Министерство образования Республики Беларусь
Учреждение образования
«Брестский государственный технический университет»
Кафедра ИИТ

Лабораторная работа №5

По дисциплине: «Современные платформы программирования»

Выполнил:
студент 3 курса
группы ПО-8
Сорока В.С.

Проверил:
Крощенко А.А.

Брест, 2024

Цель работы: приобрести практические навыки в области объектно-ориентированного проектирования

Задание 1 (Вариант 9)
Реализовать абстрактные классы или интерфейсы, а также наследование и полиморфизм для следующих классов:

interface Корабль ← class Грузовой Корабль ← class Танкер.

Спецификации ввода-вывода программы:
1. Программа создает интерфейс Ship для представления корабля.
2. Реализует два класса: CargoShip и Tanker, оба реализующих интерфейс Ship.
3. CargoShip имеет метод loadCargo(), который загружает груз, а Tanker имеет метод loadOil(), который загружает нефть.
4. В методе main происходит демонстрация полиморфизма, создавая объекты разных классов через интерфейс Ship.
5. Вызываются методы sail(), общий для всех кораблей, и методы, специфичные для каждого типа корабля (loadCargo() и loadOil()).
6. Программа выводит информацию о действиях каждого корабля на экран.

Текст программы

```
// Определяем интерфейс Корабль
interface Ship {
    void sail();
}

// Реализация интерфейса Корабль: класс ГрузовойКорабль
class CargoShip implements Ship {
    @Override
    public void sail() {
        System.out.println("Грузовой корабль плывет");
    }

    // Метод специфичный для грузового корабля
    public void loadCargo() {
        System.out.println("Грузовой корабль загружает груз");
    }
}

// Реализация интерфейса Корабль: класс Танкер
class Tanker implements Ship {
    @Override
    public void sail() {
        System.out.println("Танкер плывет");
    }

    // Метод специфичный для танкера
    public void loadOil() {
        System.out.println("Танкер загружает нефть");
    }
}

public class Main {
    public static void main(String[] args) {
        // Полиморфизм: создаем объекты разных классов, но через одинаковый
интерфейс
```

```
        Ship ship1 = new CargoShip();
        Ship ship2 = new Tanker();

        // Вызываем общий метод для всех кораблей
        ship1.sail();
        ship2.sail();

        // Вызываем методы, специфичные для каждого типа корабля
        ((CargoShip) ship1).loadCargo(); // Приведение типов, так как ship1
объявлен как Ship
        ((Tanker) ship2).loadOil(); // Приведение типов, так как ship2
объявлен как Ship
    }
}
```

Пример работы программы

```
Грузовой корабль плывет
Танкер плывет
Грузовой корабль загружает груз
Танкер загружает нефть
```

Задание 2 (Вариант 6)

Требуется создать суперкласс (абстрактный класс, интерфейс) и определить общие методы для данного класса. Создать подклассы, в которых добавить специфические свойства и методы. Часть методов переопределить. Создать массив объектов суперкласса и заполнить объектами подклассов. Объекты подклассов идентифицировать конструктором по имени или идентификационному номеру. Использовать объекты подклассов для моделирования реальных ситуаций и объектов.

Создать суперкласс Домашнее животное и подклассы Собака, Кошка, Попугай. С помощью конструктора установить имя каждого животного и его характеристики.

Спецификации ввода-вывода программы:

- Программа создает массив объектов суперкласса Pet и заполняет его объектами подклассов Dog, Cat, и Parrot.
- Каждый объект выводится с его именем, типом звука, который он издает (определяется методом makeSound()), и типичным движением (определяется методом move()).
- Методы makeSound() и move() переопределены в каждом подклассе Dog, Cat, и Parrot для предоставления специфического поведения для каждого типа животного.
- Нет ввода данных извне программы. Все данные для создания объектов передаются непосредственно в конструкторы классов при создании объектов типа Dog, Cat, и Parrot.
- Для каждого созданного объекта выводятся информация о его имени, звук, который он издает, и его типичном движении.

Текст программы

```
// Абстрактный класс Домашнее животное
abstract class Pet {
    private String name;

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

    public String getName() {
        return name;
    }

    // Общий метод для всех животных
    public void makeSound() {
        System.out.println("Животное издает звук.");
    }

    // Абстрактный метод для подклассов
    public abstract void move();
}

// Подкласс Собака
class Dog extends Pet {
    private String breed;

    public Dog(String name, String breed) {
        super(name);
        this.breed = breed;
    }

    // Переопределение метода
    @Override
    public void makeSound() {
        System.out.println("Собака лает: Гав-гав!");
    }

    // Переопределение метода
    @Override
    public void move() {
        System.out.println("Собака бегает по двору.");
    }
}

// Подкласс Кошка
class Cat extends Pet {
    private int age;

    public Cat(String name, int age) {
        super(name);
        this.age = age;
    }

    // Переопределение метода
    @Override
    public void makeSound() {
        System.out.println("Кошка мурлычет: Мур-мур-мур");
    }

    // Переопределение метода
    @Override
    public void move() {
        System.out.println("Кошка прыгает по кухонному столу.");
    }
}
```

```java
// Подкласс Попугай
class Parrot extends Pet {
    private String color;

    public Parrot(String name, String color) {
        super(name);
        this.color = color;
    }

    // Переопределение метода
    @Override
    public void makeSound() {
        System.out.println("Попугай говорит: Валюша хороши, Валюша молодец");
    }

    // Переопределение метода
    @Override
    public void move() {
        System.out.println("Попугай висит вниз головой на веточке в
клетке.");
    }
}

public class Main {
    public static void main(String[] args) {
        // Создаем массив объектов суперкласса Pet
        Pet[] pets = new Pet[3];

        // Заполняем массив объектами подклассов
        pets[0] = new Dog("Буран", "Немецкая овчарка");
        pets[1] = new Cat("Ника", 3);
        pets[2] = new Parrot("Кеша", "Пёстрый зелёный");

        // Используем объекты для моделирования ситуаций
        for (Pet pet : pets) {
            System.out.println("Имя: " + pet.getName());
            pet.makeSound();
            pet.move();
            System.out.println();
        }
    }
}
```

Пример работы программы:

```
Имя: Буран
Собака лает: Гав-гав!
Собака бегает по двору.

Имя: Ника
Кошка мурлычет: Мур-мур-мур
Кошка прыгает по кухонному столу.

Имя: Кеша
Попугай говорит: Валюша хороши, Валюша молодец
Попугай висит вниз головой на веточке в клетке.
```

Задание 3 (Задание №3 ЛР 4, Вариант 11)

В задании 3 ЛР №4, где возможно, заменить объявления суперклассов объявлениями абстрактных классов или интерфейсов.

Спецификация ввода-вывода
Интерфейсы:
1. Airport (Аэропорт):
Методы:
- getName(): Возвращает название аэропорта.
- changeDestination(String newDestination): Изменяет место назначения рейса на newDestination.
- setDestination(Airport newDestination): Изменяет место назначения рейса на newDestination.
- Хранит информацию о названии аэропорта.
- Позволяет изменять место назначения рейса.

2.Pilot (Пилот):
- Является производным классом от CrewMember.
Методы:
- flyAircraft(): Управляет самолетом.
- getVoice(): Объявляет информацию пассажирам.
- Определяет пилотов, способных управлять самолетом и объявлять информацию пассажирам.

Абстрактные классы:
1. Aircraft (Самолет):
- capacity: Вместимость самолета.
- range: Дальность полета самолета.
- model: Модель самолета.
- currentLocation: Текущее местоположение самолета.
Методы:
- flyTo(Airport destination): Выполняет перелет в destination.
- isAtDestination(Airport destination): Проверяет, находится ли самолет в destination.
- Хранит информацию о вместимости, дальности полета, модели самолета и его текущем местоположении.
- Позволяет моделировать перемещение самолета между аэропортами.

2. CrewMember (Член экипажа):
- name: Имя члена экипажа.
- age: Возраст члена экипажа.
- role: Роль члена экипажа (пилот, штурман, радист, стюардесса).
- Базовый класс для всех членов экипажа, хранит информацию об их имени, возрасте и роли.

3. Navigator (Штурман):
- Является производным классом от CrewMember.
Методы:
- navigate(Airport origin, Airport destination): Прокладывает маршрут полета из origin в destination.
- Определяет штурманов, способных прокладывать маршрут полета.

4. RadioOperator (Радист):
- Является производным классом от CrewMember.
Методы:
- speakWithEarth(String message): Отправляет сообщение диспетчерскому центру.
- receiveMessage(String message): Получает сообщение от диспетчерского центра.
- Определяет радистов, способных поддерживать связь с диспетчерским центром.

5. Stewardess (Стюардесса):
- Является производным классом от CrewMember.
Методы:
- servePassengers(): Обслуживает пассажиров.
- conductBriefing(): Проводит инструктаж пассажиров.
- Определяет стюардесс, способных обслуживать пассажиров и проводить инструктаж.

6. Flight (Рейс):
- Содержит информацию о пункте отправления, пункте назначения и самолете, выполняющем рейс.
- Метод changeDestination(String newDestination) позволяет изменить пункт назначения рейса.
- Метод getDestinationAirport() возвращает объект Airport, представляющий текущий пункт назначения.

7. Main
- Создание объектов классов Aircraft, Airport, Pilot, Navigator, RadioOperator и Stewardess.
- Формирование экипажа рейса.
- Вывод информации о рейсе.
- Определение погодных условий.
- Выполнение рейса в зависимости от погодных условий

Методы:
1. getJob
- Моделирует выполнение этапов рейса:
- Управление самолетом пилотами.
- Навигация штурманом.
- Связь с диспетчерской службой радиооператором.

- Обслуживание пассажиров стюардессами.
- Общая задержка выполнения кода для имитации продолжительности рейса.
- Случайное изменение пункта назначения в зависимости от погодных условий.

2. scheduleFlight
- Реализует задержку рейса и повторную проверку погоды.
- Отменяет рейс после 2-х проверок с плохой погодой.

3. isGoodWeather():
- Случайным образом определяет, является ли погода хорошей.

4. getWeatherCondition():
- Возвращает случайное число (0, 1, 2), соответствующее погодному условию.

Вывод

1. Вывод информации о формировании лётной бригады:
   - Выводится сообщение о формировании лётной бригады.
   - Последовательно выводится информация о пилотах, навигаторе, радиооператоре и стюардессах.
   - Для пилотов и стюардесс выводится их имя с соответствующим номером в списке.
   - Для навигатора и радиооператора выводится только их имя.

2. Вывод информации о рейсе и погодных условиях:
   - Выводится информация о рейсе (номер рейса, аэропорт отправления и назначения, модель и характеристики самолёта).
   - В зависимости от погодных условий, выводится соответствующее сообщение:
     А. При солнечной погоде в обоих аэропортах рейс разрешён.
     Б. При наличии сильных осадков и штормового предупреждения в аэропорту отправления или назначения, рейс задерживается.
     В. В случае непредвиденного состояния погоды также происходит задержка рейса.

3. Вызов методов для управления рейсом:
   - При солнечной погоде вызывается метод getJob для назначения обязанностей членам экипажа и начала рейса.
   - При наличии плохих погодных условий вызывается метод scheduleFlight для повторной проверки погоды и управления рейсом в зависимости от неё.

Код программы

Произвёл преобразования классов в абстрактные классы и интерфейсы:

```
// Интерфейс для аэропорта
interface Airport {
```

```
    String getName();
}
```

```
// Абстрактный класс для самолета
abstract class Aircraft {
    private int capacity;
    private int range;
    private String model;

    public Aircraft(int capacity, int range, String model) {
        this.capacity = capacity;
        this.range = range;
        this.model = model;
    }

    public int getCapacity() {
        return capacity;
    }

    public int getRange() {
        return range;
    }

    public String getModel() {
        return model;
    }
}
```

```
// Абстрактный класс для члена экипажа
abstract class CrewMember {
    private String name;
    private int age;

    public CrewMember(String name, int age) {
        this.name = name;
        this.age = age;
    }

    public String getName() {
        return name;
    }

    public int getAge() {
        return age;
    }
}
```

```
// Интерфейс для пилота
interface Pilot {
    void flyAircraft();
    void getVoice();
    String getName();
}
// Абстрактный класс для штурмана
abstract class Navigator extends CrewMember {
    public Navigator(String name, int age) {
        super(name, age);
    }

    public abstract void navigate();
}
```

```
// Абстрактный класс для радиста
abstract class RadioOperator extends CrewMember {
    public RadioOperator(String name, int age) {
```

```
        super(name, age);
    }

    public abstract void speakWithEarth();
}
```

```
// Абстрактный класс для стюардессы
abstract class Stewardess extends CrewMember {
    public Stewardess(String name, int age) {
        super(name, age);
    }

    public abstract void servePassengers();
    public abstract void conductBriefing();
}
```

Затем изменил классы для реализации преобразованных в абстрактные классы и интерфейсы:

```
// Класс Пилот, реализующий интерфейс Pilot
class PilotImpl extends CrewMember implements Pilot {
    public PilotImpl(String name, int age) {
        super(name, age);
    }

    @Override
    public void flyAircraft() {
        System.out.println("Главный пилот " + getName() + " управляет
самолётом");
    }

    @Override
    public void getVoice() {
        System.out.println("Помощник пилота " + getName() + " объявляет
необходимую информацию пассажирам по громкоговорителю");
    }
}
```

```
// Класс Штурман, реализующий абстрактный класс Navigator
class NavigatorImpl extends Navigator {
    public NavigatorImpl(String name, int age) {
        super(name, age);
    }

    @Override
    public void navigate() {
        System.out.println("Штурман определяет маршрут полёта...");
    }
}
```

И т.д.
Обновил метод main который теперь использует абстрактные классы и интерфейсы и создает их объекты:

```
public static void main(String[] args) {
    Main main = new Main();
    List<Stewardess> stewardessList = new ArrayList<>();
    List<Pilot> pilotList = new ArrayList<>();

    // Создаем объекты классов
    Aircraft aircraft = new AircraftImpl(200, 1500, "Boeing 737 №555");
    Airport departureAirport = new AirportImpl("Аэропорт Брестский ()");
```

```
    Airport destinationAirport = new AirportImpl("Аэропорт Иваново (IWA)");
    // Создаем рейс
    Flight flight = new Flight(departureAirport, destinationAirport,
aircraft);
    // Создаем и добавляем членов экипажа
    Pilot first_pilot = new PilotImpl("Сорока Вадим Сергеевич", 30);
    pilotList.add(first_pilot);
    Pilot second_pilot = new PilotImpl("Капитонов Максим Игоревич", 29);
    pilotList.add(second_pilot);

    Navigator navigator = new NavigatorImpl("Таразевич Никита Александрович",
29);
    RadioOperator radioOperator = new RadioOperatorImpl("Дорошков Александр
Дмитриевич", 29);

    Stewardess first_stewardess = new StewardessImpl("Сидорова Елена
Николаевна", 24);
    stewardessList.add(first_stewardess);
    Stewardess second_stewardess = new StewardessImpl("Петрова Александра
Сергеевна", 23);
    stewardessList.add(second_stewardess);

    // Вывод сообщения о формировании лётной бригады
    System.out.println("Администратор сформировал лётную бригаду:");
    // Вывод списка пилотов с нумерацией
    System.out.println("Пилоты:");
    for (int i = 0; i < pilotList.size(); i++) {
        System.out.println((i + 1) + ") " + pilotList.get(i).getName());
    }
    // Вывод навигатора и радиооператора
    System.out.println("Навигатор: " + navigator.getName());
    System.out.println("Радиооператор: " + radioOperator.getName());
    // Вывод списка стюардесс с нумерацией
    System.out.println("Стюардессы:");
    for (int i = 0; i < stewardessList.size(); i++) {
        System.out.println((i + 1) + ") " + stewardessList.get(i).getName());
    }
    System.out.println("\nРейс №228: " + departureAirport.getName() + " -- "
+ destinationAirport.getName() + "\nСамолёт: " + aircraft.getModel() +
"\nВместимость: " + aircraft.getCapacity() + "\nДальность рейса: " +
aircraft.getRange());

System.out.println("\n=========================================================
=====\n");

    int weatherCondition = getWeatherCondition();
    System.out.println(weatherCondition);
    switch (weatherCondition) {
        case 0:
            System.out.println("Солнечная погода в обоих аэропортах. Рейс
разрешен.\n");
            try {
                getJob(pilotList, navigator, radioOperator, stewardessList,
destinationAirport);
            } catch (InterruptedException e) {
                throw new RuntimeException(e);
            }
            break;
        case 1:
            System.out.println("Сильные осадки, штормовое предупреждение в
аэропорту отправления. Рейс задерживается.");
            scheduleFlight(stewardessList, pilotList, navigator,
radioOperator, destinationAirport);
            break;
        case 2:
            System.out.println("Сильные осадки, штормовое предупреждение в
аэропорту назначения. Рейс задерживается.");
```

```
            scheduleFlight(stewardessList, pilotList, navigator,
radioOperator, destinationAirport);
            break;
        default:
            System.out.println("Непредвиденное состояние погоды.");
            scheduleFlight(stewardessList, pilotList, navigator,
radioOperator, destinationAirport);
            break;
    }
}
```

Пример работы программы

А) хорошая погода без происшествий

```
Солнечная погода в обоих аэропортах. Рейс разрешен.

Главный пилот Сорока Вадим Сергеевич управляет самолётом
Помощник пилота Капитонов Максим Игоревич объявляет необходимую информацию пассажирам по громкоговорителю
Штурман определяет маршрут полёта...
Радист Дорошков Александр Дмитриевич докладывает обстановку по рации в диспетчерский центр
Стюардесса Сидорова Елена Николаевна обслуживает пассажиров в самолете
Стюардесса Петрова Александра Сергеевна проводит инструктаж пассажиров

Рейс проходит в штатном режиме...

Самолёт прибыл в Аэропорт Иваново (IWA)
Рейс №228 успешно завершён!
```

Б) хорошая погода в обоих аэропортах + техническая неисправность в полёте

```
================================================================================
Администратор сформировал лётную бригаду:
Пилоты:
1) Сорока Вадим Сергеевич
2) Капитонов Максим Игоревич
Навигатор: Таразевич Никита Александрович
Радиооператор: Дорошков Александр Дмитриевич
Стюардессы:
1) Сидорова Елена Николаевна
2) Петрова Александра Сергеевна

Рейс №228: Аэропорт Брестский () -- Аэропорт Иваново (IWA)
Самолёт: Boeing 737 №555
Вместимость: 200
Дальность рейса: 1500

================================================================

Солнечная погода в обоих аэропортах. Рейс разрешен.

Главный пилот Сорока Вадим Сергеевич управляет самолётом
Помощник пилота Капитонов Максим Игоревич объявляет необходимую информацию пассажирам по громкоговорителю
Штурман определяет маршрут полёта...
Радист Дорошков Александр Дмитриевич докладывает обстановку по рации в диспетчерский центр
Стюардесса Сидорова Елена Николаевна обслуживает пассажиров в самолете
Стюардесса Петрова Александра Сергеевна проводит инструктаж пассажиров

Рейс проходит в штатном режиме...
Говорит главный пилот самолёта Сорока Вадим Сергеевич! В связи с технической неисправностью воздушного судна мы не сможем долететь до аэропорта назначения: Аэропорт Иваново (IWA).
Поэтому мы изменяем курс на ближайший к нам аэропорт: Аэропорт Кобылёво. Авиакомпания заранее приносит свои извенения, все подробности будут позже!

Рейс проходит в аварийном режиме...

Самолёт прибыл в Аэропорт Кобылёво
Рейс №228 завершён преждевременно из-за технической неисправности!
================================================================================
```

В) плохая погода в аэропорте отправления + плохая погода в ходе рейса

```
Сильные осадки, штормовое предупреждение в аэропорту отправления. Рейс задерживается.
Ожидаем изменения погоды...
Погода улучшилась. Начинаем подготовку к рейсу...

Главный пилот Сорока Вадим Сергеевич управляет самолётом
Помощник пилота Капитонов Максим Игоревич объявляет необходимую информацию пассажирам по громкоговорителю
Штурман определяет маршрут полёта...
Радист Дорошков Александр Дмитриевич докладывает обстановку по рации в диспетчерский центр
Стюардесса Сидорова Елена Николаевна обслуживает пассажиров в самолете
Стюардесса Петрова Александра Сергеевна проводит инструктаж пассажиров

Рейс проходит в штатном режиме...

Внимание!!! В процессе рейса произошло резкое ухудшение погодных условий. Конечный аэропорт Аэропорт Иваново (IWA) временно недоступен для посадки!
Конечный пункт прибытия изменён с Аэропорт Иваново (IWA) на Аэропорт Иваново-Северный

Рейс проходит в штатном режиме...

Самолёт прибыл в Аэропорт Иваново-Северный
Рейс №228 был успешно завершён в другой конечной точке!
```

Г) хорошая погода + изменение погоды в ходе рейса

```
Солнечная погода в обоих аэропортах. Рейс разрешен.

Главный пилот Сорока Вадим Сергеевич управляет самолётом
Помощник пилота Капитонов Максим Игоревич объявляет необходимую информацию пассажирам по громкоговорителю
Штурман определяет маршрут полёта...
Радист Дорошков Александр Дмитриевич докладывает обстановку по рации в диспетчерский центр
Стюардесса Сидорова Елена Николаевна обслуживает пассажиров в самолете
Стюардесса Петрова Александра Сергеевна проводит инструктаж пассажиров

Рейс проходит в штатном режиме...

Внимание!!! В процессе рейса произошло резкое ухудшение погодных условий. Конечный аэропорт Аэропорт Иваново (IWA) временно недоступен для посадки!
Конечный пункт прибытия изменён с Аэропорт Иваново (IWA) на Аэропорт Иваново-Северный

Рейс проходит в штатном режиме...

Самолёт прибыл в Аэропорт Иваново-Северный
Рейс №228 был успешно завершён в другой конечной точке!
```

Д) отмена рейса из-за длительной плохой погоды

```
Сильные осадки, штормовое предупреждение в аэропорту назначения. Рейс задерживается.
Ожидаем изменения погоды...
Продолжается плохая погода. Повторная проверка через 5 секунд...
Ожидаем изменения погоды...
Продолжается плохая погода. Рейс отменён.
```

Вариантов различных исходов больше.

Вывод: научился создавать и использовать классы языка программирования Java при решении практических задач.