**МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ**

**Ордена Трудового Красного Знамени**

**Федеральное государственное бюджетное образовательное учреждение высшего образования**

**«Московский технический университет связи и информатики»**

Кафедра «Математическая Кибернетика и Информационные технологии»

Дисциплина «Информационные технологии и программирование»

Лабораторная работа №2

«Введение в ООП Java»

Выполнила:

Студентка группы БВТ2303

Морозова Ольга

**Цель работы:**

Изучение понятия ООП, основных его концепций и применение его на практике на языке программирования Java.

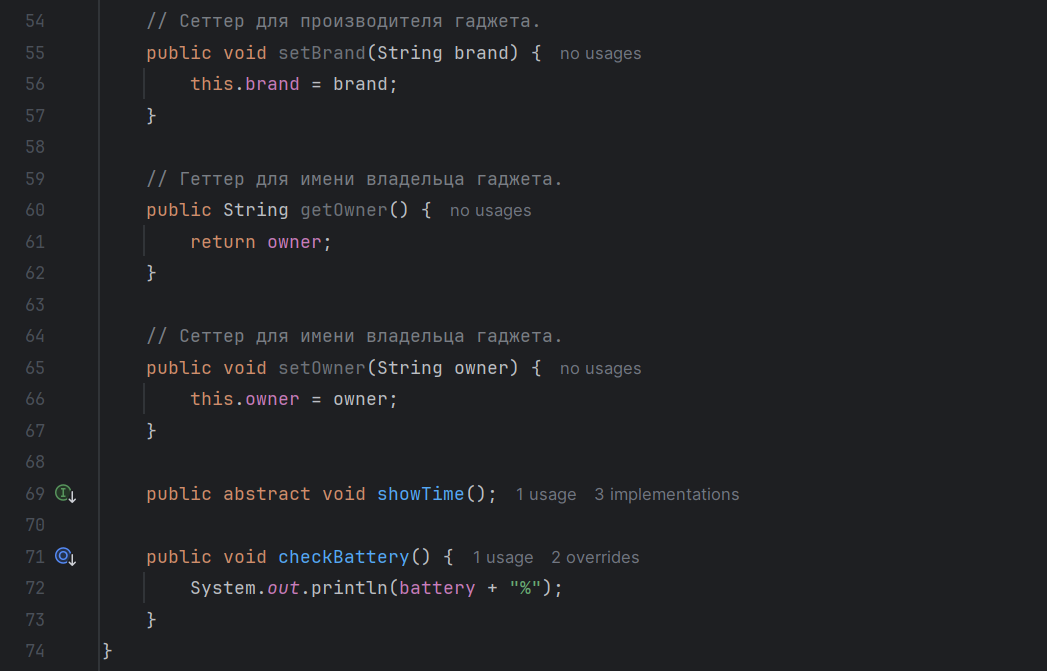
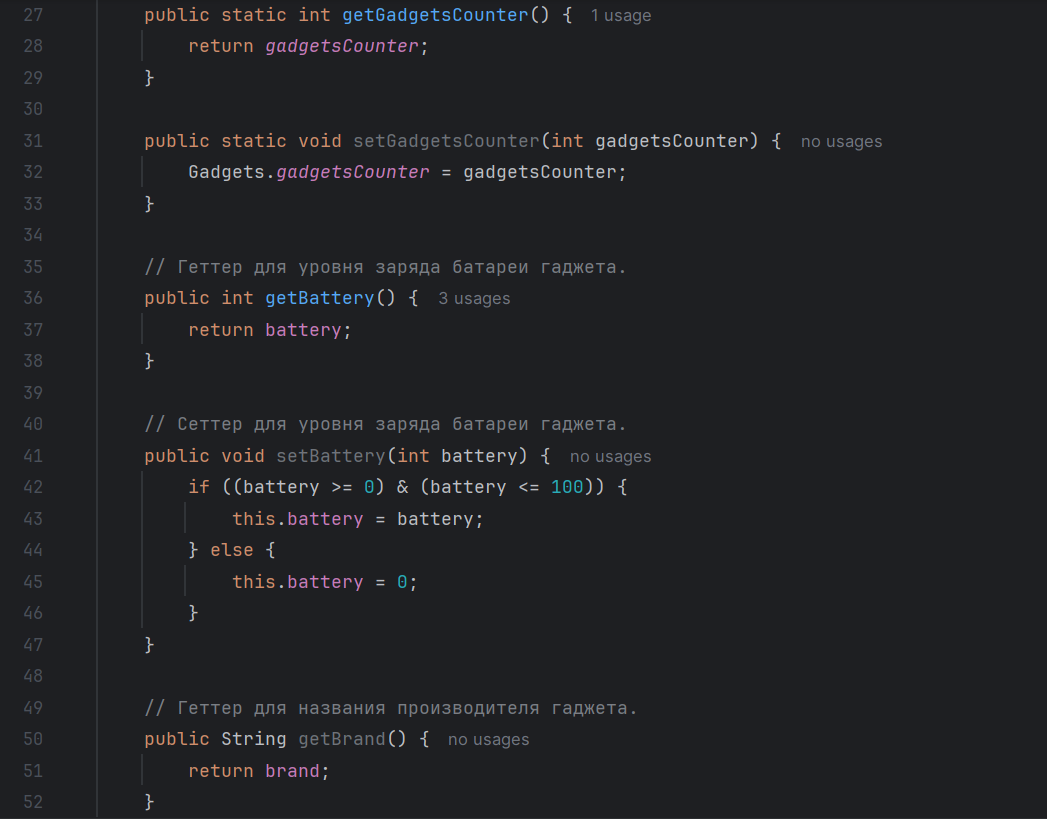
**Ход работы:**

Задание 1. Вариант 10.

1. Задание абстрактного класса.

Создаём базовый абстрактный класс Gadgets. В нём задаём поля (переменные класса) и определяем методы (хотя бы их основы), которые обязательны для всех дочерних классов (*это абстракция*). Обязательно добавляем конструкторы класса (с введёнными данными и по умолчанию), а также геттеры и сеттеры для каждого созданного поля класса. Также именно в этом классе создаём счётчик объектов, в котором будут учитываться все созданные объекты как этого класса, так и его дочерних классов.

Все методы класса имеют модификатор доступа public, что позволяет дочерним классам обращаться к этим методам и использовать их. Но все поля класса имеют модификатор доступа private, ведь это внутренняя информация класса, и к ней не должны иметь доступ напрямую. Именно для взаимодействия с переменными в данном случае и создаются геттеры и сеттеры (*это инкапсуляция*).

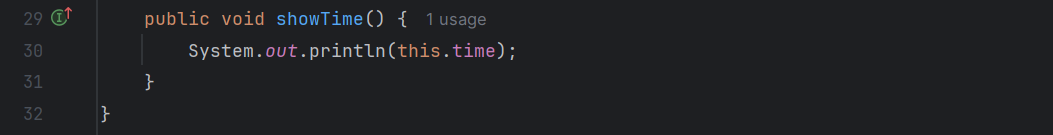
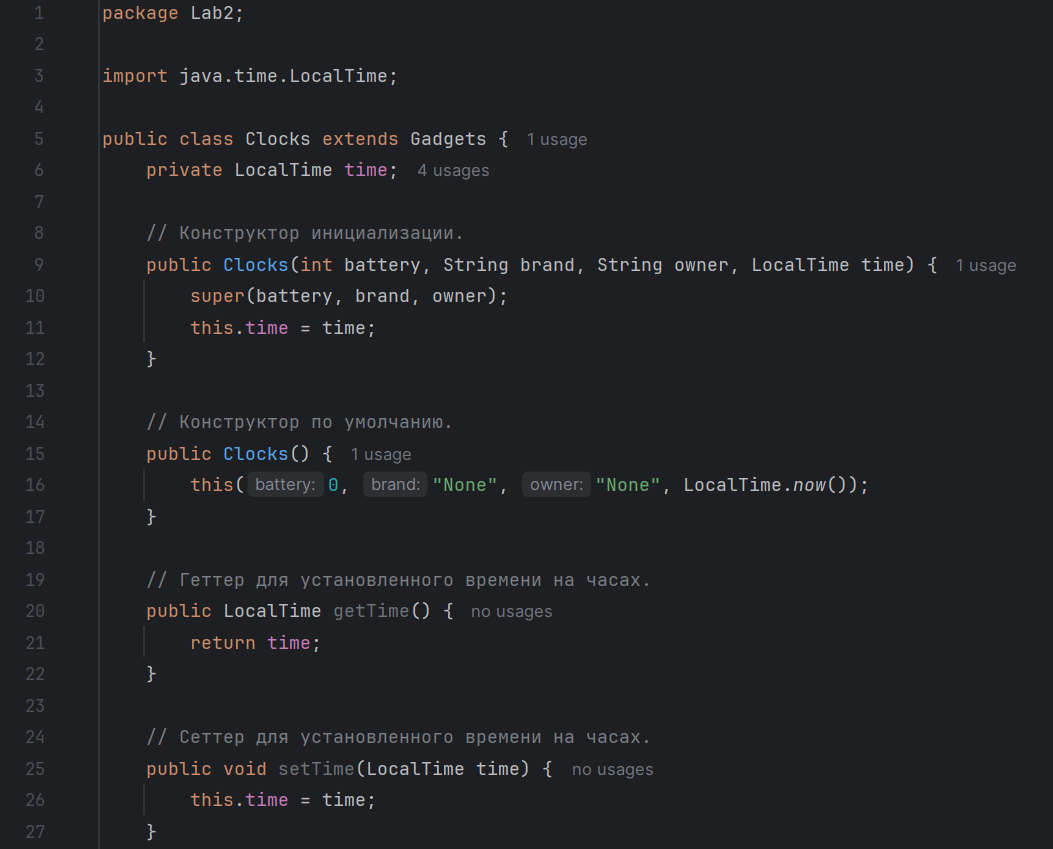


1. Создаём дочерние классы.

Для создания дочерних классов для абстрактного класса Gadgets мы используем extends после имени класса и указываем родительский класс. С помощью super() мы используем конструктор родительского класса.

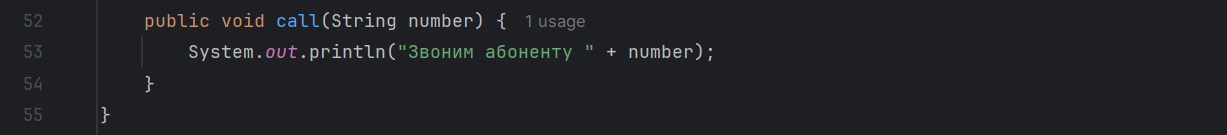
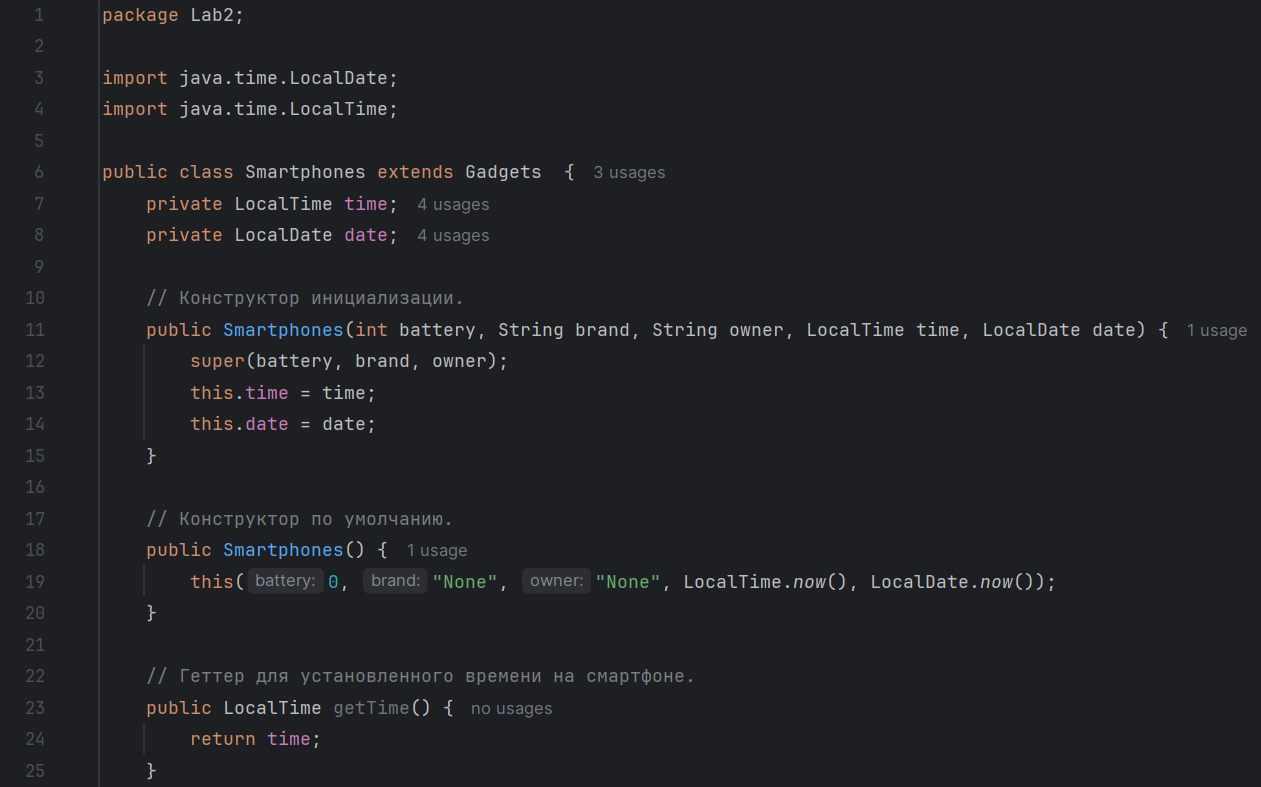
Дочерний класс Clocks.

Мы определяем абстрактный метод showTime(), который указывали в родительском классе. При этом нам не нужно заново прописывать метод chackBattery, т.к. его базовый функционал, прописанный в родительском классе, нас удовлетворяет. Это и есть *наследование*.



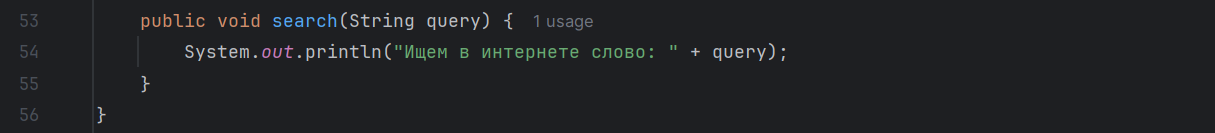
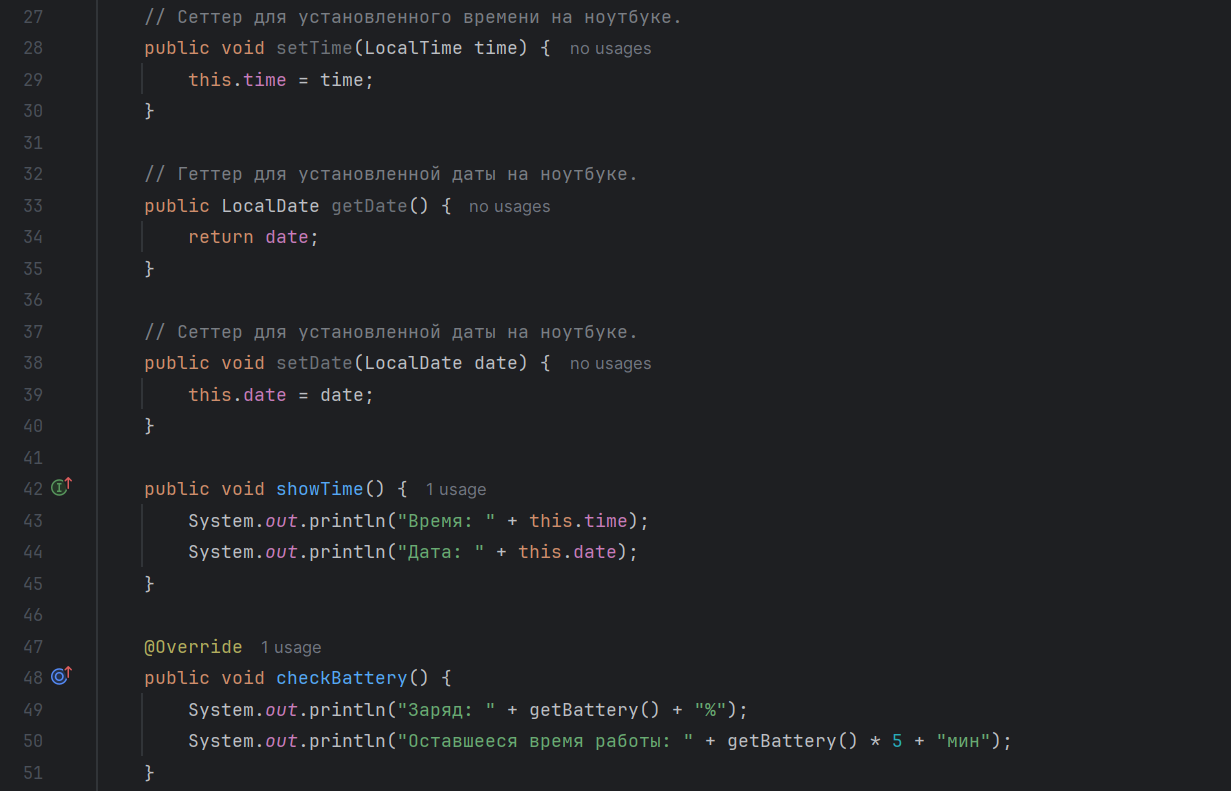
Дочерний класс SmartPhones.

В данном случае мы переопределяем родительский класс checkBattery для дочернего, т.к. хотим, чтобы для данного класса он работал по-другому. Также мы добавляем уникальный метод call(), которого не было в родительском классе.



Дочерний класс Laptops.

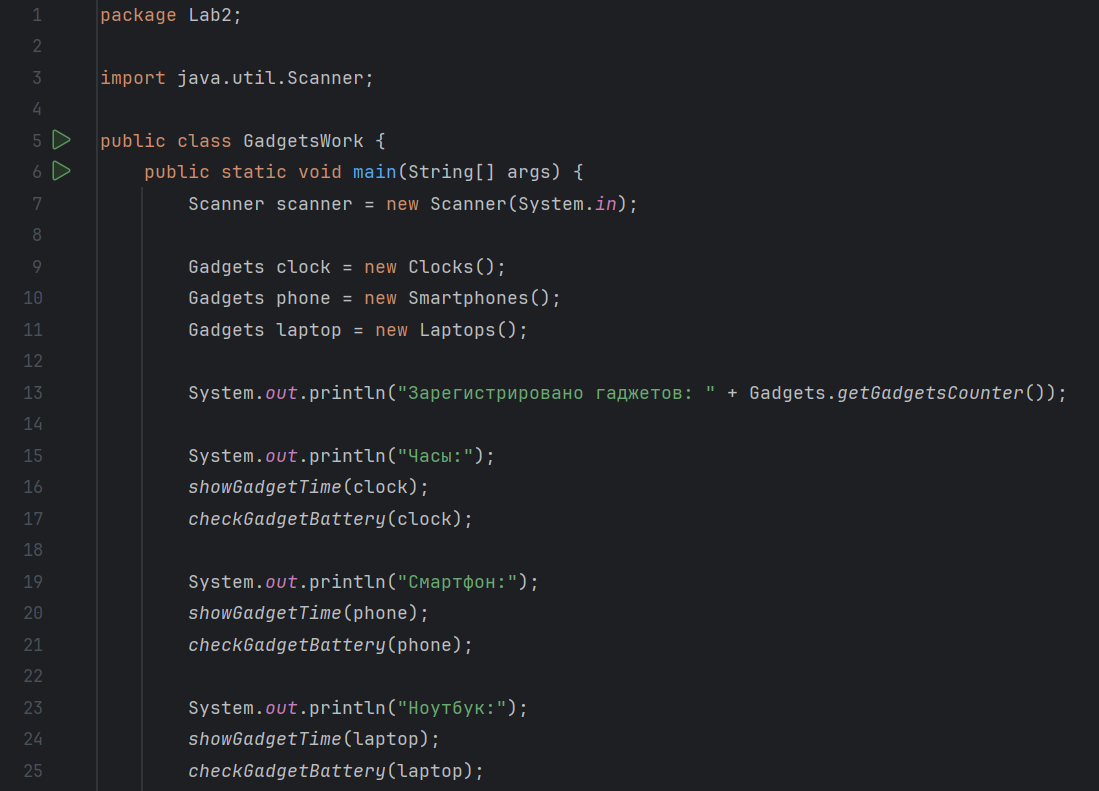
Как и в предыдущем случае мы переопределяем родительский класс checkBattery для дочернего и добавляем уникальный метод search(), которого не было в родительском классе.



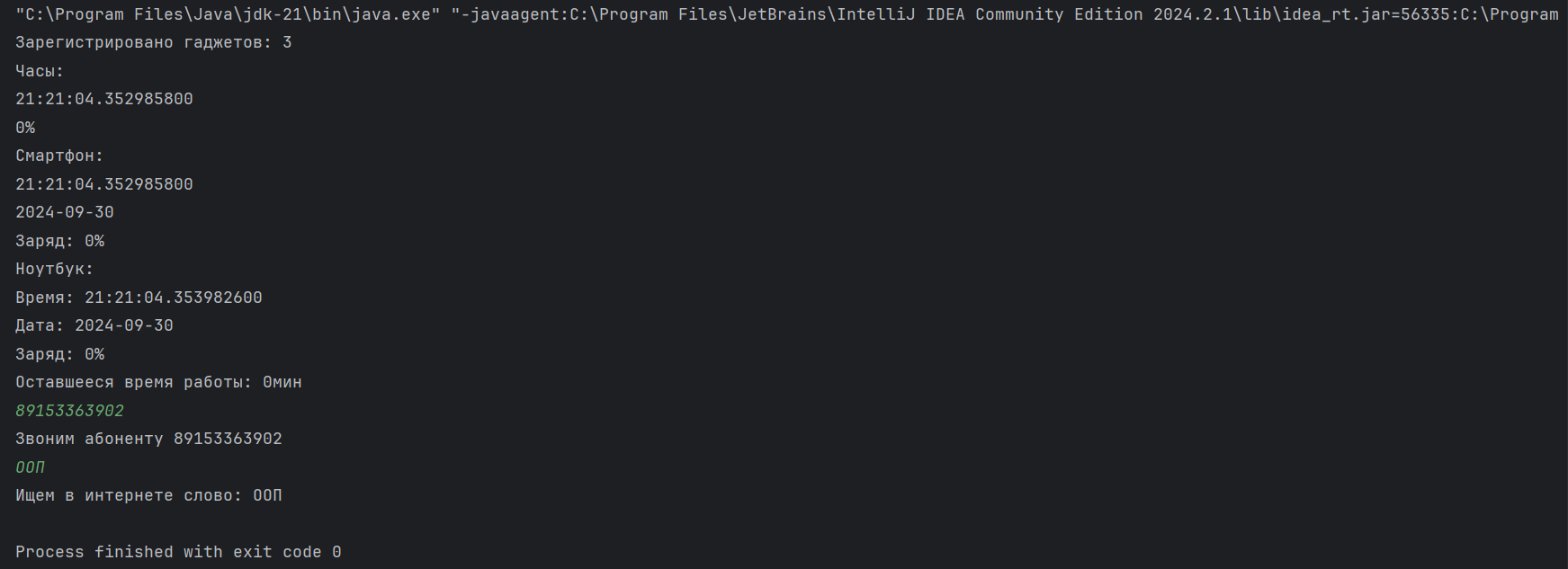
1. Создаём класс для работы с остальными.

В классе GadgetsWork мы, наконец, создаём метод main, в котором и будем создавать объекты созданных ранее классов, а также вызывать их методы.

Описывая методы для работы с объектами классов, мы создаём их из родительского абстрактного класса Gadgets, и его же задаём как тип данных в общих для всех методах. Это в итоге позволяет нам вызывать метод, переопределённый для каждого дочернего класса по-своему, используя лишь один метод в нашем текущем классе (*это полиморфизм*).



1. Наконец, тестируем работу программы.



**Вывод:**

Мы изучили понятие ООП, основные его концепции, а также написали программу на языке программирования Java с использованием полученных знаний.

GitHub - https://github.com/MiniLynx13/ITaP\_Lab2