

TEST-7

JAVA PROGRAMING

CSA0961

1.

```
class BankAccount {  
    private String accountNumber;  
    private double balance;  
  
    public BankAccount(String  
accountNumber, double initialBalance) {  
        this.accountNumber =  
accountNumber;  
        this.balance = initialBalance;  
    }  
  
    public void deposit(double amount) {  
        if (amount > 0) {
```

```
        balance += amount;
    }
}
```

```
public void withdraw(double amount) {
    if (amount > 0 && amount <=
balance) {
        balance -= amount;
    }
}
```

```
public double getBalance() {
    return balance;
}
}
```

```
class SavingsAccount extends
BankAccount {

    private static final double
MIN_BALANCE = 1000;

    private static final double
INTEREST_RATE = 0.02;


    public SavingsAccount(String
accountNumber, double initialBalance) {
        super(accountNumber, initialBalance);
        if (initialBalance < MIN_BALANCE) {
            throw new
IllegalArgumentException("Initial balance
must be at least " + MIN_BALANCE);
        }
    }

    public void addInterest() {
```

```
        double interest = getBalance() *  
INTEREST_RATE;  
        deposit(interest);  
    }
```

```
    @Override  
    public void withdraw(double amount) {  
        if (getBalance() - amount >=  
MIN_BALANCE) {  
            super.withdraw(amount);  
        } else {  
            System.out.println("Withdrawal  
denied! Minimum balance requirement  
not met.");  
        }  
    }  
}
```

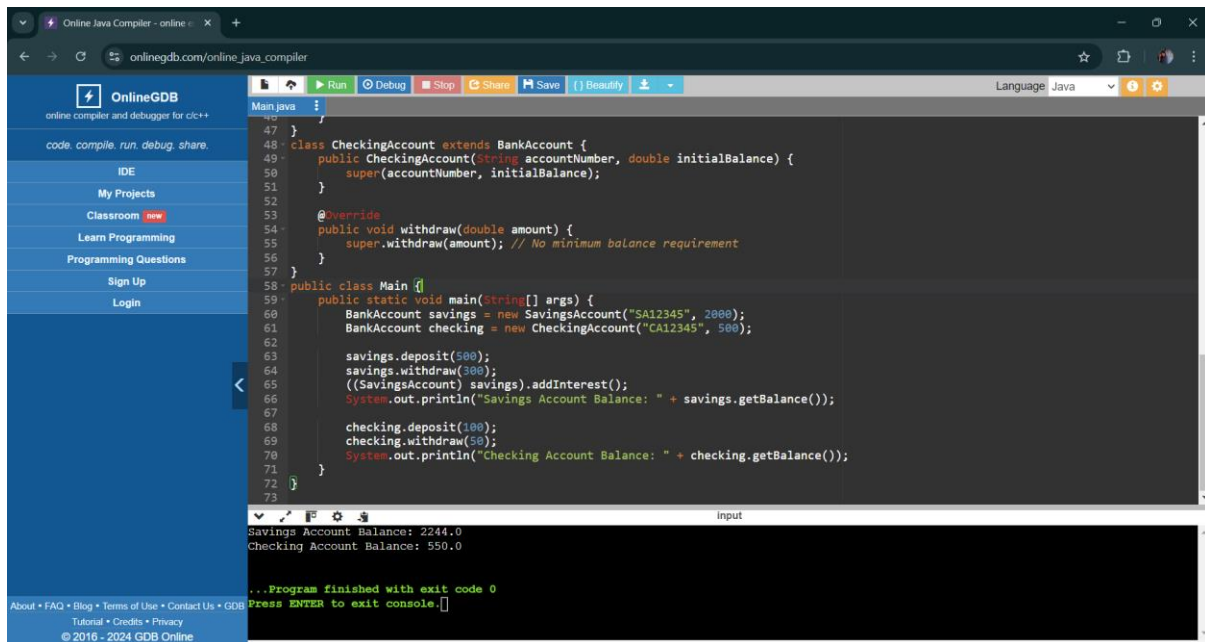
```
class CheckingAccount extends
BankAccount {
    public CheckingAccount(String
accountNumber, double initialBalance) {
        super(accountNumber, initialBalance);
    }
    @Override
    public void withdraw(double amount) {
        super.withdraw(amount); // No
minimum balance requirement
    }
}

public class Main {
    public static void main(String[] args) {
        BankAccount savings = new
SavingsAccount("SA12345", 2000);
```

```
BankAccount checking = new  
CheckingAccount("CA12345", 500);
```

```
    savings.deposit(500);  
    savings.withdraw(300);  
    ((SavingsAccount)  
savings).addInterest();  
    System.out.println("Savings Account  
Balance: " + savings.getBalance());
```

```
    checking.deposit(100);  
    checking.withdraw(50);  
    System.out.println("Checking Account  
Balance: " + checking.getBalance());  
    }  
}
```



The screenshot shows the OnlineGDB IDE interface. On the left is a sidebar with navigation links: OnlineGDB, code, compile, run, debug, share, IDE, My Projects, Classroom (marked 'new'), Learn Programming, Programming Questions, Sign Up, and Login. The main editor area displays a Java file named 'Main.java'. The code defines a 'CheckingAccount' class that extends 'BankAccount' and overrides the 'withdraw' method. A 'Main' class contains a 'main' method that creates 'SavingsAccount' and 'CheckingAccount' objects, performs deposit and withdrawal operations, and prints their balances. The output console at the bottom shows the results: 'Savings Account Balance: 2244.0' and 'Checking Account Balance: 550.0', followed by a message indicating the program finished with exit code 0.

```
47 }
48 class CheckingAccount extends BankAccount {
49     public CheckingAccount(String accountNumber, double initialBalance) {
50         super(accountNumber, initialBalance);
51     }
52
53     @Override
54     public void withdraw(double amount) {
55         super.withdraw(amount); // No minimum balance requirement
56     }
57 }
58
59 public class Main {
60     public static void main(String[] args) {
61         BankAccount savings = new SavingsAccount("SA12345", 2000);
62         BankAccount checking = new CheckingAccount("CA12345", 500);
63
64         savings.deposit(500);
65         savings.withdraw(300);
66         ((SavingsAccount) savings).addInterest();
67         System.out.println("Savings Account Balance: " + savings.getBalance());
68
69         checking.deposit(100);
70         checking.withdraw(50);
71         System.out.println("Checking Account Balance: " + checking.getBalance());
72     }
73 }
```

Savings Account Balance: 2244.0
Checking Account Balance: 550.0
...Program finished with exit code 0
Press ENTER to exit console.

2.

```
public class Main {
```

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

```
        GameCharacter warrior = new
        Warrior("Thor", 100, 1);
```

```
        GameCharacter mage = new
        Mage("Gandalf", 80, 1);
```

```
        GameCharacter archer = new
        Archer("Legolas", 90, 1);
```

```
        warrior.attack(mage);
```

```
mage.defend();  
mage.attack(archer);  
archer.defend();  
archer.attack(warrior);  
warrior.defend();
```

```
    System.out.println(warrior.getName()  
+ " Health: " + warrior.getHealth());
```

```
    System.out.println(mage.getName() +  
" Health: " + mage.getHealth());
```

```
    System.out.println(archer.getName()  
+ " Health
```

```
public class Main {
```

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

```
        Warrior warrior = new  
Warrior("Aragorn", 100, 5);
```



```
Mage mage = new Mage("Gandalf",  
80, 7);
```

```
Archer archer = new Archer("Legolas",  
90, 6);
```

```
warrior.attack(mage);
```

```
warrior.defend();
```

```
System.out.println(warrior.getName()  
+ " has " + warrior.getHealth() + " health  
left.");
```

```
mage.attack(archer);
```

```
mage.defend();
```

```
System.out.println(mage.getName() +  
" has " + mage.getHealth() + " health  
left.");
```

```
archer.attack(warrior);
```

```
archer.defend();
```

```

        System.out.println(archer.getName()
+ " has " + archer.getHealth() + " health
left.");
    }
}

```

The screenshot shows an online Java compiler interface. The main editor displays the following Java code:

```

1 abstract class GameCharacter {
2     private String name;
3     private int health;
4     private 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 abstract void attack(GameCharacter target);
11    public abstract void defend();
12    public String getName() {
13        return name;
14    }
15    public int getHealth() {
16        return health;
17    }
18    public int getLevel() {
19        return level;
20    }
21    public void takeDamage(int damage) {
22        health -= damage;
23        if (health < 0) {
24            health = 0;
25        }
26    }
27 }

```

Below the code editor, the console output shows the execution of a program that uses the `GameCharacter` class:

```

Gandalf has 65 health left.
Legolas shoots an arrow at Aragorn!
Legolas dodges swiftly!
Legolas has 70 health left.
...Program finished with exit code 0
Press ENTER to exit console.

```

3.

```

class Product {
    private String productId;
    private String name;
    private 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) {  
        double discount = isMember ? 0.1 :  
0.05;  
        return getPrice() * (1 - discount);  
    }  
}
```

```
class Clothing extends Product {  
    public Clothing(String productId, String  
name, double price) {  
        super(productId, name, price);  
    }  
}
```

```
    public double calculateDiscount(String  
season) {
```

```
        double discount =  
season.equalsIgnoreCase("Winter") ? 0.2 :  
0.1;
```

```
        return getPrice() * (1 - discount);  
    }  
}
```

```
public class Main {  
    public static void main(String[] args) {  
        Electronics laptop = new  
Electronics("E001", "Laptop", 1000);  
        double laptopPriceAfterDiscount =  
laptop.calculateDiscount(true);  
        System.out.println("Electronics Price  
after discount: $" +  
laptopPriceAfterDiscount);  
        Clothing shirt = new Clothing("C001",  
"Shirt", 50);
```

```

        double shirtPriceAfterDiscount =
shirt.calculateDiscount("Winter");

        System.out.println("Clothing Price
after discount: $" +
shirtPriceAfterDiscount);
    }
}

```

The screenshot shows the OnlineGDB Java compiler interface. The main editor displays the following Java code:

```

1 class Product {
2     private String productId;
3     private String name;
4     private double price;
5     public Product(String productId, String name, double price) {
6         this.productId=productId;
7         this.name=name;
8         this.price=price;
9     }
10    public double getPrice() {
11        return price;
12    }
13 }
14 class Electronics extends Product {
15     public Electronics(String productId, String name, double price) {
16         super(productId, name, price);
17     }
18     public double calculateDiscount(boolean isMember) {
19         double discount=isMember ? 0.1 : 0.05;
20         return getPrice() * (1 - discount);
21     }
22 }
23 class Clothing extends Product {
24     public Clothing(String productId, String name, double price) {
25         super(productId, name, price);
26     }
27     public double calculateDiscount(String season) {
28         double discount=season.equals("Winter") ? 0.2 : 0.1;
29         return getPrice() * (1 - discount);
30     }
31 }

```

The output console shows the following results:

```

Electronics Price after discount: $900.0
Clothing Price after discount: $40.0
...Program finished with exit code 0
Press ENTER to exit console.

```

4.

```

class LibraryItem {
    private String title;
    private String author;

```

```
private int year;
```

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

```
    this.title = title;
```

```
    this.author = author;
```

```
    this.year = year;
```

```
}
```

```
public void checkOut() {
```

```
    System.out.println(title + " checked  
out.");
```

```
}
```

```
public void checkIn() {
```

```
    System.out.println(title + " checked  
in.");
```

```
}
```

```
public void displayInfo() {  
    System.out.println("Title: " + title);  
    System.out.println("Author: " +  
author);  
    System.out.println("Year: " + year);  
}  
}
```

```
class Book extends LibraryItem {  
    public Book(String title, String author,  
int year) {  
        super(title, author, year);  
    }  
}
```

@Override


```
public void displayInfo() {  
    super.displayInfo();  
    System.out.println("Type: Book");  
}  
}
```

```
class DVD extends LibraryItem {  
    public DVD(String title, String author, int  
year) {  
        super(title, author, year);  
    }  
}
```

@Override

```
public void displayInfo() {  
    super.displayInfo();  
    System.out.println("Type: DVD");  
}
```

```
    }  
}  
  
public class Main {  
    public static void main(String[] args) {  
        Book book = new Book("1984",  
"George Orwell", 1949);  
        book.checkOut();  
        book.displayInfo();  
        book.checkIn();  
        DVD dvd = new DVD("The Matrix",  
"The Wachowskis", 1999);  
        dvd.checkOut();  
        dvd.displayInfo();  
        dvd.checkIn();  
    }  
}
```

OnlineGDB
online compiler and debugger for c/c++

code compile run debug share

IDE

My Projects

Classroom new

Learn Programming

Programming Questions

Sign Up

Login

Run Debug Stop Share Save Beautify

Language: Java

Main.java

```
1 class LibraryItem {
2     private String title;
3     private String author;
4     private 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
11     public void checkOut() {
12         System.out.println(title + " checked out.");
13     }
14
15     public void checkIn() {
16         System.out.println(title + " checked in.");
17     }
18     public void displayInfo() {
19         System.out.println("Title: " + title);
20         System.out.println("Author: " + author);
21         System.out.println("Year: " + year);
22     }
23 }
24 class Book extends LibraryItem {
25     public Book(String title, String author, int year) {
26         super(title, author, year);
27     }
28 }
```

input

Author: The Wachowskis
Year: 1999
Type: DVD
The Matrix checked in.

About • FAQ • Blog • Terms of Use • Contact Us • GDB
Tutorial • Credits • Privacy
© 2016 - 2024 GDB Online

...Program finished with exit code 0
Press ENTER to exit console.