Plan

2023-09-28

1. Analysis of homework
2. Inheritance in Java
    1. Introduction to inheritance
    2. Examples
    3. The extends keyword
    4. Keyword super
    5. The principle of inheritance
    6. Object

---

1. Разбор домашнего задания (максимум 30 минут !!!)
2. Наследование (Inheritance) в Java
    1. Введение в наследование
    2. Примеры
    3. Ключевое слово extends
    4. Ключевое слово super
    5. Принцип наследования
    6. Object

Theory

▶ **English**

▼ **На русском**

Наследование (Inheritance) в Java

Что такое наследование?

Наследование - это механизм объектно-ориентированного программирования, который позволяет создавать новый класс на основе существующего класса. В Java наследование позволяет одному классу (подклассу или дочернему классу) наследовать свойства и методы другого класса (суперкласса или родительского класса).

Подкласс может использовать все открытые (public) и защищенные (protected) члены суперкласса.

Преимущества наследования:

- Повторное использование кода: Вы можете использовать уже существующий код из родительского класса в подклассе, что способствует экономии времени и уменьшению дублирования кода.
- Иерархия классов: Наследование позволяет создавать иерархию классов, что упрощает организацию кода и его понимание.
- Расширение функциональности: Вы можете добавлять новые поля и методы в подкласс, расширяя функциональность базового класса.

Ключевое слово extends

В Java для создания наследования используется ключевое слово extends. Синтаксис следующий:

```
class ChildClass extends ParentClass {
    // дополнительные поля и методы
}
```

В этом примере ChildClass является подклассом, который наследует свойства и методы из ParentClass.

Примеры

Пример 1: Фигуры

Рассмотрим иерархию классов для геометрических фигур. У нас есть суперкласс "Фигура", который содержит общие свойства, такие как площадь и периметр. Из этого суперкласса мы создаем подклассы: "Круг", "Прямоугольник" и "Треугольник". Каждый подкласс имеет свои уникальные свойства и методы, но наследует общие характеристики от суперкласса.

Пример 2: Автомобили

Предположим, у нас есть класс "Автомобиль", который описывает общие характеристики автомобилей. Мы создаем подклассы для разных типов автомобилей, таких как "Легковой автомобиль" и "Грузовик". Каждый подкласс может иметь свои уникальные свойства и методы, но также унаследует общие черты от суперкласса.

Ключевое слово super

super - это ключевое слово, которое используется для обращения к членам суперкласса (родительского класса) из подкласса. Оно позволяет подклассу получать доступ к методам, полям или конструкторам, унаследованным от суперкласса.

Использование super для вызова конструктора суперкласса

Одним из наиболее распространенных способов использования super является вызов конструктора суперкласса из конструктора подкласса. Это полезно, когда подкласс хочет

выполнить инициализацию, определенную в суперклассе, перед добавлением своей собственной инициализации.

```
class SuperClass {
    int value;

    SuperClass(int value) {
        this.value = value;
    }
}

class SubClass extends SuperClass {
    int counter;

    Подкласс(int value, int counter) {
        super(value); // Вызываем конструктор суперкласса
        this.counter = counter;
    }
}
```

В этом примере конструктор SubClass вызывает конструктор суперкласса с использованием super(значение). Это позволяет установить значение поля значение в суперклассе.

Принцип наследования

- Подкласс может наследовать все non-private поля и методы из суперкласса.
- Подкласс может переопределить (override) методы суперкласса, чтобы изменить их поведение.
- Подкласс может добавить свои собственные поля и методы.

Прородитель всех классов в Java - Object

В Java все классы неявно наследуются от класса Object. Это означает, что каждый класс, который вы создаете в Java, автоматически становится подклассом Object, даже если вы явно не указываете extends Object. Этот механизм наследования обеспечивает общий набор методов и функциональность для всех объектов в Java.

Вот несколько ключевых аспектов связанных с наследованием от класса Object:

Класс Object определяет несколько методов, которые унаследованы всеми классами в Java. Некоторые из наиболее часто используемых методов Object включают:

- toString(): Этот метод возвращает строковое представление объекта. По умолчанию, если вы не переопределяете его в своем собственном классе, он возвращает строку,

содержащую имя класса и хэш-код объекта.
- equals(Object obj): Этот метод используется для сравнения объектов на равенство. По умолчанию, он сравнивает объекты на основе ссылок, но вы можете переопределить его в своем классе, чтобы определить собственное правило сравнения.
- hashCode(): Этот метод возвращает хэш-код объекта. Хэш-код используется в хэш-таблицах и для оптимизации процесса поиска объектов.
- getClass(): Этот метод возвращает объект типа Class, который представляет класс объекта.
- clone(): Этот метод используется для создания копии объекта. Он требует, чтобы класс реализовал интерфейс Cloneable и переопределил метод clone().

Основные понятия:

1. Суперкласс (родительский класс) Суперкласс - это класс, от которого происходит наследование. Он содержит общие свойства и методы, которые могут быть использованы в подклассах. Например, давайте рассмотрим класс "Животное":

```
public class Animal {
    String name;

    public void eat() {
        System.out.println(name + " ест.");
    }
}
```

2. Подкласс (дочерний класс) Подкласс - это класс, который наследует свойства и методы от суперкласса. Он может также добавлять новые свойства и методы или изменять унаследованные. Например, создадим подкласс "Собака":

```
public class Dog extends Animal {
    public void bark() {
        System.out.println(name + " лает.");
    }
}
```

`

3. Иерархия классов Иерархия классов - это иерархическая структура классов, где один класс может быть суперклассом для нескольких подклассов, создавая древовидную структуру наследования.
4. Ключевое слово **extends**: В языке Java наследование реализуется с использованием ключевого слова **extends**
5. **Единичное наследование**: В Java класс может наследовать только один суперкласс. Это означает, что множественное наследование классов (когда класс наследует сразу

несколько классов) **не поддерживается**.
6. Вызов конструктора суперкласса: Конструктор подкласса может вызывать конструктор суперкласса с помощью ключевого слова **super()**. Это позволяет инициализировать члены суперкласса перед инициализацией членов подкласса.
7. Конструкторы в подклассе: **Подкласс должен иметь хотя бы один конструктор**. Если в суперклассе нет конструктора без аргументов, то в конструкторе подкласса нужно явно вызвать конструктор суперкласса с помощью **super()**.

Homework

▶ **English**

▼ **На русском**

Задача 1: Управление студентами

Создайте базовый класс `Student` (Студент) с атрибутами `имя`, `возраст` и `средний балл`. Затем создайте два подкласса: `HighSchoolStudent` (Старшеклассник) и `CollegeStudent` (Студент колледжа).

Добавьте методы в базовый класс для вычисления статуса студента на основе среднего балла. Например, если средний балл больше или равен 4.5, студент считается "отличником", если между 3.0 и 4.49 - "хорошистом", и так далее.

Создайте несколько объектов разных классов студентов и вызовите методы для определения их статуса.

Задача 2: Иерархия компьютерных устройств

Создайте иерархию классов для компьютерных устройств: ноутбуки и стационарные компьютеры. Каждое устройство имеет атрибуты `марка` и `объем памяти`.

Создайте методы для вывода информации о каждом устройстве, включая его тип (ноутбук или стационарный компьютер), марку и объем памяти.

Создайте объекты разных типов компьютерных устройств и выведите информацию о каждом из них.

Задача 3*: Иерархия животных

Создайте иерархию классов для животных, включая млекопитающих и птиц. У каждого класса будут общие атрибуты `имя` и `место обитания`, а также специфичные атрибуты: у млекопитающих - `тип питания` (травоядные, хищники) и `количество ног`, у птиц - `размах крыльев` и `тип перьев` (пушистые, пернатые).

Создайте методы для каждого класса, позволяющие животным издавать свои звуки (мяукать, гавкать, чирикать и т. д.). Также реализуйте метод, который выводит информацию о животных, включая их общие и специфичные атрибуты.

Создайте объекты разных типов животных и вызовите методы для издания звуков и вывода информации о них.

Задачи на приведение типов, для закрепления:

Задача 4: Приведение типов в иерархии животных

Предположим, у нас есть иерархия классов животных:

```
class Animal { /* ... */ }
class Mammal extends Animal { /* ... */ }
class Bird extends Animal { /* ... */ }
```

Создайте объекты разных классов и попробуйте выполнить приведение типов между ними. Например, создайте объект типа Mammal, а затем попробуйте привести его к типу Animal.

Задача 5: Приведение типов в иерархии студентов

У нас есть иерархия классов студентов:

```
class Student { /* ... */ }
class HighSchoolStudent extends Student { /* ... */ }
class CollegeStudent extends Student { /* ... */ }
```

Создайте объекты разных классов студентов и попробуйте выполнить приведение типов между ними. Например, создайте объект типа CollegeStudent, а затем попробуйте привести его к типу Student

Задача 6: Приведение типов в иерархии компьютерных устройств

У нас есть иерархия классов компьютерных устройств:

```
class ComputerDevice { /* ... */ }
class Laptop extends ComputerDevice { /* ... */ }
class DesktopComputer extends ComputerDevice { /* ... */ }
```

Создайте объекты разных классов компьютерных устройств и попробуйте выполнить приведение типов между ними. Например, создайте объект типа Laptop, а затем попробуйте привести его к типу ComputerDevice.

Code

code/HwSolution_17/src/task2/Book.java

```java
package task2;

/**
 * @author Andrej Reutow
 * created on 28.09.2023
 */
public class Book {

    private String title;
    private String author;
    private int year;
    private boolean isBorrowed;


    public Book(String title, String author, int year) {
        this.title = title;
        this.author = author;
        this.year = year;
    }

    public String getTitle() {
        return title;
    }

    public void setTitle(String title) {
        this.title = title;
    }

    public String getAuthor() {
        return author;
    }

    public void setAuthor(String author) {
        this.author = author;
    }

    public int getYear() {
        return year;
    }

    public void setYear(int year) {
```

```
        this.year = year;
    }

    public boolean isBorrowed() {
        return isBorrowed;
    }

    public void setBorrowed(boolean borrowed) {
        isBorrowed = borrowed;
    }
}
```

code/HwSolution_17/src/task2/Library.java

```
package task2;

/**
 * @author Andrej Reutow
 * created on 28.09.2023
 */

/*
Создайте класс Library с приватным полем books, которое будет представлять
  его в конструкторе.
- Определите метод addBook(Book book), который добавляет книгу в список биб
- Определите метод removeBook(Book book), который удаляет книгу из списка б
- Определите метод getAllBooks(), который возвращает список всех книг в биб
- Определите метод isBookBorrowed(String title, String author), который про
  автору.
- Определите метод borrowBook(String title, String author), который устанав
- Определите метод returnBook(String title, String author), который устанав

 */
public class Library {

    private Book[] books;
//    private Book[] books = new Book[50];

    public Library(Book[] books) {
        this.books = books;
    }

    public Library(int booksSize) {
        this.books = new Book[booksSize];
```

```
    }

    public void addBook(Book bookToAdd) {
        boolean isAdded = false;
        for (int i = 0; i < books.length; i++) {
            if (books[i] == null) {
                books[i] = bookToAdd;
                isAdded = true;
            }
        }
        if (!isAdded) {
            System.out.println("Для книги " + bookToAdd.getAuthor() + " " +
        }
    }

    public void removeBook(Book bookToAdd) {
        String title = bookToAdd.getTitle();
        String author = bookToAdd.getAuthor();

        for (int i = 0; i < books.length; i++) {
            Book currentBook = books[i];
            //todo проверить является ли currentBook == null
            if (currentBook.getTitle().equals(title) && currentBook.getAuth
                books[i] = null;
            }
        }
    }

    public boolean isBookBorrowed(String title, String author) {
        boolean isBookBorrowed = false;

        for (int i = 0; i < books.length; i++) {
            Book currentBook = books[i];
            if (currentBook.getTitle().equals(title) && currentBook.getAuth
                isBookBorrowed = currentBook.isBorrowed();
                break;
            }
        }

        return isBookBorrowed;
    }

    public void borrowBook(String title, String author) {
```

```
            for (int i = 0; i < books.length; i++) {
                Book currentBook = books[i];
                if (currentBook.getTitle().equals(title) && currentBook.getAuth
                    currentBook.setBorrowed(true);
                    break;
                }
            }
        }

        public void returnBook(String title, String author) {
            for (int i = 0; i < books.length; i++) {
                Book currentBook = books[i];
                if (currentBook.getTitle().equals(title) && currentBook.getAuth
                    currentBook.setBorrowed(false);
                    break;
                }
            }
        }

        public void getAllBooks() {
            for (int i = 0; i < books.length; i++) {
                Book currentBook = books[i];
                //todo проверить является ли currentBook == null
                System.out.println("Author: " + currentBook.getAuthor() +
                        "Title: " + currentBook.getTitle() +
                        "Year: " + currentBook.getYear() +
    //                    "Borrowed: " + (currentBook.isBorrowed() ? " is avvai
                        "Borrowed: " + currentBook.isBorrowed()
                );
            }
        }
    }
```

code/HwSolution_17/src/task2/LibraryApp.java

```
    package task2;

    /**
     * @author Andrej Reutow
     * created on 28.09.2023
     */
    public class LibraryApp {

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

```
        Book book1 = new Book("Harry Poter", "J. Rowling", 1998);
        Book book2 = new Book("Властелин колец", "Джон Рональд Руэл Толкин"
        Book book3 = new Book("1984", "Джордж Оруэлл", 1949);
        Book book4 = new Book("50 оттенков серого", "Э. Л. Джеймс", 2011);
        Book book5 = new Book("Игра престолов", "Джордж Р. Р. Мартин", 1996

        Book[] books = new Book[3];
        books[0] = book1;
        books[1] = book2;
        books[2] = book3;


        Library library = new Library(books);
        // todo fix it NullPointerException
        // library.addBook(book4);
        library.borrowBook("Harry Poter", "J. Rowling");
        library.borrowBook("50 оттенков серого", "Э. Л. Джеймс");

        // todo fix NullPointerException
        library.removeBook(book1);
        System.out.println();
        library.getAllBooks();
    }
}
```

code/Lesson_18/src/object/IntObject.java

```
package object;

/**
 * @author Andrej Reutow
 * created on 28.09.2023
 */
public class IntObject {

    private int value;

    public IntObject(int value) {
        this.value = value;
    }

    public int getValue() {
        return value;
    }
```

```
    public void setValue(int value) {
        this.value = value;
    }
}
```

code/Lesson_18/src/object/Book.java

```
package object;

/**
 * @author Andrej Reutow
 * created on 28.09.2023
 */
public class Book {

    private String title;
    private String author;
    private int year;
    private boolean isBorrowed;


    public Book(String title, String author, int year) {
        this.title = title;
        this.author = author;
        this.year = year;
    }

    public String getTitle() {
        return title;
    }

    public void setTitle(String title) {
        this.title = title;
    }

    public String getAuthor() {
        return author;
    }

    public void setAuthor(String author) {
        this.author = author;
    }
```

```
        public int getYear() {
            return year;
        }

        public void setYear(int year) {
            this.year = year;
        }

        public boolean isBorrowed() {
            return isBorrowed;
        }

        public void setBorrowed(boolean borrowed) {
            isBorrowed = borrowed;
        }
    }
```

code/Lesson_18/src/object/Application.java

```
    package object;

    /**
     * @author Andrej Reutow
     * created on 28.09.2023
     */
    public class Application {

        public static void main(String[] args) {
            Book book1 = new Book("Harry Poter", "J. Rowling", 1998);
            Book book2 = new Book("Harry Poter", "J. Rowling", 1998);
            Book book3 = book2;
            Book book4 = book1;
            Book book5 = book2;

            book2.setYear(2022);

            System.out.println("Вывод 1 книги");
            System.out.println("Author: " + book1.getAuthor() +
                    " Title: " + book1.getTitle() +
                    " Year: " + book1.getYear() +
                    " Borrowed: " + book1.isBorrowed()
            );

            System.out.println();
```

```
        System.out.println("Вывод 2 книги");
        System.out.println("Author: " + book2.getAuthor() +
                " Title: " + book2.getTitle() +
                " Year: " + book2.getYear() +
                " Borrowed: " + book2.isBorrowed()
        );


        int a = 5;
        int b = 10;

        int c = a; // c =  5
        c = 100;
        System.out.println();

        IntObject intObjectA = new IntObject(5);
        IntObject intObjectC = intObjectA;
        intObjectC.setValue(100);

        System.out.println();
    }
}
```

code/Lesson_18/src/inheritance/auto/child/Bus.java

```
package inheritance.auto.child;

import inheritance.auto.model.Auto;

/**
 * @author Andrej Reutow
 * created on 28.09.2023
 */
public class Bus extends Auto {

    public void transportPassengers() {
        System.out.println("bus");
        printDetails();
    }
}
```

code/Lesson_18/src/inheritance/auto/child/Truck.java

```java
package inheritance.auto.child;

import inheritance.auto.model.Auto;

/**
 * @author Andrej Reutow
 * created on 28.09.2023
 */
public class Truck extends Auto {

    public void driveTrailer() {
        System.out.println("Trailer");
    }
}
```

code/Lesson_18/src/inheritance/auto/model/Auto.java

```java
package inheritance.auto.model;

/**
 * @author Andrej Reutow
 * created on 28.09.2023
 */
public class Auto {

    private String model;
    private int year;
    private int power;
    private String color;

    public void printDetails() {
        System.out.println("Модель " + this.model + ", Год " + this.year);
    }

    public String getModel() {
        return model;
    }

    public void setModel(String model) {
        this.model = model;
    }

    public int getYear() {
```

```
        return year;
    }

    public void setYear(int year) {
        this.year = year;
    }

    public String getColor() {
        return color;
    }

    public void setColor(String color) {
        this.color = color;
    }
}
```

code/Lesson_18/src/inheritance/auto/Main.java

```
package inheritance.auto;

import inheritance.auto.child.Bus;
import inheritance.auto.child.Truck;

/**
 * @author Andrej Reutow
 * created on 28.09.2023
 * <p>
 * Предположим, у нас есть класс "Автомобиль", который описывает общие хара
 * Мы создаем подклассы для разных типов автомобилей, таких как "Легковой а
 * Каждый подкласс может иметь свои уникальные свойства и методы, но также
 */
public class Main {

    public static void main(String[] args) {
        Truck truck = new Truck();
        Bus bus = new Bus();

        System.out.println();

        truck.setColor("Black");
        bus.setYear(2023);

        truck.driveTrailer();
        bus.transportPassengers();
```

```
        truck.printDetails();
        bus.printDetails();

        System.out.println();
    }
}
```

code/Lesson_18/src/inheritance/shape/Shape.java

```
package inheritance.shape;

/**
 * @author Andrej Reutow
 * created on 28.09.2023
 */

/*
Рассмотрим иерархию классов для геометрических фигур.
У нас есть суперкласс "Фигура", который содержит общие свойства, такие как
 Из этого суперкласса мы создаем подклассы: "Круг", "Прямоугольник" и "Треу
  Каждый подкласс имеет свои уникальные свойства и методы, но наследует общ
 */
public class Shape { // суперкласс

    protected int area;
    private int perimeter;
    private String name;

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

    public double calculatePerimeter() {
        System.out.println("Расчет периметра в классе Shape");
        return -1;
    }

    public int getArea() {
        return area;
    }

    public void setArea(int area) {
        this.area = area;
```

```
    }

    public int getPerimeter() {
        return perimeter;
    }

    public void setPerimeter(int perimeter) {
        this.perimeter = perimeter;
    }
}
```

code/Lesson_18/src/inheritance/shape/Triangle.java

```
package inheritance.shape;

/**
 * @author Andrej Reutow
 * created on 28.09.2023
 */
public class Triangle extends Shape {

    private int sideA;
    private int sideB;
    private int sideC;

    public Triangle(String name, int sideA, int sideB, int sideC) {
        super(name);
        this.sideA = sideA;
        this.sideB = sideB;
        this.sideC = sideC;
    }

    @Override
    public double calculatePerimeter() {
        System.out.println("Расчет периметра в классе Triangle");
        return sideA + sideB + sideC;
    }
}
```

code/Lesson_18/src/inheritance/shape/Circle.java

```
package inheritance.shape;

/**
```

```
 * @author Andrej Reutow
 * created on 28.09.2023
 */
public class Circle extends Shape {
    private int radius;

    public Circle(String name, int radius) {
        super(name);
        this.radius = radius;
    }

    @Override
    public double calculatePerimeter() {
        System.out.println("Расчет периметра в классе Circle");
        return (2 * 3.14) * radius;
    }

    public void getAreaCircle() {
        super.getArea();
    }

    public int getArea() {
        return -1;
    }
}
```

code/Lesson_18/src/inheritance/shape/Main.java

```
package inheritance.shape;

import java.util.Scanner;

/**
 * @author Andrej Reutow
 * created on 28.09.2023
 */
public class Main {

    public static void main(String[] args) {
//        Shape shape = new Shape("Фигура");
//        Circle circle = new Circle("Окружность", 10);
//        Triangle triangle = new Triangle("Треугольник", 5, 6, 7);
//
//        double perimeterShape = shape.calculatePerimeter();
```

```
//        double perimeterTriangle = triangle.calculatePerimeter();
//        double perimeterCircle = circle.calculatePerimeter();

        System.out.println();

        Shape circle = new Circle("Окружность", 10);
        Shape triangle = new Triangle("Треугольник", 5, 6, 7);

        circle.calculatePerimeter();
        triangle.calculatePerimeter();

        // downcasting объектов
        Circle circle1 = (Circle) circle; // низходящее преобразование перм
        circle1.getAreaCircle();

        // upcasting объектов
        Circle circle2 = new Circle("Окружность", 10);
        circle2.getAreaCircle();
        Shape circle3 = circle2;


        Circle circle4 = new Circle("Окружность", 10);
        Shape circle5 = circle4;
        if (circle5 instanceof Triangle) {
            Triangle triangle1 = (Triangle) circle5;
        }

        System.out.println(circle4 instanceof Circle);
        System.out.println(circle4 instanceof Shape);
        System.out.println(circle5 instanceof Shape);
        System.out.println(circle5 instanceof Triangle);

        System.out.println(circle5 instanceof Circle);
        System.out.println(circle5 instanceof Triangle);
        System.out.println(circle5 instanceof Shape);

        System.out.println(triangle instanceof Shape);
        System.out.println(triangle instanceof Triangle);
        System.out.println(triangle instanceof Circle);

        System.out.println(circle4 instanceof Object);
        System.out.println(circle1 instanceof Object);
        System.out.println(circle2 instanceof Object);
```

```
        System.out.println(circle3 instanceof Object);
        System.out.println(triangle instanceof Object);

    }
}
```