Міністерство освіти і науки України
Чернівецький національний університет імені Юрія Федьковича
Навчально-науковий інститут фізико-технічних та комп'ютерних наук
Кафедра математичних проблем управління і кібернетики

Звіт

про виконання лабораторної роботи № 3 з
дисципліни *«Програмування мовою Java»*

Виконала студентка групи 241Б:
Похілевич Анастасія
Перевірив: Кириченко
О.О.

Чернівці, 2025

Лабораторна робота №3

Завдання 1. *Побудувати ієрархію класів відповідно до варіанта завдання. Згідно завдання вибрати суперклас (базовий клас) та підкласи (похідні класи). В класах задати поля, які характерні для кожного класу. Для всіх класів розробити метод Show(), який виводить дані про об'єкт класу. Розробити програму, яка вводить інформацію про об'єкти заданих сутностей згідно варіанту в масив типу суперкласу та друкує введений масив (з використанням методу Show()).*
Варіант 10. Квитанція, накладна, документ, рахунок

Суперклас:

```
import java.time.LocalDate;

public class Document {
    protected String documentNumber; // Номер документа
    protected LocalDate documentDate;  // Дата створення
    protected String organization;     // Організація, що видала документ

    public Document(String documentNumber, LocalDate documentDate, String
organization) {
        this.documentNumber = documentNumber;
        this.documentDate = documentDate;
        this.organization = organization;
    }

   public void Show() {
        System.out.println("Номер документа: " + documentNumber);
        System.out.println("Дата документа: " + documentDate);
        System.out.println("Організація: " + organization);
    }
}
```

Підклас 1:

```
import java.time.LocalDate;
public class Receipt extends Document {
    private String payer;    // Платник
    private double amount;   // Сплачена сума
    private String service;  // Призначення платежу (послуга)

    public Receipt(String documentNumber, LocalDate documentDate, String
organization, String payer, double amount, String service) {
        // Виклик конструктора базового класу
        super(documentNumber, documentDate, organization);
        this.payer = payer;
        this.amount = amount;
        this.service = service;
    }
 @Override
    public void Show() {
        System.out.println("===== КВИТАНЦІЯ =====");
        super.Show(); // Виклик методу базового класу для друку спільних
полів
```

```
        System.out.println("Платник: " + payer);
        System.out.println("Послуга: " + service);
        // Форматуємо вивід суми до двох знаків після коми
        System.out.println("Сума: " + String.format("%.2f", amount) + "
грн");
    }
}
```

Підклас 2:

```
import java.time.LocalDate;
import java.util.Arrays;

public class Invoice extends Document {
    private String supplier; // Постачальник
    private String customer; // Отримувач
    private String[] items;  // Перелік товарів

    public Invoice(String documentNumber, LocalDate documentDate, String
organization, String supplier, String customer, String[] items) {
        // Виклик конструктора базового класу
        super(documentNumber, documentDate, organization);
        this.supplier = supplier;
        this.customer = customer;
        this.items = items;
    }
@Override
    public void Show() {
        System.out.println("===== НАКЛАДНА =====");
        super.Show(); // Виклик методу базового класу
        System.out.println("Постачальник: " + supplier);
        System.out.println("Отримувач: " + customer);
        System.out.println("Перелік товарів: " + Arrays.toString(items));
    }
}
```

Підклас 3:

```
import java.time.LocalDate;

public class Bill extends Document {
    private String client;         // Клієнт, якому виставлено рахунок
    private LocalDate dueDate;     // Термін оплати
    private double totalAmount;    // Сума до сплати

    public Bill(String documentNumber, LocalDate documentDate, String
organization, String client, LocalDate dueDate, double totalAmount) {
        // Виклик конструктора базового класу
        super(documentNumber, documentDate, organization);
        this.client = client;
        this.dueDate = dueDate;
        this.totalAmount = totalAmount;
```

```
    }

  @Override
    public void Show() {
        System.out.println("===== РАХУНОК =====");
        super.Show(); // Виклик методу базового класу
        System.out.println("Клієнт: " + client);
        System.out.println("Сума до сплати: " + String.format("%.2f",
totalAmount) + " грн");
        System.out.println("Сплатити до: " + dueDate);
    }
}
```

Программа виконання:

```
import java.time.LocalDate;

public class Main {
    public static void main(String[] args) {
        Document[] documents = new Document[3];

        documents[0] = new Receipt(
                "КВ-001",
                LocalDate.of(2025, 10, 16),
                "КП 'Водоканал'",
                "Іваненко І.І.",
                450.75,
                "Послуги водопостачання за вересень"
        );

        String[] itemsForInvoice = {"Ноутбук 'Legion'", "Миша 'HyperX'",
"Монітор 'Dell'"};
        documents[1] = new Invoice(
                "НК-2045",
                LocalDate.of(2025, 10, 15),
                "ТОВ 'Комп'ютерний Світ'",
                "ТОВ 'Комп'ютерний Світ'",
                "Петренко П.П.",
                itemsForInvoice
        );

        documents[2] = new Bill(
                "РАХ-987",
                LocalDate.of(2025, 10, 14),
                "ФОП Сидоренко В.В.",
                "ТОВ 'Будівельна компанія'",
                LocalDate.of(2025, 10, 31),
                25000.00
        );
        System.out.println("Друк інформації про всі документи в масиві:");
        System.out.println("==========================================");

        for (Document doc : documents) {
```

```
            doc.Show();
            System.out.println("----------------------------------------
-");
        }
    }
}
```

Результат:

```
Друк інформації про всі документи в масиві:
==========================================
===== КВИТАНЦІЯ =====
Номер документа: КВ-001
Дата документа: 2025-10-16
Організація: КП 'Водоканал'
Платник: Іваненко І.І.
Послуга: Послуги водопостачання за вересень
Сума: 450,75 грн
------------------------------------------
===== НАКЛАДНА =====
Номер документа: НК-2045
Дата документа: 2025-10-15
Організація: ТОВ 'Комп'ютерний Світ'
Постачальник: ТОВ 'Комп'ютерний Світ'
Отримувач: Петренко П.П.
Перелік товарів: [Ноутбук 'Legion', Миша 'HyperX', Монітор 'Dell']
------------------------------------------
===== РАХУНОК =====
Номер документа: РАХ-987
Дата документа: 2025-10-14
Організація: ФОП Сидоренко В.В.
Клієнт: ТОВ 'Будівельна компанія'
Сума до сплати: 25000,00 грн
Сплатити до: 2025-10-31
------------------------------------------
```

Пояснення: Ієрархія класів призначена для моделювання та обробки різних типів фінансових документів. Вона включає базовий клас Document, що містить спільні поля, як-от номер, дата та організація, і похідні класи: Receipt (Квитанція), Invoice (Накладна) та Bill (Рахунок), які додають власні унікальні атрибути. Кожен клас реалізує метод Show(), який виводить на екран повну інформацію про об’єкт, причому похідні класи розширюють функціональність базового методу для відображення своїх специфічних полів. Головна програма Main демонструє роботу ієрархії: створюється масив об’єктів базового типу Document, який наповнюється екземплярами різних похідних класів. При переборі масиву для кожного елемента викликається метод Show(), що наочно ілюструє механізм поліморфізму, коли для кожного об'єкта автоматично виконується його власна версія методу. Таким чином, робота програми показує ключові принципи об’єктно-орієнтованого підходу: успадкування (створення нових класів на основі існуючих), поліморфізм (однаковий

інтерфейс для об'єктів різних класів) та інкапсуляцію (об'єднання даних та методів в єдину структуру).

Завдання 2: *Реалізувати абстрактний базовий клас з вказаними абстрактними методами. Створити підкласи(похідні класи) суперкласу(базового класу), в яких здійснити реалізацію всіх абстрактних методів. Самостійно визначити, які поля необхідні і які з них визначити в базовому класі, а які – в похідних. В похідних класах мають бути перевантажені методи toString та equal. Створити масив об'єктів. Проілюструвати роботу всіх методів підкласів(похідних класів).*

Варіант 8. Створити абстрактний базовий клас Pair з абстрактними арифметичними операціями додавання, віднімання, множення і ділення на ціле число. Створити похідні класи Money (гроші) та Fraction (дріб).

Абстрактний базовий клас:

```
public abstract class Pair {
    protected long first;
    protected long second;
public Pair(long first, long second) {
        this.first = first;
        this.second = second;
    }
    /** Додавання іншої пари */
    public abstract Pair add(Pair other);

    /** Віднімання іншої пари */
    public abstract Pair subtract(Pair other);

    /** Множення на ціле число */
    public abstract Pair multiply(int factor);

    /** Ділення на ціле число */
    public abstract Pair divide(int divisor);
}
```

Похідний клас:

```
import java.util.Objects;

public class Money extends Pair {

    public Money(long hryvnia, long kopecks) {
        super(hryvnia, kopecks);
        normalize(); // Приводимо гроші до канонічного вигляду (напр., 1 грн
150 коп -> 2 грн 50 коп)
    }

private void normalize() {
        if (second >= 100) {
            first += second / 100;
            second %= 100;
```

```
        }
    }

    // Реалізація абстрактних методів

    @Override
    public Pair add(Pair other) {
        if (!(other instanceof Money)) {
            throw new IllegalArgumentException("Додавати можна лише об'єкти
одного типу (Money).");
        }
        Money otherMoney = (Money) other;
        long totalKopecks1 = this.first * 100 + this.second;
        long totalKopecks2 = otherMoney.first * 100 + otherMoney.second;
        long resultKopecks = totalKopecks1 + totalKopecks2;
        return new Money(resultKopecks / 100, resultKopecks % 100);
    }

    @Override
    public Pair subtract(Pair other) {
        if (!(other instanceof Money)) {
            throw new IllegalArgumentException("Віднімати можна лише об'єкти
одного типу (Money).");
        }
        Money otherMoney = (Money) other;
        long totalKopecks1 = this.first * 100 + this.second;
        long totalKopecks2 = otherMoney.first * 100 + otherMoney.second;
        long resultKopecks = totalKopecks1 - totalKopecks2;
        return new Money(resultKopecks / 100, resultKopecks % 100);
    }

    @Override
    public Pair multiply(int factor) {
        long totalKopecks = (this.first * 100 + this.second) * factor;
        return new Money(totalKopecks / 100, totalKopecks % 100);
    }

    @Override
    public Pair divide(int divisor) {
        if (divisor == 0) {
            throw new ArithmeticException("Ділення на нуль неможливе.");
        }
        long totalKopecks = (this.first * 100 + this.second) / divisor;
        return new Money(totalKopecks / 100, totalKopecks % 100);
    }

    // Перевантаження методів toString та equals

    @Override
    public String toString() {
        // Форматуємо, щоб копійки завжди мали 2 цифри (напр., 5 -> "05")
```

```
        return String.format("%d.%02d грн", first, second);
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true; // Чи це не той самий об'єкт?
        if (o == null || getClass() != o.getClass()) return false; // Чи
об'єкт не null і того ж класу?
        Money money = (Money) o; // Приводимо тип
        // Порівнюємо відповідні поля
        return first == money.first && second == money.second;
    }
}
```

Похідний клас 2:

```
public class Fraction extends Pair {

    public Fraction(long numerator, long denominator) {
        super(numerator, denominator);
        if (denominator == 0) {
            throw new IllegalArgumentException("Знаменник не може
дорівнювати нулю.");
        }
        simplify(); // Спрощуємо дріб одразу при створенні
    }
  private long gcd(long a, long b) {
        return b == 0 ? a : gcd(b, a % b);
    }
  private void simplify() {
        long commonDivisor = gcd(Math.abs(first), Math.abs(second));
        first /= commonDivisor;
        second /= commonDivisor;

        // Зберігаємо знак мінуса в чисельнику
        if (second < 0) {
            first = -first;
            second = -second;
        }
    }

    // Реалізація абстрактних методів

    @Override
    public Pair add(Pair other) {
        if (!(other instanceof Fraction)) {
            throw new IllegalArgumentException("Додавати можна лише об'єкти
одного типу (Fraction).");
        }
        Fraction otherFraction = (Fraction) other;
        long newNumerator = this.first * otherFraction.second +
otherFraction.first * this.second;
        long newDenominator = this.second * otherFraction.second;
```

```
        return new Fraction(newNumerator, newDenominator);
    }

    @Override
    public Pair subtract(Pair other) {
        if (!(other instanceof Fraction)) {
            throw new IllegalArgumentException("Віднімати можна лише об'єкти
одного типу (Fraction).");
        }
        Fraction otherFraction = (Fraction) other;
        long newNumerator = this.first * otherFraction.second -
otherFraction.first * this.second;
        long newDenominator = this.second * otherFraction.second;
        return new Fraction(newNumerator, newDenominator);
    }

    @Override
    public Pair multiply(int factor) {
        return new Fraction(this.first * factor, this.second);
    }

    @Override
    public Pair divide(int divisor) {
        if (divisor == 0) {
            throw new ArithmeticException("Ділення на нуль неможливе.");
        }
        return new Fraction(this.first, this.second * divisor);
    }

    // Перевантаження методів toString та equals

    @Override
    public String toString() {
        return first + "/" + second;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        Fraction fraction = (Fraction) o;
        // Оскільки дроби завжди спрощені, ми можемо просто порівняти поля
        return first == fraction.first && second == fraction.second;
    }
}
```

Програма виконання:

```
public class Task2 {
    public static void main(String[] args) {
        // Створюємо масив типу базового класу
        Pair[] pairs = new Pair[4];
```

```
        // Заповнюємо масив об'єктами похідних класів
        pairs[0] = new Money(120, 75); // 120.75 грн
        pairs[1] = new Money(50, 50);  // 50.50 грн
        pairs[2] = new Fraction(1, 2); // 1/2
        pairs[3] = new Fraction(4, 8); // стане 1/2 після спрощення

        System.out.println("--- Початковий стан масиву (демонстрація
toString) ---");
        for (Pair p : pairs) {
            System.out.println("Об'єкт: " + p.toString());
        }

        System.out.println("\n\n--- Демонстрація роботи з класом Money ---
");
        Money m1 = (Money) pairs[0];
        Money m2 = (Money) pairs[1];
        System.out.println("Сума " + m1 + " та " + m2 + " = " + m1.add(m2));
        System.out.println("Різниця " + m1 + " та " + m2 + " = " +
m1.subtract(m2));
        System.out.println("Множення " + m1 + " на 2 = " + m1.multiply(2));
        System.out.println("Ділення " + m2 + " на 5 = " + m2.divide(5));

        System.out.println("\n\n--- Демонстрація роботи з класом Fraction --
-");
        Fraction f1 = (Fraction) pairs[2];
        Fraction f2 = new Fraction(1, 3);
        System.out.println("Сума " + f1 + " та " + f2 + " = " + f1.add(f2));
        System.out.println("Різниця " + f1 + " та " + f2 + " = " +
f1.subtract(f2));
        System.out.println("Множення " + f2 + " на 3 = " + f2.multiply(3));
        System.out.println("Ділення " + f1 + " на 2 = " + f1.divide(2));

        System.out.println("\n\n--- Демонстрація роботи equals ---");
        Fraction f_from_array = (Fraction) pairs[3];
        System.out.println("Чи дорівнює дріб " + f1 + " дробу " +
f_from_array + "? -> " + f1.equals(f_from_array));
        System.out.println("Чи дорівнює дріб " + f1 + " дробу " + f2 + "? ->
" + f1.equals(f2));
        System.out.println("Чи дорівнює сума " + m1 + " самій собі? -> " +
m1.equals(m1));
    }
}
```

Результат:

```
--- Початковий стан масиву (демонстрація toString) ---
Об'єкт: 120.75 грн
Об'єкт: 50.50 грн
Об'єкт: 1/2
Об'єкт: 1/2


--- Демонстрація роботи з класом Money ---
Сума 120.75 грн та 50.50 грн = 171.25 грн
Різниця 120.75 грн та 50.50 грн = 70.25 грн
Множення 120.75 грн на 2 = 241.50 грн
Ділення 50.50 грн на 5 = 10.10 грн


--- Демонстрація роботи з класом Fraction ---
Сума 1/2 та 1/3 = 5/6
Різниця 1/2 та 1/3 = 1/6
Множення 1/3 на 3 = 1/1
Ділення 1/2 на 2 = 1/4


--- Демонстрація роботи equals ---
Чи дорівнює дріб 1/2 дробу 1/2? -> true
Чи дорівнює дріб 1/2 дробу 1/3? -> false
Чи дорівнює сума 120.75 грн самій собі? -> true
```

Пояснення: Ієрархія класів призначена для роботи з різними типами числових пар, що підтримують арифметичні операції. В її основі лежить абстрактний базовий клас Pair, який визначає загальний "контракт" — набір абстрактних методів для додавання, віднімання, множення та ділення. Похідні класи Money (гроші) та Fraction (дріб) успадковують цей клас і надають конкретну реалізацію абстрактних методів відповідно до своєї унікальної логіки: для грошей операції виконуються через переведення у копійки, а для дробів — за математичними правилами. У підкласах також перевантажені методи toString для зручного текстового представлення (наприклад, "125.50 грн" або "3/4") та equals для коректного порівняння об'єктів за їхнім значенням. Головна програма Task2 демонструє роботу ієрархії: створюється масив об'єктів типу Pair, який містить екземпляри обох підкласів, та ілюструється виконання арифметичних операцій для кожного з них. Робота програми наочно показує застосування ключових принципів ООП: абстракції (виділення загальної поведінки в базовому класі без конкретної реалізації), успадкування (розширення функціональності базового класу) та

поліморфізму (можливість працювати з об'єктами різних класів через єдиний інтерфейс).

Завдання 3. *Виконати попереднє завдання (Завдання 2) замінивши абстрактний клас інтерфейсом.*

Інтерфейс:

```
public interface Pair_task3 {

    Pair_task3 add(Pair_task3 other);
    Pair_task3 subtract(Pair_task3 other);
    Pair_task3 multiply(int factor);
    Pair_task3 divide(int divisor);

}
```

Клас 1:

```
public class Money_task3 implements Pair_task3 {
    private long first;  // гривні
    private long second; // копійки

    public Money_task3(long hryvnia, long kopecks) {
        this.first = hryvnia;
        this.second = kopecks;
        normalize();
    }

    private void normalize() {
        if (second >= 100) {
            first += second / 100;
            second %= 100;
        }
    }

    // Реалізація методів, визначених в інтерфейсі Pair_task3

   @Override
    public Pair_task3 add(Pair_task3 other) {
        if (!(other instanceof Money_task3)) {
            throw new IllegalArgumentException("Додавати можна лише об'єкти
типу Money_task3.");
        }
        Money_task3 otherMoney = (Money_task3) other;
        long totalKopecks1 = this.first * 100 + this.second;
        long totalKopecks2 = otherMoney.first * 100 + otherMoney.second;
        long resultKopecks = totalKopecks1 + totalKopecks2;
        return new Money_task3(resultKopecks / 100, resultKopecks % 100);
    }

    @Override
```

```
    public Pair_task3 subtract(Pair_task3 other) {
        if (!(other instanceof Money_task3)) {
            throw new IllegalArgumentException("Віднімати можна лише об'єкти
типу Money_task3.");
        }
        Money_task3 otherMoney = (Money_task3) other;
        long totalKopecks1 = this.first * 100 + this.second;
        long totalKopecks2 = otherMoney.first * 100 + otherMoney.second;
        long resultKopecks = totalKopecks1 - totalKopecks2;
        return new Money_task3(resultKopecks / 100, resultKopecks % 100);
    }

    @Override
    public Pair_task3 multiply(int factor) {
        long totalKopecks = (this.first * 100 + this.second) * factor;
        return new Money_task3(totalKopecks / 100, totalKopecks % 100);
    }

    @Override
    public Pair_task3 divide(int divisor) {
        if (divisor == 0) throw new ArithmeticException("Ділення на нуль.");
        long totalKopecks = (this.first * 100 + this.second) / divisor;
        return new Money_task3(totalKopecks / 100, totalKopecks % 100);
    }

    // Перевантаження стандартних методів

    @Override
    public String toString() {
        return String.format("%d.%02d грн", first, second);
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        Money_task3 money = (Money_task3) o;
        return first == money.first && second == money.second;
    }
}
```

Клас 2:

```
public class Fraction_task3 implements Pair_task3 {

    private long first;  // чисельник
    private long second; // знаменник

    public Fraction_task3(long numerator, long denominator) {
        if (denominator == 0) {
            throw new IllegalArgumentException("Знаменник не може бути
```

```
нулем.");
        }
        this.first = numerator;
        this.second = denominator;
        simplify();
    }

    private long gcd(long a, long b) {
        return b == 0 ? a : gcd(b, a % b);
    }

    private void simplify() {
        long commonDivisor = gcd(Math.abs(first), Math.abs(second));
        first /= commonDivisor;
        second /= commonDivisor;
        if (second < 0) {
            first = -first;
            second = -second;
        }
    }

    // Реалізація методів, визначених в інтерфейсі Pair_task3

    @Override
    public Pair_task3 add(Pair_task3 other) {
        if (!(other instanceof Fraction_task3)) {
            throw new IllegalArgumentException("Додавати можна лише об'єкти
типу Fraction_task3.");
        }
        Fraction_task3 otherFraction = (Fraction_task3) other;
        long newNumerator = this.first * otherFraction.second +
otherFraction.first * this.second;
        long newDenominator = this.second * otherFraction.second;
        return new Fraction_task3(newNumerator, newDenominator);
    }

    @Override
    public Pair_task3 subtract(Pair_task3 other) {
        if (!(other instanceof Fraction_task3)) {
            throw new IllegalArgumentException("Віднімати можна лише об'єкти
типу Fraction_task3.");
        }
        Fraction_task3 otherFraction = (Fraction_task3) other;
        long newNumerator = this.first * otherFraction.second -
otherFraction.first * this.second;
        long newDenominator = this.second * otherFraction.second;
        return new Fraction_task3(newNumerator, newDenominator);
    }

    @Override
    public Pair_task3 multiply(int factor) {
```

```
        return new Fraction_task3(this.first * factor, this.second);
    }

    @Override
    public Pair_task3 divide(int divisor) {
        if (divisor == 0) throw new ArithmeticException("Ділення на нуль.");
        return new Fraction_task3(this.first, this.second * divisor);
    }

    // Перевантаження стандартних методів

    @Override
    public String toString() {
        return first + "/" + second;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        Fraction_task3 fraction = (Fraction_task3) o;
        return first == fraction.first && second == fraction.second;
    }
}
```

Клас 3:

```
public class Money_task3 implements Pair_task3 {
    private long first;  // гривні
    private long second; // копійки

    public Money_task3(long hryvnia, long kopecks) {
        this.first = hryvnia;
        this.second = kopecks;
        normalize();
    }

    private void normalize() {
        if (second >= 100) {
            first += second / 100;
            second %= 100;
        }
    }

    // Реалізація методів, визначених в інтерфейсі Pair_task3

    @Override
    public Pair_task3 add(Pair_task3 other) {
        if (!(other instanceof Money_task3)) {
            throw new IllegalArgumentException("Додавати можна лише об'єкти
типу Money_task3.");
```

```
        }
        Money_task3 otherMoney = (Money_task3) other;
        long totalKopecks1 = this.first * 100 + this.second;
        long totalKopecks2 = otherMoney.first * 100 + otherMoney.second;
        long resultKopecks = totalKopecks1 + totalKopecks2;
        return new Money_task3(resultKopecks / 100, resultKopecks % 100);
    }

    @Override
    public Pair_task3 subtract(Pair_task3 other) {
        if (!(other instanceof Money_task3)) {
            throw new IllegalArgumentException("Віднімати можна лише об'єкти
типу Money_task3.");
        }
        Money_task3 otherMoney = (Money_task3) other;
        long totalKopecks1 = this.first * 100 + this.second;
        long totalKopecks2 = otherMoney.first * 100 + otherMoney.second;
        long resultKopecks = totalKopecks1 - totalKopecks2;
        return new Money_task3(resultKopecks / 100, resultKopecks % 100);
    }

    @Override
    public Pair_task3 multiply(int factor) {
        long totalKopecks = (this.first * 100 + this.second) * factor;
        return new Money_task3(totalKopecks / 100, totalKopecks % 100);
    }

    @Override
    public Pair_task3 divide(int divisor) {
        if (divisor == 0) throw new ArithmeticException("Ділення на нуль.");
        long totalKopecks = (this.first * 100 + this.second) / divisor;
        return new Money_task3(totalKopecks / 100, totalKopecks % 100);
    }

    // Перевантаження стандартних методів

    @Override
    public String toString() {
        return String.format("%d.%02d грн", first, second);
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        Money_task3 money = (Money_task3) o;
        return first == money.first && second == money.second;
    }
}
```

Програма виконання:

```
public class Task3 {
    public static void main(String[] args) {
        Pair_task3[] pairs = new Pair_task3[4];

        // Заповнюємо масив об'єктами класів, що реалізують інтерфейс
        pairs[0] = new Money_task3(250, 15); // 250.15 грн
        pairs[1] = new Money_task3(100, 90);  // 100.90 грн
        pairs[2] = new Fraction_task3(2, 3); // 2/3
        pairs[3] = new Fraction_task3(10, 15); // стане 2/3 після спрощення

        System.out.println("--- Початковий стан масиву (демонстрація
toString) ---");
        for (Pair_task3 p : pairs) {
            System.out.println("Об'єкт: " + p.toString());
        }

        System.out.println("\n\n--- Демонстрація роботи з класом Money_task3
---");
        // Потрібне явне приведення типів, щоб викликати методи
        Money_task3 m1 = (Money_task3) pairs[0];
        Money_task3 m2 = (Money_task3) pairs[1];
        System.out.println("Сума " + m1 + " та " + m2 + " = " + m1.add(m2));
        System.out.println("Різниця " + m1 + " та " + m2 + " = " +
m1.subtract(m2));
        System.out.println("Множення " + m1 + " на 3 = " + m1.multiply(3));
        System.out.println("Ділення " + m2 + " на 10 = " + m2.divide(10));

        System.out.println("\n\n--- Демонстрація роботи з класом
Fraction_task3 ---");
        Fraction_task3 f1 = (Fraction_task3) pairs[2];
        Fraction_task3 f2 = new Fraction_task3(1, 4);
        System.out.println("Сума " + f1 + " та " + f2 + " = " + f1.add(f2));
        System.out.println("Різниця " + f1 + " та " + f2 + " = " +
f1.subtract(f2));
        System.out.println("Множення " + f1 + " на 5 = " + f1.multiply(5));
        System.out.println("Ділення " + f2 + " на 2 = " + f2.divide(2));

        System.out.println("\n\n--- Демонстрація роботи equals ---");
        Fraction_task3 f_from_array = (Fraction_task3) pairs[3];
        System.out.println("Чи дорівнює дріб " + f1 + " дробу " +
f_from_array + "? -> " + f1.equals(f_from_array));
        System.out.println("Чи дорівнює сума " + m1 + " сумі " + m2 + "? ->
" + m1.equals(m2));
    }
}
```

Результат:

```
--- Початковий стан масиву (демонстрація toString) ---
Об'єкт: 250.15 грн
Об'єкт: 100.90 грн
Об'єкт: 2/3
Об'єкт: 2/3


--- Демонстрація роботи з класом Money_task3 ---
Сума 250.15 грн та 100.90 грн = 351.05 грн
Різниця 250.15 грн та 100.90 грн = 149.25 грн
Множення 250.15 грн на 3 = 750.45 грн
Ділення 100.90 грн на 10 = 10.09 грн


--- Демонстрація роботи з класом Fraction_task3 ---
Сума 2/3 та 1/4 = 11/12
Різниця 2/3 та 1/4 = 5/12
Множення 2/3 на 5 = 10/3
Ділення 1/4 на 2 = 1/8


--- Демонстрація роботи equals ---
Чи дорівнює дріб 2/3 дробу 2/3? -> true
Чи дорівнює сума 250.15 грн сумі 100.90 грн? -> false
```

Пояснення: Структура класів призначена для роботи з різними типами числових пар, об'єднаних спільним набором арифметичних операцій, з використанням інтерфейсу для визначення загальної поведінки. В її основі лежить інтерфейс Pair_task3, який виступає в ролі "контракту", що лише оголошує набір абстрактних методів (add, subtract тощо), не маючи власних полів чи реалізації. Класи Money_task3 та Fraction_task3 реалізують (implements) цей інтерфейс. Вони самостійно визначають необхідні поля для зберігання даних (гривні/копійки та чисельник/знаменник) і надають конкретну логіку для кожного з методів, вимаганих контрактом. Також у цих класах перевантажено методи toString для зручного відображення та equals для коректного порівняння. Головна програма Task3 демонструє створення масиву типу інтерфейсу, який наповнюється об'єктами обох реалізуючих класів, ілюструючи гнучкість такого підходу. Робота програми демонструє потужний механізм абстракції через інтерфейси, що дозволяє повністю відокремити специфікацію поведінки від її конкретної реалізації. Вона також є яскравим прикладом поліморфізму, де об'єкти різних класів, що реалізують спільний інтерфейс, можуть оброблятися уніфікованим способом, забезпечуючи високу гнучкість та розширюваність коду.