

**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»**

Кафедра Математическая кибернетика и информационные технологии

Отчет по лабораторной работе № 2

Выполнил: студент группы БПИ2401

Трухина Анастасия Александровна

Проверил: Харрасов Камиль Раисович

Москва,

2025

## Оглавление

Цель работы: .....	3
Задание: .....	3
Основная часть .....	3
Ответы на контрольные вопросы: .....	<b>Error! Bookmark not defined.</b>
Заключение .....	11

Цель работы:

Цель работы —

Задание:

Задание 1. Создайте иерархию классов в соответствии с вариантом. Ваша иерархия должна содержать:

- абстрактный класс;
- два уровня наследуемых классов (классы должны содержать в себе минимум 3 поля и 2 метода, описывающих поведение объекта);
- демонстрацию реализации всех принципов ООП;
- наличие конструкторов (в том числе по умолчанию);
- наличие геттеров и сеттеров;
- ввод/вывод информации о создаваемых объектах;
- предусмотрите в одном из классов создание счетчика созданных объектов с использованием статической переменной, продемонстрируйте работу

Вариант 12:

Базовый класс: Приложение.

Дочерние классы: Социальная сеть, Игра, Погода.

Основная часть

Реализуем базовый класс с полями и методами — в данном случае, абстрактный класс Application:

```
1. package Laba2;
2.
3. public abstract class Application {
4.
5.     private String name;
6.     private String description;
7.     private int version;
8.
9.     public Application(String name, String description, int version) {
10.         this.name = name;
11.         this.description = description;
12.         this.version = version;
13.     }
14.
15.     public Application(){
16.     }
17.
18.     public abstract void onStart();
```

```

19.         public abstract void onStop();
20.         public abstract void onDestroy();
21.
22.         public int getVersion(){
23.             return this.version;
24.         }
25.
26.         public void setVersion(int version){
27.             this.version = version;
28.         }
29.
30.         public String getDescription(){
31.             return this.description;
32.         }
33.
34.         public void setDescription(String description) {
35.             this.description = description;
36.         }
37.
38.         public String getName(){
39.             return this.name;
40.         }
41.
42.         public void setName(String name) {
43.             this.name = name;
44.         }
45.
46.     }
47.

```

Создадим дочерние классы согласно требованиям заданий:

```

1. package Laba2;
2.
3. public class Game extends Application {
4.
5.     private String genre;
6.     private boolean isMultiplayer;
7.     private int currentLevel;
8.
9.     public Game() {
10.         super();
11.     }
12.
13.     public Game(String name, String description, int version,
String genre, boolean isMultiplayer, int currentLevel) {
14.         super(name, description, version);
15.         this.genre = genre;
16.         this.isMultiplayer = isMultiplayer;
17.         this.currentLevel = currentLevel;
18.     }
19.
20.     public String getGenre() {
21.         return genre;
22.     }
23.
24.     public void setGenre(String genre) {
25.         this.genre = genre;
26.     }
27.
28.     public boolean isMultiplayer() {
29.         return isMultiplayer;

```

```

30.     }
31.
32.     public void setMultiplayer(boolean multiplayer) {
33.         isMultiplayer = multiplayer;
34.     }
35.
36.     public int getCurrentLevel() {
37.         return currentLevel;
38.     }
39.
40.     public void setCurrentLevel(int currentLevel) {
41.         this.currentLevel = currentLevel;
42.     }
43.
44.     public void startGame() {
45.         System.out.println("Запуск игры: " + getName());
46.         System.out.println("Жанр: " + genre + ", уровень: " +
currentLevel);
47.     }
48.
49.     public void saveProgress() {
50.         System.out.println("Прогресс сохранён. Текущий уровень: "
+ currentLevel);
51.     }
52.
53.     @Override
54.     public void onStart() {
55.         System.out.println("Игра " + getName() + " запущена.");
56.     }
57.
58.     @Override
59.     public void onStop() {
60.         System.out.println("Игра " + getName() + "
приостановлена.");
61.     }
62.
63.     @Override
64.     public void onDestroy() {
65.         System.out.println("Игра " + getName() + " завершена.");
66.     }
67.
68.     public void displayInfo() {
69.         System.out.println("=== Информация об игре ===");
70.         System.out.println("Название: " + getName());
71.         System.out.println("Описание: " + getDescription());
72.         System.out.println("Версия: " + getVersion());
73.         System.out.println("Жанр: " + genre);
74.         System.out.println("Мультиплеер: " + (isMultiplayer ?
"да" : "нет"));
75.         System.out.println("Текущий уровень: " + currentLevel);
76.         System.out.println("=====");
77.     }
78. }

```

```

1. package Laba2;
2.
3. import java.util.Scanner;
4.
5. public class SocialNetwork extends Application {
6.
7.     private String login;
8.     private String password;
9.     private String email;
10.
11.     public SocialNetwork(String name, String description, int
version,
12.                           String email, String password, String
login){
13.         super(name, description, version);
14.         this.login = login;
15.         this.password = password;
16.         this.email = email;
17.     }
18.
19.     public SocialNetwork(){
20.         super();
21.     }
22.
23.     public void setPassword(String password){
24.         this.password = password;
25.     }
26.
27.     public String getPassword(){
28.         return this.password;
29.     }
30.
31.     public void setLogin(String login){
32.         this.login = login;
33.     }
34.
35.     public String getLogin(){
36.         return this.login;
37.     }
38.
39.     public void setEmail(String email){
40.         this.email = email;
41.     }
42.
43.     public String getEmail(){
44.         return this.email;
45.     }
46.
47.     public void LoginWithEmail(){
48.         System.out.println("[ " + getName() + "]" +
49.                             "пользователь " + getLogin() +
50.                             " вошел с почтой " + getEmail());
51.     }
52.
53.     public void AddFriend(){
54.         System.out.println("Пользователь " + getLogin() +
"добавил друга");
55.     }
56.
57.     @Override
58.     public void onStart() {
59.         System.out.println("Соцсеть " + getName() + " запущена");
60.         if (!login.isEmpty() && (login != null)){
61.             System.out.println("Пользователь " + login + "
выполнил вход");
62.         } else {

```

```

63.         System.out.println("Пользователь " + login + " не
выполнял вход");
64.     }
65. }
66.
67. @Override
68. public void onStop() {
69.     System.out.println("Соцсеть приостановлена");
70. }
71.
72. @Override
73. public void onDestroy() {
74.     System.out.println("Приложение " + getName() + "
завершает работу");
75. }
76.
77. public void displayInfo() {
78.     System.out.println("=== Информация о приложении ===");
79.     System.out.println("Название: " + getName());
80.     System.out.println("Описание: " + getDescription());
81.     System.out.println("Версия: " + getVersion());
82.     System.out.println("Логин: " + login);
83.     System.out.println("Email: " + email);
84.     System.out.println("=====");
85. }
86.
87. }

```

```

1. package Laba2;
2.
3. public class WeatherApp extends Application {
4.
5.     private static int instanceCount = 0;
6.
7.     private String location;
8.     private double temperature;
9.     private boolean isCelsius;
10.
11.     public WeatherApp() {
12.         super();
13.         instanceCount++;
14.     }
15.
16.     public WeatherApp(String name, String description, int
version,
17.         String location, double temperature,
boolean isCelsius) {
18.         super(name, description, version);
19.         this.location = location;
20.         this.temperature = temperature;
21.         this.isCelsius = isCelsius;
22.         instanceCount++;
23.     }
24.
25.     public String getLocation() {
26.         return location;
27.     }
28.
29.     public void setLocation(String location) {
30.         this.location = location;

```

```

31.     }
32.
33.     public double getTemperature() {
34.         return temperature;
35.     }
36.
37.     public void setTemperature(double temperature) {
38.         this.temperature = temperature;
39.     }
40.
41.     public boolean isCelsius() {
42.         return isCelsius;
43.     }
44.
45.     public void setCelsius(boolean celsius) {
46.         isCelsius = celsius;
47.     }
48.
49.     public void updateweather() {
50.         System.out.println("Обновление погоды в "
51.             + location + ": "
52.             + temperature + "°"
53.             + (isCelsius ? "C" : "F"));
54.     }
55.
56.     public void showForecast() {
57.         System.out.println("Прогноз: погода в "
58.             + location + " на сегодня - "
59.             + temperature + "°");
60.     }
61.
62.     @Override
63.     public void onStart() {
64.         System.out.println("Погодное приложение " + getName() + "
запущено.");
65.     }
66.
67.     @Override
68.     public void onStop() {
69.         System.out.println("Погода " + getName() + "
приостановлена.");
70.     }
71.
72.     @Override
73.     public void onDestroy() {
74.         System.out.println("Погодное приложение " + getName() + "
завершено.");
75.     }
76.
77.     public void displayInfo() {
78.         System.out.println("=== Информация о погодном приложении
===");
79.         System.out.println("Название: " + getName());
80.         System.out.println("Описание: " + getDescription());
81.         System.out.println("Версия: " + getVersion());
82.         System.out.println("Местоположение: " + location);
83.         System.out.println("Температура: " + temperature + "°" +
(isCelsius ? "C" : "F"));
84.         System.out.println("=====");
85.     }
86.
87.     public static int getInstanceCount() {
88.         return instanceCount;
89.     }
90. }

```



Работа кода:

```
1. package Laba2;
2.
3. public class Run {
4.     public static void main(String[] args) {
5.         SocialNetwork sn = new SocialNetwork("vk", "Социальная сеть",
6.         10, "test@example.com", "password123", "user123");
7.         Game game = new Game("Minecraft", "Игра", 1, "Sandbox", true,
8.         3);
9.         WeatherApp weather = new WeatherApp("Accuweather", "Погода",
10.        5, "Moscow", 18.5, true);
11.
12.        System.out.println("=== Социальная сеть ===");
13.        sn.setLogin("newUser");
14.        sn.setEmail("new@example.com");
15.        sn.setPassword("newPassword");
16.        System.out.println("Логин: " + sn.getLogin());
17.        System.out.println("Email: " + sn.getEmail());
18.        System.out.println("Пароль: " + sn.getPassword());
19.        sn.LogInWithEmail();
20.        sn.AddFriend();
21.        sn.onStart();
22.        sn.onStop();
23.        sn.onDestroy();
24.        sn.displayInfo();
25.
26.        System.out.println("\n=== Игра ===");
27.        game.setGenre("RPG");
28.        game.setMultiplayer(false);
29.        game.setCurrentLevel(10);
30.        System.out.println("Жанр: " + game.getGenre());
31.        System.out.println("Мультиплеер: " +
32.        game.isMultiplayer());
33.        System.out.println("Уровень: " + game.getCurrentLevel());
34.        game.startGame();
35.        game.saveProgress();
36.        game.onStart();
37.        game.onStop();
38.        game.onDestroy();
39.        game.displayInfo();
40.
41.        System.out.println("\n=== Погодное приложение ===");
42.        weather.setLocation("London");
43.        weather.setTemperature(15.0);
44.        weather.setCelsius(false);
45.        System.out.println("Местоположение: " +
46.        weather.getLocation());
47.        System.out.println("Температура: " +
48.        weather.getTemperature());
49.        System.out.println("Цельсий: " + weather.isCelsius());
50.        weather.updateweather();
51.        weather.showForecast();
52.        weather.onStart();
53.        weather.onStop();
54.        weather.onDestroy();
55.        weather.displayInfo();
56.
57.        System.out.println("\n=== Счётчик ===");
58.        System.out.println("Создано погодных приложений: " +
59.        WeatherApp.getInstanceCount());
60.    }
61. }
```

```
"C:\Program Files\Java\jdk-22\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2024.2.5\lib\idea_rt.jar=61095:C:\Program Files\
=== Социальная сеть ===
Логин: newUser
Email: new@example.com
Пароль: newPassword
[ VK]пользователь newUser вошел с почтой new@example.com
Пользователь newUserдобавил друга
Соцсеть VKзапущена
Пользователь newUser выполнил вход
Соцсеть приостановлена
Приложение VK завершает работу
=== Информация о приложении ===
Название: VK
Описание: Социальная сеть
Версия: 10
Логин: newUser
Email: new@example.com
=====

=== Игра ===
Жанр: RPG
Мультиплеер: false
Уровень: 10
Запуск игры: Minecraft
Жанр: RPG, уровень: 10
Прогресс сохранён. Текущий уровень: 10
Игра Minecraft запущена.
Игра Minecraft приостановлена.
Игра Minecraft завершена.
=== Информация об игре ===
Название: Minecraft
Описание: Игра
Версия: 1
Жанр: RPG
```

```
Игра Minecraft запущена.
Игра Minecraft приостановлена.
Игра Minecraft завершена.
=== Информация об игре ===
Название: Minecraft
Описание: Игра
Версия: 1
Жанр: RPG
Мультиплеер: Нет
Текущий уровень: 10
=====

=== Погодное приложение ===
Местоположение: London
Температура: 15.0
Цельсий: false
Обновление погоды в London: 15.0°F
Прогноз: погода в London на сегодня - 15.0°
Погодное приложение AccuWeather запущено.
Погода AccuWeather приостановлена.
Погодное приложение AccuWeather завершено.
=== Информация о погодном приложении ===
Название: AccuWeather
Описание: Погода
Версия: 5
Местоположение: London
Температура: 15.0°F
=====

=== Счётчик ===
Создано погодных приложений: 1

Process finished with exit code 0
```

Ответы на контрольные вопросы:

1. Что такое абстракция и как она реализуется в языке Java?

Абстракция — это выделение существенных характеристик объекта и сокрытие деталей реализации. В Java она реализуется с помощью абстрактных классов и интерфейсов, которые определяют общий шаблон поведения, оставляя конкретные реализации подклассам. Абстрактный класс может содержать как абстрактные, так и обычные методы, но создать его экземпляр нельзя.

2. Что такое инкапсуляция и как она реализуется в языке Java?

Инкапсуляция — это сокрытие внутренней реализации объекта от внешнего мира и предоставление доступа к данным только через определённые методы. В Java это достигается с помощью модификаторов доступа, особенно `'private'` для полей и `'public'` для геттеров и сеттеров, что защищает данные от несанкционированного изменения.

3. Что такое наследование и как оно реализуется в языке Java?

Наследование — это механизм, позволяющий одному классу наследовать поля и методы другого, что помогает избежать дублирования кода и упрощает его поддержку. В Java наследование реализуется с помощью ключевого слова `'extends'`, и класс может наследоваться только от одного другого класса, но реализовывать несколько интерфейсов.

4. Что такое полиморфизм и как он реализуется в языке Java?

Полиморфизм позволяет объектам разных типов реагировать по-разному на один и тот же вызов. В Java он реализуется через переопределение методов (динамический полиморфизм) и перегрузку методов (статический полиморфизм), что позволяет использовать один интерфейс для разных реализаций.

5. Что такое множественное наследование и есть ли оно в Java?

Множественное наследование — это возможность класса наследовать сразу от нескольких классов, что может привести к неоднозначности. В Java это запрещено для классов, но разрешено через интерфейсы, что позволяет классу реализовать несколько контрактов одновременно.

#### 6. Для чего нужно ключевое слово `final`?

Ключевое слово `final` используется для ограничения изменений: переменные не могут быть переназначены, методы не могут быть переопределены, а классы — наследованы. Это помогает защитить важные элементы программы от изменения.

#### 7. Какие в Java есть модификаторы доступа?

В Java есть четыре уровня доступа: `public` — доступ везде, `protected` — доступ в классе, пакете и подклассах, `package-private` (по умолчанию) — доступ только в пакете, и `private` — доступ только внутри класса.

#### 8. Что такое конструктор? Какие типы конструкторов бывают в Java?

Конструктор — это специальный метод, вызываемый при создании объекта, инициализирующий его состояние. В Java бывают конструкторы по умолчанию (без параметров) и параметризованные (с параметрами), а также конструктор копирования, который создаёт объект на основе другого.

#### 9. Для чего нужно ключевое слово `this` в Java?

Ключевое слово `this` ссылается на текущий экземпляр класса. Оно используется для разрешения конфликта имён, вызова другого конструктора и возврата текущего объекта, что делает код более читаемым и понятным.

#### 10. Для чего нужно ключевое слово `super` в Java?

Ключевое слово ``super`` ссылается на суперкласс и используется для вызова конструктора родителя или доступа к его методам и полям. Это особенно полезно при переопределении методов, когда нужно вызвать реализацию родителя.

#### 11. Что такое геттеры и сеттеры? Зачем они нужны?

Геттеры — это методы, возвращающие значение приватного поля, а сеттеры — устанавливающие его. Они нужны для контролируемого доступа к данным, обеспечивая инкапсуляцию и возможность валидации при изменении значений.

#### 12. Что такое переопределение?

Переопределение — это предоставление новой реализации метода, объявленного в суперклассе. Переопределённый метод должен иметь ту же сигнатуру, и его обычно помечают аннотацией ``@Override`` для проверки компилятором.

#### 13. Что такое перегрузка?

Перегрузка — это наличие нескольких методов с одинаковым именем, но разными параметрами. Компилятор выбирает нужный метод в зависимости от переданных аргументов, что позволяет использовать одинаковые имена для схожих действий.

#### Заключение

Вывод: В ходе выполнения лабораторной работы была реализована иерархия классов в соответствии с вариантом №12: базовый абстрактный класс `Application` и три дочерних класса — `SocialNetwork`, `Game`, `WeatherApp`. Все требования задания соблюдены: созданы абстрактный класс, реализованы конструкторы, геттеры и сеттеры, добавлены методы поведения, предусмотрен счётчик объектов в классе `WeatherApp`, а также продемонстрированы все принципы объектно-ориентированного программирования — инкапсуляция, наследование, полиморфизм и

абстракция. Работа показала важность использования ООП-подходов для создания структурированного и удобного в сопровождении кода.

Ссылка на ГитХаб с файлами кода: [NT-005-TN/ITiP\\_LabWorks\\_Trukhina](#)