МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ
ОРДЕНА ТРУДОВОГО КРАСНОГО ЗНАМЕНИ ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«МОСКОВСКИЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ СВЯЗИ И ИНФОРМАТИКИ»
(МТУСИ)

Факультет
Информационные технологии
Искусственный интеллект и машинное обучение

по теме:

Разработка многоуровневой иерархии классов с реализацией принципов объектно-ориентированного программирования

Студент:
БВТ 2401 *А.И. Меланич*

Предподаватель:

Москва 2025

СОДЕРЖАНИЕ

ЦЕЛЬ ЛАБОРАТОРНОЙ РАБОТЫ

Закрепить теоретические знания и получить практические навыки проектирования и реализации объектно-ориентированной системы, включающей абстрактный класс и многоуровневое наследование.

Конкретные задачи, которые необходимо решить:

- освоить принципы создания и использования абстрактных классов;
- научиться реализовывать многоуровневую иерархию наследования;
- получить навыки разработки классов с инкапсуляцией данных через геттеры и сеттеры;
- изучить механизмы работы конструкторов, включая конструктор по умолчанию;
- освоить применение статических переменных для реализации счётчика объектов;
- научиться реализовывать и демонстрировать все основные принципы объектно-ориентированного программирования (наследование, инкапсуляция, полиморфизм, абстракция);
- получить опыт работы с вводом и выводом информации об объектах.

Ожидаемые результаты:

- создание функциональной объектно-ориентированной системы;
- демонстрация понимания принципов ООП на практике;
- формирование навыков проектирования иерархий классов;
- развитие умений реализации сложных программных конструкций.

1 ЗАДАНИЕ

Описание программного кода лабораторной работы

Представленный код демонстрирует реализацию **абстрактного класса** Bicycle, который служит основой для создания иерархии классов велосипедов. Рассмотрим основные компоненты и особенности реализации:

1. **Структура класса**

- Класс объявлен как абстрактный, что означает невозможность создания его экземпляров напрямую
- Используются **приватные поля** для хранения характеристик велосипеда:
  - model (модель)
  - yearOfManufacture (год выпуска)
  - hasBasket (наличие корзины)

2. **Система подсчета объектов**

- Реализован статический механизм подсчета созданных объектов через:
  - Статическую переменную instanceCount типа Map
  - Автоматическое увеличение счетчика при создании каждого объекта
  - Метод getInstanceCount для получения количества созданных объектов определенного класса

3. **Конструкторы класса**

- Предоставлены два варианта конструкторов:
  - Конструктор по умолчанию с инициализацией базовых значений
  - Параметризованный конструктор для установки конкретных характеристик

4. **Механизмы доступа**

– Реализованы **геттеры и сеттеры** для всех полей класса, обеспечивающие:

  – Контроль доступа к приватным полям
  – Поддержание инкапсуляции данных

5. **Абстрактный метод**

– Определен абстрактный метод getDetails(), который:

  – Должен быть реализован в классах-наследниках
  – Обеспечивает полиморфизм в иерархии классов

Данный код демонстрирует применение следующих принципов ООП:

– **Инкапсуляция** через использование приватных полей и публичных методов доступа
– **Наследование** через возможность создания производных классов
– **Абстракция** через определение базового функционала и абстрактного метода
– **Полиморфизм** через реализацию абстрактного метода в наследниках

Код служит основой для дальнейшей разработки иерархии классов велосипедов, демонстрируя корректную реализацию объектно-ориентированного подхода в программировании.

1.1 Bicycle.java

```java
import java.util.HashMap;
import java.util.Map;

abstract class Bicycle {
  private String model = "Unknown";
  private int yearOfManufacture = 0;
  private boolean hasBasket = false;
  private static final Map<Class<?>, Integer> instanceCount = new HashMap<>();

  public Bicycle() {
    Class<?> clazz = getClass();
    instanceCount.put(clazz, instanceCount.getOrDefault(clazz, 0) + 1);
```

```
  }

  public Bicycle(String model, int yearOfManufacture, boolean hasBasket) {
    this();
    this.model = model;
    this.yearOfManufacture = yearOfManufacture;
    this.hasBasket = hasBasket;
  }

  public static <T> int getInstanceCount(Class<T> clazz) {
    return instanceCount.getOrDefault(clazz, 0);
  }

  public String getModel() {
    return model;
  }

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

  public int getYearOfManufacture() {
    return yearOfManufacture;
  }

  public void setYearOfManufacture(int yearOfManufacture) {
    this.yearOfManufacture = yearOfManufacture;
  }

  public boolean isHasBasket() {
    return hasBasket;
  }

  public void setHasBasket(boolean hasBasket) {
    this.hasBasket = hasBasket;
  }

  public abstract String getDetails();

}
```

1.2 BMX.java

```
class BMX extends Bicycle {
  private boolean hasWheels;

  public BMX() {
    super();
    this.hasWheels = true;
  }

  public BMX(String model, int yearOfManufacture, boolean hasBasket,
      boolean hasWheels) {
    super(model, yearOfManufacture, hasBasket);
    this.hasWheels = hasWheels;
  }
```

```
  public boolean isHasWheels() {
    return hasWheels;
  }

  public void setHasWheels(boolean hasWheels) {
    this.hasWheels = hasWheels;
  }

  @Override
  public String getDetails() {
    return "BMX: Model=" + getModel() + ", Year of Manufacture="
        + getYearOfManufacture() + ", Has Basket=" + isHasBasket()
        + ", Has Wheels=" + isHasWheels();
  }
}
```

1.3 MountainBike.java

```
class MountainBike extends Bicycle {
  private boolean hasSuspension;

  public MountainBike() {
    super();
    this.hasSuspension = false;
  }

  public MountainBike(String model, int yearOfManufacture, boolean hasBasket,
      boolean hasSuspension) {
    super(model, yearOfManufacture, hasBasket);
    this.hasSuspension = hasSuspension;
  }

  public boolean isHasSuspension() {
    return hasSuspension;
  }

  public void setHasSuspension(boolean hasSuspension) {
    this.hasSuspension = hasSuspension;
  }

  @Override
  public String getDetails() {
    return "Mountain Bike: Model=" + getModel() + ", Year of Manufacture="
        + getYearOfManufacture() + ", Has Basket=" + isHasBasket()
        + ", Has Suspension=" + isHasSuspension();
  }
}
```

1.4 ChildrenBike.java

```
class ChildrenBike extends Bicycle {
```

```
  private boolean hasBell;

  public ChildrenBike() {
    super();
    this.hasBell = false;
  }

  public ChildrenBike(String model, int yearOfManufacture, boolean hasBasket,
      boolean hasBell) {
    super(model, yearOfManufacture, hasBasket);
    this.hasBell = hasBell;
  }

  public boolean isHasBell() {
    return hasBell;
  }

  public void setHasBell(boolean hasBell) {
    this.hasBell = hasBell;
  }

  @Override
  public String getDetails() {
    return "Children's Bike: Model=" + getModel() + ", Year of Manufacture="
        + getYearOfManufacture() + ", Has Basket=" + isHasBasket()
        + ", Has Bell=" + isHasBell();
  }
}
```

ВЫВОД

Вывод по лабораторной работе

В ходе выполнения лабораторной работы были успешно достигнуты поставленные цели и решены все поставленные задачи.

Основные результаты работы:

1. Практическое освоение принципов объектно-ориентированного программирования:

– Создана иерархия классов с использованием абстрактного базового класса
– Реализованы механизмы наследования и инкапсуляции
– Продемонстрирован принцип полиморфизма через абстрактные методы
– Использована абстракция при проектировании базового класса

2. Полученные практические навыки:

– Разработка абстрактных классов и их наследников
– Создание многоуровневой системы наследования
– Реализация конструкторов различных типов
– Разработка механизмов доступа через геттеры и сеттеры
– Работа со статическими переменными и методами

3. Достигнутые результаты:

– Разработана функциональная система классов с учетом всех требований задания
– Реализован механизм подсчета созданных объектов
– Обеспечен корректный ввод и вывод информации об объектах
– Продемонстрирована работа всех компонентов системы

Выводы:

В результате выполнения лабораторной работы был создан работоспособный программный продукт, демонстрирующий основные принципы

объектно-ориентированного программирования. Получены практические навыки проектирования и реализации иерархических структур классов, что является фундаментальным элементом современного программирования.