**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)

Информационные технологии и программирование

Отчет по лабораторной работе №1

|  |  |
| --- | --- |
| Выполнил: | студент группы БПИ2305 Архипов А.А. |

Москва

Цель работы

1. Освоить базовые принципы объектно-ориентированного программирования (абстракция, инкапсуляция, полиморфизм, наследование).
2. Научиться реализовывать иерархию классов на языке программирования Java.
3. Применить концепции наследования, перегрузки, переопределения и модификаторов доступа на практике.

Задание

**Задание 1**

Создать иерархию классов, которая включает в себя:

1. Абстрактный класс.
2. Два уровня наследуемых классов, которые содержат минимум три поля и два метода, описывающих поведение объектов.
3. Применение всех принципов ООП: абстракция, инкапсуляция, наследование, полиморфизм.
4. Конструкторы (включая конструктор по умолчанию), геттеры и сеттеры.
5. Реализацию счетчика созданных объектов с использованием статической переменной в одном из классов.
6. Демонстрацию ввода и вывода информации о создаваемых объектах.

Ход выполнения работы

**Часть 1: Создание иерархии классов**

**Код классов:**

**Абстрактный класс Application.java:**

**public** **abstract** **class** App {

**private** **String** name;

**private** **String** developer;

**private** **double** version;

**public** App(**String** *name*, **String** *developer*, **double** *version*) {

*this*.name **=** name;

*this*.developer **=** developer;

*this*.version **=** version;

    }

**public** **String** getName() {

**return** name;

    }

**public** **void** setName(**String** *name*) {

*this*.name **=** name;

    }

**public** **String** getDeveloper() {

**return** developer;

    }

**public** **void** setDeveloper(**String** *developer*) {

*this*.developer **=** developer;

    }

**public** **double** getVersion() {

**return** version;

    }

**public** **void** setVersion(**double** *version*) {

*this*.version **=** version;

    }

**public** **abstract** **void** launch();

}

**Класс Game.java (наследуется от Application):**

**public** **class** Game **extends** Application {

**private** **String** genre;

**private** **static** **int** instanceCount **=** 0;

**public** Game(**String** *name*, **String** *developer*, **double** *version*, **String** *genre*) {

*super*(name, developer, version);

*this*.genre **=** genre;

        instanceCount**++**;

    }

**public** **String** getGenre() {

**return** genre;

    }

**public** **void** setGenre(**String** *genre*) {

*this*.genre **=** genre;

    }

**public** **static** **int** getInstanceCount() {

**return** instanceCount;

    }

    @**Override**

**public** **void** launch() {

        System.out.println("Запуск игры: " **+** getName() **+** " версии " **+** getVersion());

    }

}

**Класс MobileGame.java (наследуется от Game):**

**public** **class** MobileGame **extends** Game {

**private** **String** platform;

**public** MobileGame(**String** *name*, **String** *developer*, **double** *version*, **String** *genre*, **String** *platform*) {

*super*(name, developer, version, genre);

*this*.platform **=** platform;

    }

**public** **String** getPlatform() {

**return** platform;

    }

**public** **void** setPlatform(**String** *platform*) {

*this*.platform **=** platform;

    }

    @**Override**

**public** **void** launch() {

        System.out.println("Запуск мобильной игры: " **+** getName() **+** " на платформе " **+** platform);

    }

}

**Часть 2: Демонстрация работы иерархии классов**

**Код демонстрационного класса Main.java:**

**public** **class** Main {

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

**Game** game **=** **new** Game("Chess", "ClassicGames Inc.", 1.0, "Strategy");

        game.launch();

        System.out.println("Жанр игры: " **+** game.getGenre());

        System.out.println("Количество созданных игр: " **+** Game.getInstanceCount());

**MobileGame** mobileGame **=** **new** MobileGame("Angry Birds", "Rovio", 2.3, "Arcade", "Android");

        mobileGame.launch();

        System.out.println("Платформа игры: " **+** mobileGame.getPlatform());

        System.out.println("Количество созданных игр: " **+** Game.getInstanceCount());

    }

}

**Описание работы программы:**

1. Создан абстрактный класс Application, который содержит общие поля и методы для всех приложений.
2. Класс Game наследуется от Application и добавляет поле genre, а также статическую переменную instanceCount, которая подсчитывает количество созданных объектов данного класса.
3. Класс MobileGame наследуется от Game и добавляет поле platform, а также переопределяет метод launch().
4. В классе Main демонстрируется создание объектов классов Game и MobileGame, вызов их методов и вывод информации на экран.

Выводы

В ходе выполнения лабораторной работы я изучил основные принципы объектно-ориентированного программирования, такие как наследование, инкапсуляция, полиморфизм и абстракция. Я научился создавать иерархии классов, использовать абстрактные классы, переопределять методы, а также работать с геттерами и сеттерами. Дополнительно был реализован счетчик созданных объектов с использованием статической переменной, что позволяет отслеживать количество экземпляров класса.