String name, int health, int level) { 6- this.name = name; 7- this.health = health; 8- this.level = level; 9- } 10- public void attack() { 11- System.out.println("Attacking..."); 12- } 13- public void defend() { 14- System.out.println("Defending..."); 15- } 16- } 17- class Warrior extends GameCharacter { 18- public Warrior(String name, int health, int level) { 19- super(name, health, level); 20- } 21- public void attack() { 22- System.out.println("Warrior is attacking fiercely!"); 23- } 24- public void defend() { 25- System.out.println("Warrior is defending with a shield!"); 26- } 27- } 28- class Mage extends GameCharacter { 29- public Mage(String name, int health, int level) { 30- super(name, health, level); 31- } 32- public void attack() {</pre>	<pre>java -cp /tmp/QRGpN4fhYN/Main Warrior is attacking fiercely! Warrior is defending with a shield! Mage is casting a powerful spell! Mage is creating a magical barrier for defense! Archer is shooting arrows swiftly! Archer is evading attacks with agility! === Code Execution Successful ===</pre>

Main.java	Output
<pre>32- public void attack() { 33- System.out.println("Mage is casting a powerful spell!"); 34- } 35- public void defend() { 36- System.out.println("Mage is creating a magical barrier for defense!"); 37- } 38- } 39- class Archer extends GameCharacter { 40- public Archer(String name, int health, int level) { 41- super(name, health, level); 42- } 43- public void attack() { 44- System.out.println("Archer is shooting arrows swiftly!"); 45- } 46- public void defend() { 47- System.out.println("Archer is evading attacks with agility!"); 48- } 49- } 50- public class Main { 51- public static void main(String[] args) { 52- Warrior warrior = new Warrior("Aldric", 100, 10); 53- Mage mage = new Mage("Sylvia", 80, 12); 54- Archer archer = new Archer("Robin", 90, 11); 55- 56- warrior.attack(); 57- warrior.defend(); 58- 59- mage.attack(); 60- mage.defend(); 61- 62- archer.attack(); 63- archer.defend(); 64- } 65- }</pre>	<pre>java -cp /tmp/QRGpN4fhYN/Main Warrior is attacking fiercely! Warrior is defending with a shield! Mage is casting a powerful spell! Mage is creating a magical barrier for defense! Archer is shooting arrows swiftly! Archer is evading attacks with agility! === Code Execution Successful ===</pre>

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.

Main.java	Output
<pre>1- class Product { 2- int productId; 3- String name; 4- double price; 5- 6- public Product(int productId, String name, double price) { 7- this.productId = productId; 8- this.name = name; 9- this.price = price; 10- } 11- } 12- 13- class Electronics extends Product { 14- public Electronics(int productId, String name, double price) { 15- super(productId, name, price); 16- } 17- 18- public double calculateDiscount(double price, boolean isMember) { 19- if (isMember) { 20- return price * 0.1; 21- } else { 22- return 0.0; 23- } 24- } 25- } 26- class Clothing extends Product { 27- public Clothing(int productId, String name, double price) { 28- super(productId, name, price); 29- } 30- 31- public double calculateDiscount(double price, boolean isSeasonSale) { 32- if (isSeasonSale) { 33- return price * 0.2; 34- } else { 35- return 0.0; 36- } 37- } 38- }</pre>	<pre>java -cp /tmp/9UemYHe0RL/Main Electronic Product Discount: \$100.0 Clothing Product Discount: \$4.0 === Code Execution Successful ===</pre>

Main.java	Output
<pre>25- } 26- class Clothing extends Product { 27- public Clothing(int productId, String name, double price) { 28- super(productId, name, price); 29- } 30- 31- public double calculateDiscount(double price, boolean isSeasonSale) { 32- if (isSeasonSale) { 33- return price * 0.2; 34- } else { 35- return 0.0; 36- } 37- } 38- } 39- public class Main { 40- public static void main(String[] args) { 41- Electronics electronicProduct = new Electronics(1, "Laptop", 1000.0); 42- Clothing clothingProduct = new Clothing(2, "T-shirt", 20.0); 43- 44- boolean isMember = true; 45- boolean isSeasonSale = true; 46- 47- double electronicDiscount = electronicProduct.calculateDiscount 48- (electronicProduct.price, isMember); 49- double clothingDiscount = clothingProduct.calculateDiscount(clothingProduct 50- .price, isSeasonSale); 51- 52- System.out.println("Electronic Product Discount: \$" + electronicDiscount); 53- System.out.println("Clothing Product Discount: \$" + clothingDiscount); 54- } 55- }</pre>	<pre>java -cp /tmp/9UemYHe0RL/Main Electronic Product Discount: \$100.0 Clothing Product Discount: \$4.0 === Code Execution Successful ===</pre>

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.

Main.java	Output
<pre>1- class LibraryItem { 2- String title; 3- String author; 4- int year; 5- public LibraryItem(String title, String author, int year) { 6- this.title = title; 7- this.author = author; 8- this.year = year; 9- } 10- public void checkOut() { 11- System.out.println("Checking out: " + title); 12- } 13- public void checkIn() { 14- System.out.println("Checking in: " + title); 15- } 16- public void displayInfo() { 17- System.out.println("Title: " + title); 18- System.out.println("Author: " + author); 19- System.out.println("Year: " + year); 20- } 21- } 22- class Book extends LibraryItem { 23- public Book(String title, String author, int year) { 24- super(title, author, year); 25- } 26- } 27- class DVD extends LibraryItem { 28- public DVD(String title, String author, int year) { 29- super(title, author, year); 30- } 31- }</pre>	<pre>java -cp /tmp/nztwhQa1VT/Main Title: The Great Gatsby Author: F. Scott Fitzgerald Year: 1925 Checking out: The Great Gatsby Checking in: The Great Gatsby Title: Inception Author: Christopher Nolan Year: 2010 Checking out: Inception Checking in: Inception === Code Execution Successful ===</pre>

Main.java	Output
<pre>17- System.out.println("Title: " + title); 18- System.out.println("Author: " + author); 19- System.out.println("Year: " + year); 20- } 21- } 22- class Book extends LibraryItem { 23- public Book(String title, String author, int year) { 24- super(title, author, year); 25- } 26- } 27- class DVD extends LibraryItem { 28- public DVD(String title, String author, int year) { 29- super(title, author, year); 30- } 31- } 32- public class Main { 33- public static void main(String[] args) { 34- Book book = new Book("The Great Gatsby", "F. Scott Fitzgerald", 1925); 35- DVD dvd = new DVD("Inception", "Christopher Nolan", 2010); 36- 37- book.displayInfo(); 38- book.checkOut(); 39- book.checkIn(); 40- 41- System.out.println(); 42- 43- dvd.displayInfo(); 44- dvd.checkOut(); 45- dvd.checkIn(); 46- } 47- } 48-</pre>	<pre>java -cp /tmp/nztwhQa1VT/Main Title: The Great Gatsby Author: F. Scott Fitzgerald Year: 1925 Checking out: The Great Gatsby Checking in: The Great Gatsby Title: Inception Author: Christopher Nolan Year: 2010 Checking out: Inception Checking in: Inception === Code Execution Successful ===</pre>