Plan

2023-09-27

1. Analysis of homework
2. Polymorphism
3. Practice

---

1. Разбор домашнего задания
2. Полиморфизм
3. Практика

Theory

▶ **English**

▼ **На русском**

Полиморфизм в ООП

Полиморфизм - это один из ключевых принципов объектно-ориентированного программирования (ООП), который позволяет использовать одно имя для разных действий.

- Полиморфизм может проявляться в двух формах: **статическом** и **динамическом**.

Статический Полиморфизм

Статический полиморфизм связан с перегрузкой методов и операторов. Это означает, что один метод может иметь несколько определений с разными параметрами, и компилятор будет выбирать подходящий метод на основе переданных аргументов.

Динамический полиморфизм (рассмотрим после наследования)

Перегрузка методов

Перегрузка методов — это приём программирования, который позволяет разработчику в одном классе для методов с разными параметрами использовать одно и то же имя. В этом случае мы говорим, что метод перегружен.

Зачем мне использовать перегрузку методов?

Использование перегрузки делает ваш код чище и проще для чтения, а также помогает избежать ошибок в программе.

Перегрузка конструкторов

В Java можно создавать несколько конструкторов с разными параметрами для одного класса. Это называется перегрузкой конструкторов (constructor overloading). Перегруженные конструкторы позволяют инициализировать объекты разными способами, в зависимости от переданных параметров.

Что такое конструктор

Конструктор — это специальный метод в классе, который вызывается при создании нового объекта этого класса. Он имеет тот же имя, что и класс, и не возвращает никакого значения (даже void). Задача конструктора — инициализировать поля объекта значениями, которые передаются в качестве параметров.

Homework

▶ **English**

▼ **На русском**

Задача 1

Создать класс **Cube**, описывающий куб со стороной a. Реализовать в нем методы определения площади s (сумма площадей всех граней) и объема v. Создать приложение **CubeAppl**, в котором создать несколько экземпляров класса Cube. Для каждого экземпляра вызвать методы класса Cube и рассчитать s и v.

Задача 2**

Задача: Необходимо разработать приложение для библиотеки, которое позволяет управлять списком книг. Каждая книга имеет следующие характеристики: название, автор, год издания, ISBN и флаг, указывающий, взята ли книга кем-то.

Основные требования:

Создать класс **Book**, представляющий книгу, с полями для хранения **названия**, **автора**, **года издания**, **ISBN** и флага (`boolean isBorrowed`), указывающего, взята ли книга кем-то.

Создать класс **Library**, представляющий библиотеку, с методами для управления списком книг. Методы должны включать **добавление книги**, **удаление книги**, **получение списка всех книг**, **проверку, взята ли книга**, и **установку флага, что книга взята или возвращена**.

- Реализовать основной класс приложения **LibraryApp**, который создает объект библиотеки, добавляет книги в библиотеку, проверяет статус книги (взята или нет), и устанавливает флаги, когда книги берутся или возвращаются.

Приложение должно предоставлять пользователю возможность управлять списком книг, добавлять новые книги, узнавать статус книг и устанавливать флаги, когда книги берутся или возвращаются.

▼ Инструкиця

Шаг 1: Создание класса Book

Создайте класс Book с приватными полями для названия (title), автора (author), года издания (year), ISBN (isbn) и флага, указывающего, взята ли книга (isBorrowed). Определите конструктор класса Book, который принимает параметры для инициализации полей. Реализуйте геттеры и сеттеры для полей класса Book, включая геттер и сеттер для флага isBorrowed.

Шаг 2: Создание класса Library

- Создайте класс Library с приватным полем books, которое будет представлять список книг в библиотеке. Инициализируйте его в конструкторе.
- Определите метод addBook(Book book), который добавляет книгу в список библиотеки.
- Определите метод removeBook(Book book), который удаляет книгу из списка библиотеки.
- Определите метод getAllBooks(), который возвращает список всех книг в библиотеке.
- Определите метод isBookBorrowed(String title, String author), который проверяет, взята ли книга по ее названию и автору.
- Определите метод borrowBook(String title, String author), который устанавливает флаг, что книга взята.
- Определите метод returnBook(String title, String author), который устанавливает флаг, что книга возвращена.

Шаг 3: Создание основного класса LibraryApp

- Создайте класс LibraryApp для выполнения основной логики приложения.
- В методе main создайте объект Library для представления библиотеки.
- Создайте несколько объектов Book и используйте методы addBook, чтобы добавить их в библиотеку.
- Используйте методы isBookBorrowed, borrowBook и returnBook, чтобы проверить статус книг и установить флаги взятия и возврата.

Шаг 4: Тестирование приложения

- Запустите приложение и проверьте, что вы можете успешно добавлять, удалять и проверять статус книг в библиотеке.

Шаг 5: Расширение функциональности (по желанию)

- Расширьте функциональность приложения, добавив дополнительные методы и опции, такие как поиск книг по различным критериям, вывод списка доступных книг и т. д.

Code

code/HwSolution_16/src/task1/User.java

```
package task1;

/**
 * @author Andrej Reutow
 * created on 26.09.2023
 */
public class User {
    // поля класса
    private String name;
    private long age;
    private String email;


    // конструктор
    public User(String name, int age, String email) {
        this.name = name;
        this.age = age;
        this.email = email;
    }

    // геттер для поля имя
    public String getName() {
        return this.name;
    }

    // сеттер для поля имя
    public void setName(String name) {
        this.name = name;
    }

    // геттер для поля возраст
    public long getAge() {
        return this.age;
```

```
    }

    // сеттер для поля возраст
    public void setAge(long age) {
        this.age = age;
    }

    // геттер для поля email
    public String getEmail() {
        return this.email;
    }

    // сеттер для поля email
    public void setEmail(String email){
        this.email = email;

    }



}
```

code/HwSolution_16/src/task1/Main.java

```
package task1;

/**
 * @author Andrej Reutow
 * created on 26.09.2023
 */
public class Main {
    public static void main(String[] args) {
        User user = new User("Andre", 35, "ar@gmail.com");

        // значение такие же как передавали в конструктор
        System.out.println(user.getName());     // Andre
        System.out.println(user.getAge());      // 35
        System.out.println(user.getEmail());    // ar@gmail.com

        System.out.println();
        // изменение значений
        user.setName("Vasja");  // Vasja
        user.setAge(18);        // 18
```

```
        user.setEmail("vasja@gmail.com"); // vasja@gmail.com

        // значения изменились
        System.out.println(user.getName());     // Vasja
        System.out.println(user.getAge());      // 18
        System.out.println(user.getEmail());    // vasja@gmail.com
    }
}
```

code/HwSolution_16/src/task3/BankAccount.java

```
package task3;

/**
 * @author Andrej Reutow
 * created on 27.09.2023
 */
public class BankAccount {
    private String accountNumber; // 100
    private double balance;

    public BankAccount(String accountNumber, double balance) {
        this.accountNumber = accountNumber;
        this.balance = balance;
    }

    public String getAccountNumber() {
        return accountNumber;
    }

    public void setAccountNumber(String accountNumber) {
        this.accountNumber = accountNumber;
    }

    public double getBalance() {
        return balance;
    }

    public void setBalance(double balance) {
        this.balance = balance;
    }

    // Создайте метод deposit, который принимает сумму для внесения и увели
    // Создайте метод withdraw, который принимает сумму для снятия и уменьш
```

```
        // Если сумма для снятия больше баланса, выведите сообщение об ошибке.

        public void deposit(double summ) {
            this.balance += summ;
        }

        public void withdraw(double summ) {
            if (this.balance >= summ) {
                this.balance -= summ;
                System.out.println("Сумма " + summ + " получена");
            } else {
                System.out.println("Не достаточно денег на вашем счету. Текущи
            }
        }
    }
```

code/HwSolution_16/src/task3/Main.java

```
    package task3;

    /**
     * @author Andrej Reutow
     * created on 26.09.2023
     * <p>
     * Цель: Создать класс "Банковский счет" с использованием инкапсуляции и ре
     * <p>
     * Создайте класс с именем "БанковскийСчет" (BankAccount) с приватными поля
     * <p>
     * accountNumber (номер счета, строковое значение)
     * balance (баланс счета, десятичное число с двумя знаками после запятой)
     * Добавьте конструктор класса, который принимает параметры для номера счет
     * <p>
     * Создайте геттеры и сеттеры для номера счета и баланса.
     * <p>
     * Создайте метод deposit, который принимает сумму для внесения и увеличива
     * Создайте метод withdraw, который принимает сумму для снятия и уменьшает
     * В методе main создайте объекты класса "BankAccount", инициализируйте их
     * <p>
     * Попробуйте разные операции с внесением и снятием средств, включая случаи
     */
    public class Main {
```

```
        public static void main(String[] args) {
            BankAccount bankAccount = new BankAccount("100", 150.58);
            System.out.println(bankAccount.getAccountNumber()); // 100
            bankAccount.setAccountNumber("5");
            System.out.println(bankAccount.getAccountNumber()); // 5

            System.out.println("Текущий баланс: " + bankAccount.getBalance());

            bankAccount.deposit(100.02);
            System.out.println("Текущий баланс после внесения суммы 100.2: " +

            bankAccount.withdraw(250);
            System.out.println("Текущий баланс после снятия суммы 250.60: " + b

            bankAccount.withdraw(2000);
            System.out.println("Текущий баланс после снятия суммы 2000: " + ban
        }
    }
```

code/Lesson_17/src/oop_part2/Calculator.java

```
package oop_part2;

/**
 * @author Andrej Reutow
 * created on 27.09.2023
 */
public class Calculator {

    public int addNumbers(int a, int b) {
        System.out.println("int + int");
        return a + b;
    }
    // overload - перегрузка методов
    public int addNumbers(long a, int b) {
        System.out.println("long + int");
        return (int) (a + b);
    }

    public int addNumbers(int a, long b) {
        System.out.println("int + long");
        return (int) (a + b);
```

```
    }

    public int addNumbers(long a, long b) {
        System.out.println("long + long");
        return (int) (a + b);
    }

    public int addNumbers(double a, double b) {
        System.out.println("double + double");
        return (int) (a + b);
    }

    public int addNumbers(short a, short b) {
        return (int) (a + b);
    }

    public int addNumbers(byte a, byte b) {
        return (int) (a + b);
    }

    public int divide(int a, int b) {
        return a / b;
    }
}
```

code/Lesson_17/src/oop_part2/CalculatorRunner.java

```
package oop_part2;

/**
 * @author Andrej Reutow
 * created on 27.09.2023
 */
public class CalculatorRunner {
    public static void main(String[] args) {
        Calculator paramPamPamCalculator = new Calculator(); // создание но

        int aInt = 10;
        int bInt = 5;
        int result = paramPamPamCalculator.addNumbers(aInt, bInt); // вызов
        System.out.println(result); // 15

        long aLong = 1000L;
        long bLong = 1000L;
```

```
        result = paramPamPamCalculator.addNumbers(aLong, bLong);
        System.out.println(result);

        paramPamPamCalculator.addNumbers(aInt, bLong); // int , long

        double aDouble = 10.0;
        double bDouble = 12.0;

        paramPamPamCalculator.addNumbers(aDouble, bDouble);
        paramPamPamCalculator.addNumbers(aInt, bInt);
    }
}
```

code/AitStudents/src/entity/Student.java

```
package entity;

/**
 * @author Andrej Reutow
 * created on 27.09.2023
 */
public class Student {

    private int id; // значение по умолчанию 0
    private String name; // значение по умолчанию null
    private String lastName; // значение по умолчанию null
    private boolean isPresent; // значение по умолчанию false


    public Student(int id, String name, String lastName) {
        this.id = id;
        this.name = name;
        this.lastName = lastName;
    }

    public Student(int id) {
        this.id = id;
    }

    public Student(String name, String lastName) {
        this.name = name;
        this.lastName = lastName;
```

```
    }

    public Student(int id, String lastName) {
        this.id = id;
        this.lastName = lastName;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getLastName() {
        return lastName;
    }

    public void setLastName(String lastName) {
        this.lastName = lastName;
    }

    public boolean isPresent() {
        return isPresent;
    }

    public void setPresent(boolean present) {
        isPresent = present;
    }
}
```

code/AitStudents/src/Main.java

```
import entity.Student;
```

```java
/**
 * @author Andrej Reutow
 * created on 27.09.2023
 * <p>
 * Написать приложение учета присутвия студентов на лекции.
 * <br>
 * - создать класс описывающий студента и его присутсвие (id, name, lastNam
 * <br>
 * - создать несколько экзепмляров (студентов) используя разные конструкто
 * <br>
 * - сохранить всех студентов в массиве
 * <br>
 * - вывести информацию (id, name, lastName, isPresent) для каждого студент
 * </p>
 */
public class Main {
    public static void main(String[] args) {
        // объявил переменную студент с типом Student
        // Student student;

        // Bobkova Maryna
        Student studentMaryna = new Student(14, "Bobkova");
        Student studentNastja = new Student(2, "Anastasia", "Chalova");
        Student studentFarkhunda = new Student(9, "Farkhunda", "Odinaeva");
        Student studentVitalij = new Student("Vitalij", "Korniienko");
        Student studentMartins = new Student(8);
        Student studentDenys = new Student(6, "Denys", "Liubchenko");


        studentMaryna.setName("Maryna");
        System.out.println(studentMaryna.getId()); // 14
        System.out.println(studentMaryna.getName()); // Maryna
        System.out.println(studentMaryna.getLastName()); // Bobkova

        studentVitalij.setId(5);

        studentMartins.setName("Martins");
        studentMartins.setLastName("Groza");

        studentMaryna.setPresent(true);
        studentNastja.setPresent(true);
        studentFarkhunda.setPresent(true);
        studentVitalij.setPresent(true);
```

```
        studentMartins.setPresent(true);
        studentDenys.setPresent(false);

        System.out.println();

        Student[] students = new Student[6];
        students[0] = studentMaryna;
        students[1] = studentNastja;
        students[2] = studentFarkhunda;
        students[3] = studentVitalij;
        students[4] = studentMartins;
        students[5] = studentDenys;

        for (int i = 0; i < students.length; i++) {
            Student currentStudent = students[i];
            System.out.println("id: " + currentStudent.getId()
                    + " Name: " + currentStudent.getName()
                    + " LastName: " + currentStudent.getLastName()
                    + " is present: " + currentStudent.isPresent()
            );
        }
    }
}
```