Лабораторна робота №3

Тема: Успадкування, поліморфізм, інкапсуляція

Завдання

Створити консольну програму, що задовольняє таким вимогам:

- Використовує можливості ООП: класи, успадкування, поліморфізм, інкапсуляція.
- Кожен клас повинен мати назву, що відповідає його змісту, та складається з полів та методів, указаних у завданні.
- У базовому класі та його підкласах визначте додаткові поля та методи, що потрібні згідно з предметною областю.
- Під час кодування мають бути використані домовленості про оформлення коду java code convention.
- Класи мають бути грамотно розкладені за пакетами.
- Консольне меню має бути мінімальним.

Розроблена програма повинна супроводжуватися діаграмою класів.

1. Тварини

Створити абстрактний клас **Animal** і класи **Dog**, **Cat**, **Hamster**, що його розширюють.

Клас **Animal** містить змінну **name** та абстрактні методи **makeNoise**, **eat**, **getDescription**. Метод **makeNoise**, наприклад, повертає опис звуків, що видає тварина. Метод **eat** повертає масив того, чим харчується ця тварина. Метод **getDescription** повертає опис тварини.

Dog, **Cat**, **Hamster** перевизначають методи **makeNoise**, **eat**, **getDescription**.

Створіть клас **Ветеринар**, у якому визначте метод void **treatAnimal**(Animal animal). Нехай цей метод виводить **name** та опис тварини, яка прийшла на консультацію.

У методі **main** створіть масив типу **Animal**, у який запишіть тварин усіх наявних у вас типів. У циклі відправляйте їх на консультацію до ветеринара. В окремому циклі викличте методи **makeNoise**, **eat** для кожної тварини.

2. Лікарі

Створити абстрактний клас **Doctor** і класи **Therapist**, **Surgeon**, **Ophthalmologist**, що його розширюють.

Клас **Doctor** містить змінну **name** та абстрактні методи **cure**, **consult**, **getDescription**. Метод **cure**, наприклад, повертає способи та засоби лікування. Метод **consult** повертає масив хвороб, які діагностує цей доктор. Метод **getDescription** повертає опис задач доктора.

Therapist, **Surgeon**, **Ophthalmologist** перевизначають методи **cure**, **consult**, **getDescription**.

Створіть клас **Patient**, у якому визначте метод void **visit**(Doctor doctor). Нехай цей метод виводить **name** та опис доктора, до якого він прийшов на консультацію.

У методі **main** створіть масив типу **Doctor**, у який запишіть лікарів усіх наявних у вас типів. У циклі відправляйте до них на консультацію пацієнта. В окремому циклі викличте методи **consult**, **cure** для кожного лікаря.

3. Музичні інструменти

Створити абстрактний клас **Instrument** і класи **Piano**, **Guitar**, **Drum**, що його розширюють.

Клас **Instrument** містить змінну **name** та абстрактні методи **play**, **type**, **getDescription**. Метод **play**, наприклад, повертає рядок нот, супроводжуючи його назвою інструмента. Метод **type** повертає тип отримання звуку з інструменту (струнний, духовий, ударний). Метод **getDescription** повертає опис інструмента.

Piano, **Guitar**, **Drum** перевизначають методи **play**, **type**, **getDescription**.

Створіть клас **Musician**, у якому визначте метод void **use**(Instrument instrument). Нехай цей метод виводить **name** та опис інструмента, на якому він грає.

У методі **main** створіть масив типу **Instrument**, у який запишіть інструменти всіх наявних у вас типів. У циклі передавайте їх музиканту для виконання мелодії. В окремому циклі викличте методи **play**, **type** для кожного інструмента.

4. Фігури на площині

Створити абстрактний клас **Figure** і класи **Circle**, **Square**, **Triangle**, що його розширюють.

Клас **Figure** містить змінну **name** та абстрактні методи **perimeter**, **area**, **getDescription**. Метод **perimeter** повинен повертати периметр фігури. Метод **area** повертає площу фігури. Метод **getDescription** повертає опис фігури.

Circle, **Square**, **Triangle** перевизначають методи **perimeter**, **area**, **getDescription**.

Створіть клас **Picture**, у якому визначте метод void **draw**(Figure figure). Нехай цей метод виводить **name** та опис фігури.

У методі **main** створіть масив типу **Figure**, у який запишіть фігури усіх наявних у вас типів. У циклі передавайте їх у Picture для малювання (draw). В окремому циклі викличте методи **perimeter**, **area** для кожної фігури.

5. Залізничні вагони

Створити абстрактний клас **RailCar** і класи **PostCar** (поштовий вагон), **PassengerCar** (пасажирський вагон), **AnimalCar** (вагон для тварин).

Клас **RailCar** містить змінну **number** та абстрактні методи **cargo**, **volume**, **getDescription**. Метод **cargo** повертає тип вантажу (поштовий вантаж, люди, тварини). Метод **volume** повертає кількість вантажу. Метод **getDescription** повертає опис вагона та всі його характеристики.

Класи **PostCar**, **PassengerCar**, **AnimalCar** перевизначають методи **cargo**, **volume**, **getDescription**.

Створіть клас **Checker**, у якому визначте метод **check**(RailCar railCar). Нехай цей метод виводить **number** та опис вагона.

У методі **main** створіть масив типу **RailCar**, у який запишіть вагони всіх наявних у вас типів. У циклі передавайте їх у Checker для перевірки (check). В окремому циклі викличте методи **cargo**, **volume** для кожного вагона.

6. Прикраси

Створити абстрактний клас **Jewelry** і класи **Necklace** (намисто), **Earring** (сережка), **Ring** (каблучка).

Клас **Jewelry** містить змінну **nameProduct** та абстрактні методи **price**, **size**, **getDescription**.

Метод **price** повертає вартість прикраси, наприклад, залежно від сплаву та наявності коштовного каміння тощо. Метод **size** повертає розмір, (для намиста — це довжина, для каблучки розмір визначається за діаметром, для сережки — це довжина). Метод **getDescription** повертає опис прикраси та всі її характеристики.

Класи **Necklace**, **Earring**, **Ring** перевизначають методи **price**, **size**, **getDescription**.

Створіть клас **Customer**, у якому визначте метод **jewelryOrder**(Jewelry jewelry). Нехай цей метод виводить **nameProduct** та опис прикраси.

У методі **main** створіть масив типу **Jewelry**, у який запишіть прикраси всіх наявних у вас типів. У циклі передавайте їх у Customer для замовлення (jewelryOrder). В окремому циклі викличте методи **price**, **size** для кожної прикраси.

7. Таксопарк

Створити абстрактний клас **Vehicle** і класи **Car**, **Truck**, **Minivan**.

Клас **Vehicle** містить змінну **number** та абстрактні методи **passengers**, **speed**, **getDescription**.

Метод **passengers** повертає максимальну кількість пасажирів (для **Car** — 4, для **Truck** — 0, для **Minivan** — число вказане в конструкторі). Метод **speed** повертає максимальну

швидкість автівки. Метод **getDescription** повертає опис автівки та всі її характеристики.

Класи **Car**, **Truck**, **Minivan** перевизначають методи **cargo**, **volume**, **getDescription**.

Створіть клас **Customer**, у якому визначте метод **placeOrder**(Vehicle vehicle). Нехай цей метод виводить **number** та опис автівки.

У методі **main** створіть масив типу **Vehicle**, у який запишіть автівки всіх наявних у вас типів. У циклі передавайте їх у Customer для замовлення (placeOrder). В окремому циклі викличте методи **passengers**, **speed** для кожної автівки.

8. Меблі

Створити абстрактний клас **Furniture** і класи **Wardrobe**, **Sofa**, **Bookcase**, що його розширюють.

Клас **Furniture** містить змінну **name** та абстрактні методи **quantity**, **sizes**, **getDescription**.

Метод **quantity** повертає деяку числову характеристику: (для **Wardrobe** — кількість дверей, для **Sofa** — кількість спальних місць, для **Bookcase** — кількість поличок). Метод **sizes** повертає габарити. Метод **getDescription** повертає опис предмета та його характеристики.

Класи **Wardrobe**, **Sofa**, **Bookcase** перевизначають методи **quantity**, **sizes**, **getDescription**.

Створіть клас **Designer**, у якому визначте метод **design**(Furniture furniture). Нехай цей метод виводить **name** та опис предмета.

У методі **main** створіть масив типу **Furniture**, у який запишіть меблі всіх наявних у вас типів. У циклі передавайте їх у Designer для проєктування (design). В окремому циклі викличте методи **quantity**, **sizes** для кожного предмета.

9. Дитячі іграшки

Створити абстрактний клас **Toy** і класи **Doll**, **Ball**, **Constructor**, що його розширюють.

Клас **Toy** містить змінну **name** та абстрактні методи **size**, **age**, **price**, **getDescription**.

Метод **size** повертає деяку числову характеристику: (для **Doll** — зріст, для **Ball** — діаметр, для **Constructor** — кількість деталей). Метод **age** повертає діапазон віку дитини. Метод **price** повертає вартість іграшки, що залежить від її інших характеристик. Метод **getDescription** повертає опис іграшки та її характеристики.

Класи **Doll**, **Ball**, **Constructor** перевизначають методи **size**, **age**, **price**, **getDescription**.

Створіть клас **Child**, у якому визначте метод **play**(Toy toy). Нехай цей метод виводить **name** та опис іграшки.

У методі **main** створіть масив типу **Toy**, у який запишіть іграшки усіх наявних у вас типів. У циклі передавайте їх у Child для гри (play). В окремому циклі викличте методи **size**, **age**, **price** для кожної іграшки.

10. Парк розваг

Створити абстрактний клас **Amusement** і класи **FerrisWheel**, **RollerCoaster**, **PanicRoom**, що його розширюють.

Клас **Amusement** містить змінну **name** та абстрактні методи **age**, **price**, **getDescription**.

Метод **age** повертає мінімальний вік людини, що допускається до атракціону. Метод **price** повертає вартість розваги, що залежить від її інших характеристик. Метод **getDescription** повертає опис атракціону та його характеристики.

Класи **FerrisWheel**, **RollerCoaster**, **PanicRoom** перевизначають методи **age**, **price**, **getDescription**.

Створіть клас **Visitor**, у якому визначте метод **haveFun**(Amusement amusement). Нехай цей метод виводить **name** та опис атракціону.

У методі **main** створіть масив типу **Amusement**, у який запишіть атракціони всіх наявних у вас типів. У циклі передавайте їх у Visitor для розваги (haveFun). В окремому циклі викличте методи **age**, **price** для кожного атракціону.

Example. Птахи

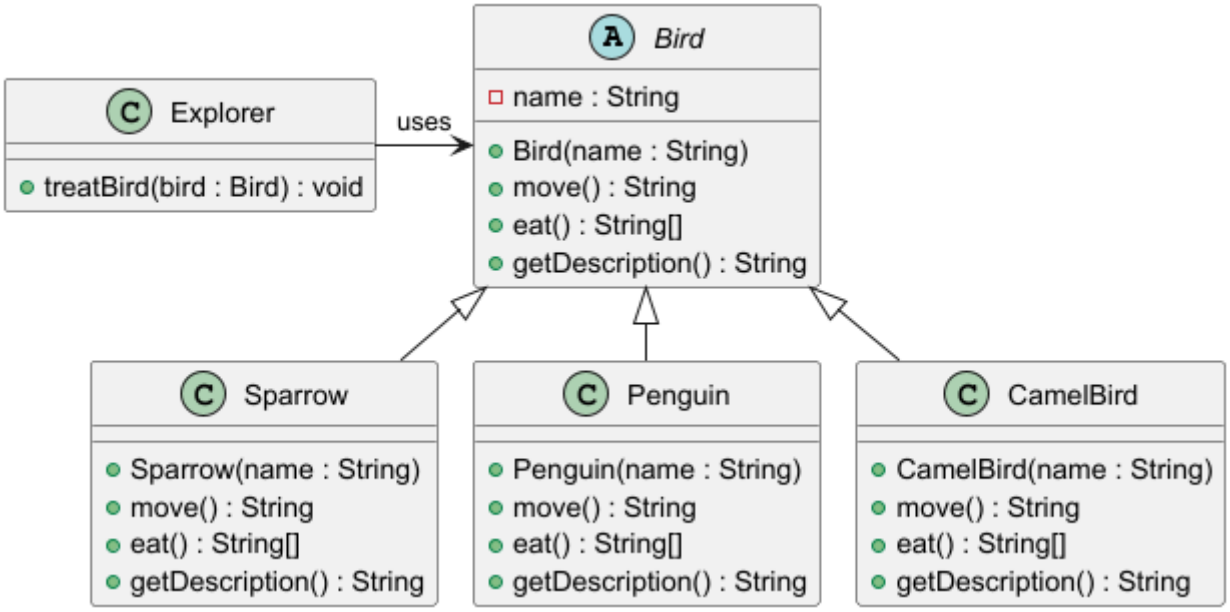
Створити абстрактний клас **Bird** і класи **Sparrow**, **Penguin**, **CamelBird**, що його розширюють.

Клас **Bird** містить змінну **name** та абстрактні методи **move**, **eat**, **getDescription**. Метод **move**, наприклад, повертає спосіб руху птаха (біжить, летить, ходить). Метод **eat** повертає масив того, чим харчується цей птах. Метод **getDescription** повертає опис тварини.

Sparrow, **Penguin**, **CamelBird** перевизначають методи **move**, **eat**, **getDescription**.

Створіть клас **Explorer**, у якому визначте метод void **treatBird**(Bird bird). Нехай цей метод виводить **name** та опис птаха, якого він побачив.

У методі **main** створіть масив типу **Bird**, у який запишіть птахів усіх наявних у вас типів. У циклі відправляйте їх на перегляд дослідникові. В окремому циклі викличте методи **move**, **eat** для кожного птаха.



Абстрактний клас Bird

```
abstract class Bird {
    protected String name;

    public Bird(String name) {
        this.name = name;
    }

    public abstract String move();
    public abstract String[] eat();
    public abstract String getDescription();

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

Клас Sparrow

```
class Sparrow extends Bird {

    public Sparrow(String name) {
        super(name);
    }

    @Override
    public String move() {
        return "летить";
    }

    @Override
    public String[] eat() {
        return new String[]{"зерно", "комахи"};
    }

    @Override
    public String getDescription() {
        return "Горобець – маленький птах, що добре літає.";
    }
}
```

Клас Penguin

```
class Penguin extends Bird {

    public Penguin(String name) {
        super(name);
    }

    @Override
    public String move() {
        return "ходить та плаває";
    }

    @Override
    public String[] eat() {
        return new String[]{"риба", "креветки"};
    }

    @Override
    public String getDescription() {
        return "Пінгвін – нелітний птах, чудово плаває.";
    }
}
```

Клас CamelBird

```
class CamelBird extends Bird {

    public CamelBird(String name) {
        super(name);
    }

    @Override
    public String move() {
        return "бігає";
    }

    @Override
    public String[] eat() {
        return new String[]{"листя", "трава", "зерно"};
    }

    @Override
    public String getDescription() {
        return "Страус (CamelBird) — найбільший птах, що швидко бігає.";
    }
}
```

Клас Explorer

```
class Explorer {

    public void treatBird(Bird bird) {
        System.out.println("Дослідник бачить птаха: " + bird.getName());
        System.out.println("Опис: " + bird.getDescription());
        System.out.println();
    }
}
```

Головний клас

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

        Bird[] birds = {
            new Sparrow("Джек"),
            new Penguin("Пінго"),
            new CamelBird("Степан")
        };

        Explorer explorer = new Explorer();

        System.out.println("=== Перегляд птахів дослідником ===");
        for (Bird bird : birds) {
            explorer.treatBird(bird);
        }

        System.out.println("=== Перевірка поведінки птахів ===");
        for (Bird bird : birds) {
            System.out.println(bird.getName() + " " + bird.move());
            System.out.print("Харчування: ");
            for (String food : bird.eat()) {
                System.out.print(food + " ");
            }
            System.out.println("\n");
        }
    }
}
```