Санкт-Петербургский политехнический университет Петра Великого

Институт компьютерных наук и кибербезопасности

Высшая школа «Компьютерных технологий и информационных систем»

**ОТЧЕТ по лабораторной работе №2**

на тему “Типы данных в Java” по дисциплине

«Системный подход в разработке программного обеспечения»

**Выполнил:**

студент группы 5130902/20201 \_\_\_\_\_\_\_\_\_\_\_\_\_ А. И. Сафонов

подпись

**Проверил:**

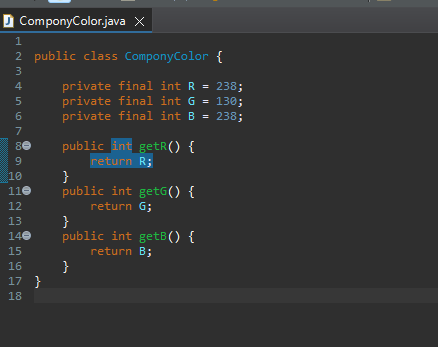
Кандидат тех. Наук., доцент \_\_\_\_\_\_\_\_\_\_\_\_ С. А. Нестеров

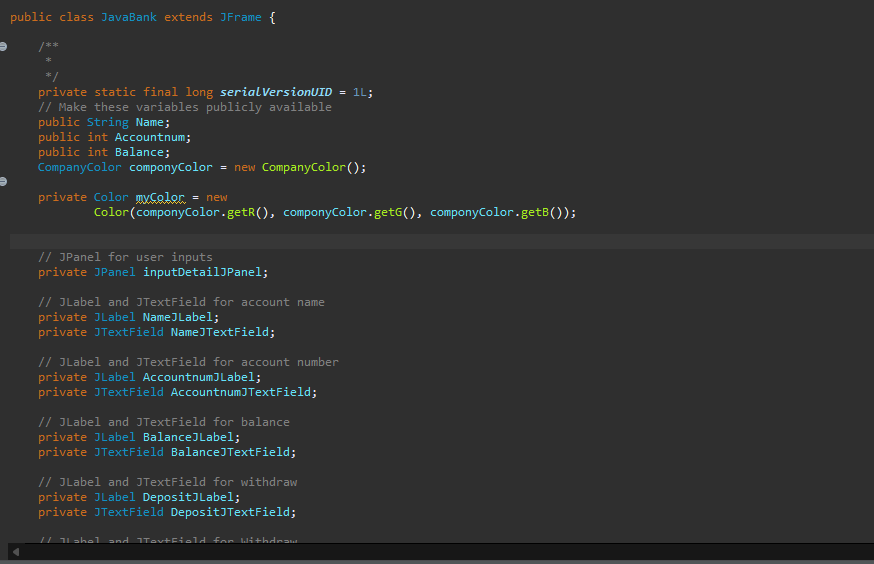
подпись

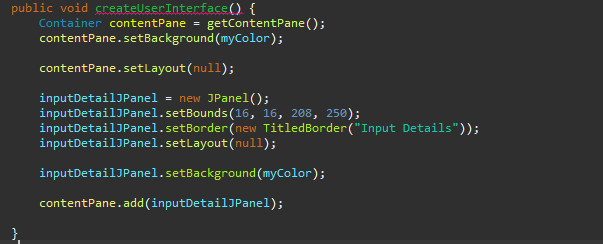
«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_ 2025г.

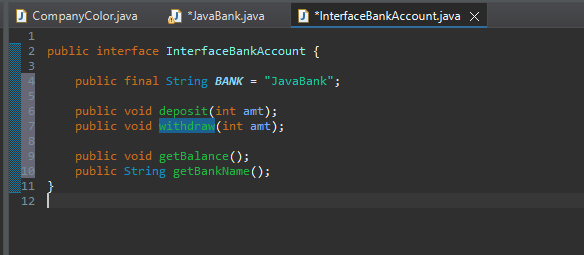
Санкт-Петербург, 2025

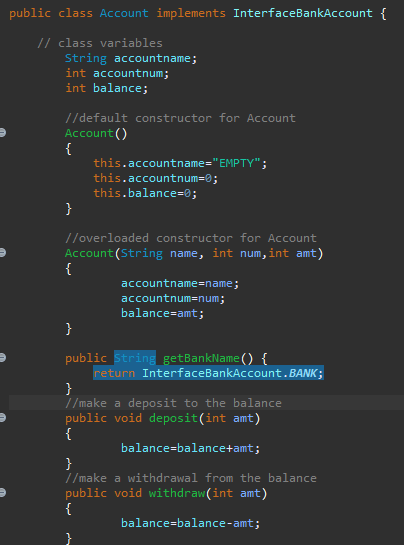
Задание из слайдов урока 2.2



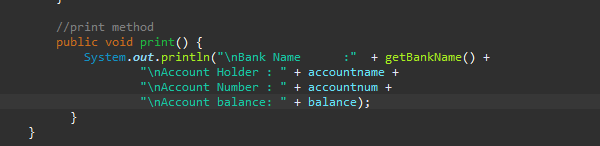


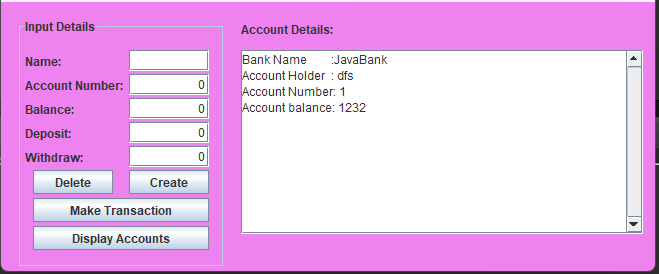




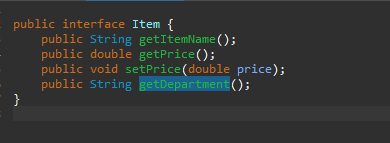


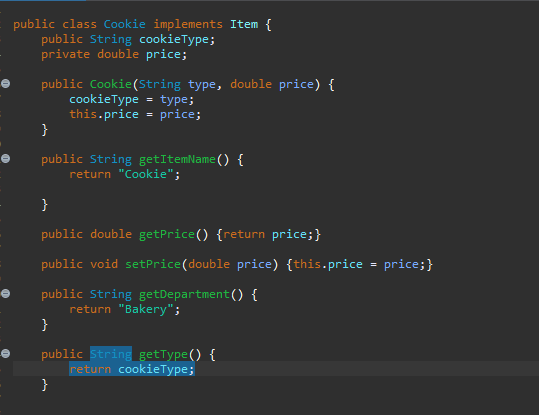


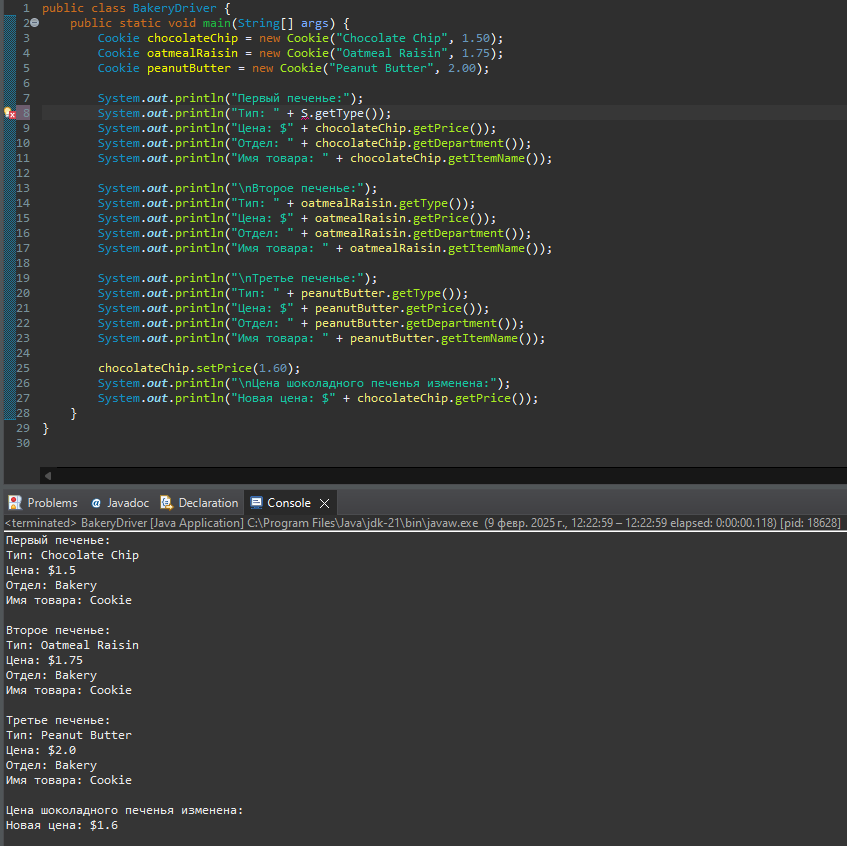




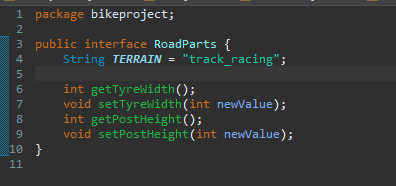
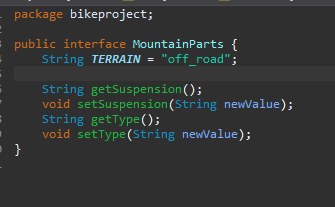
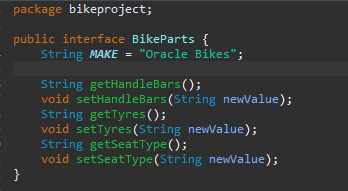
Bakery

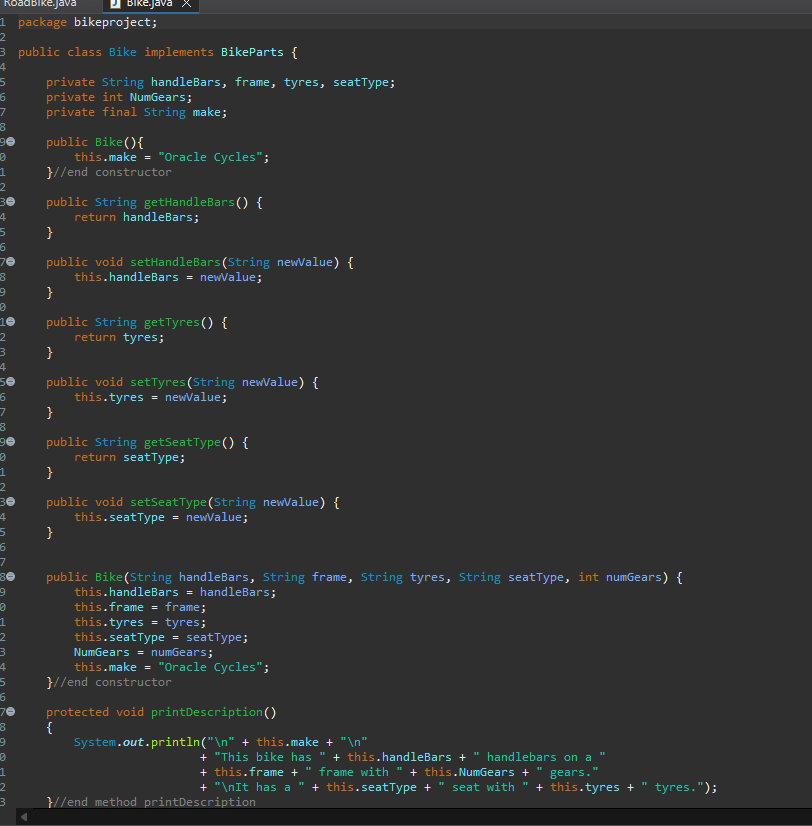
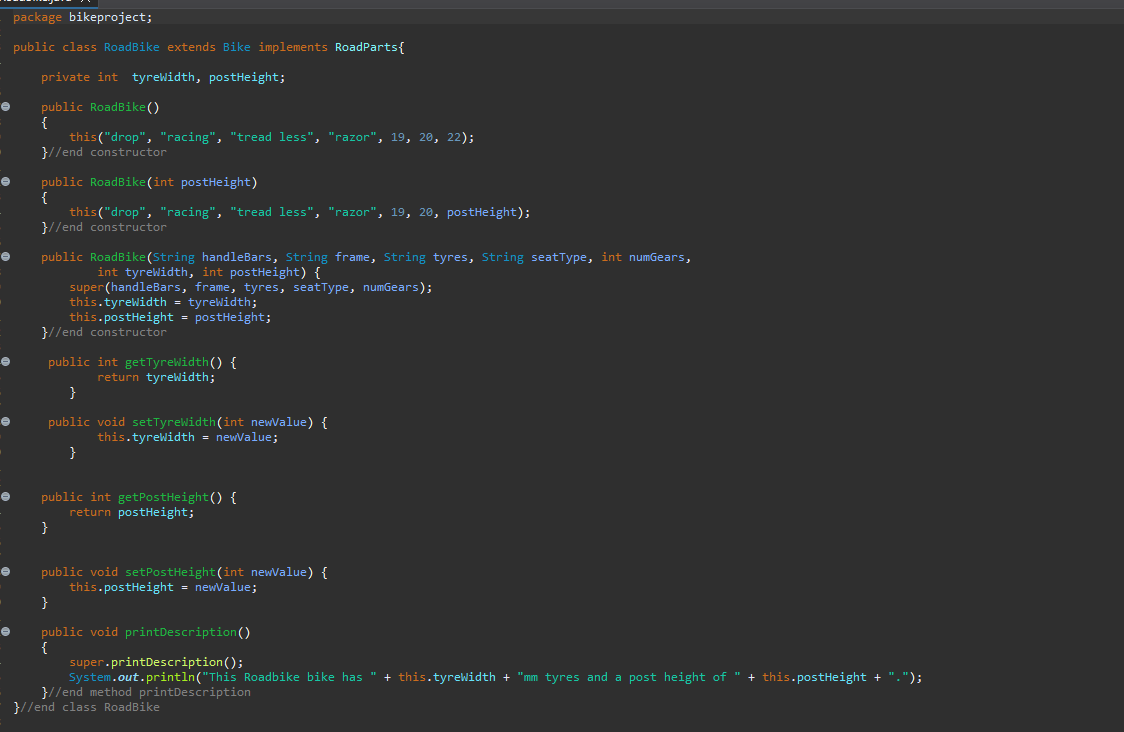
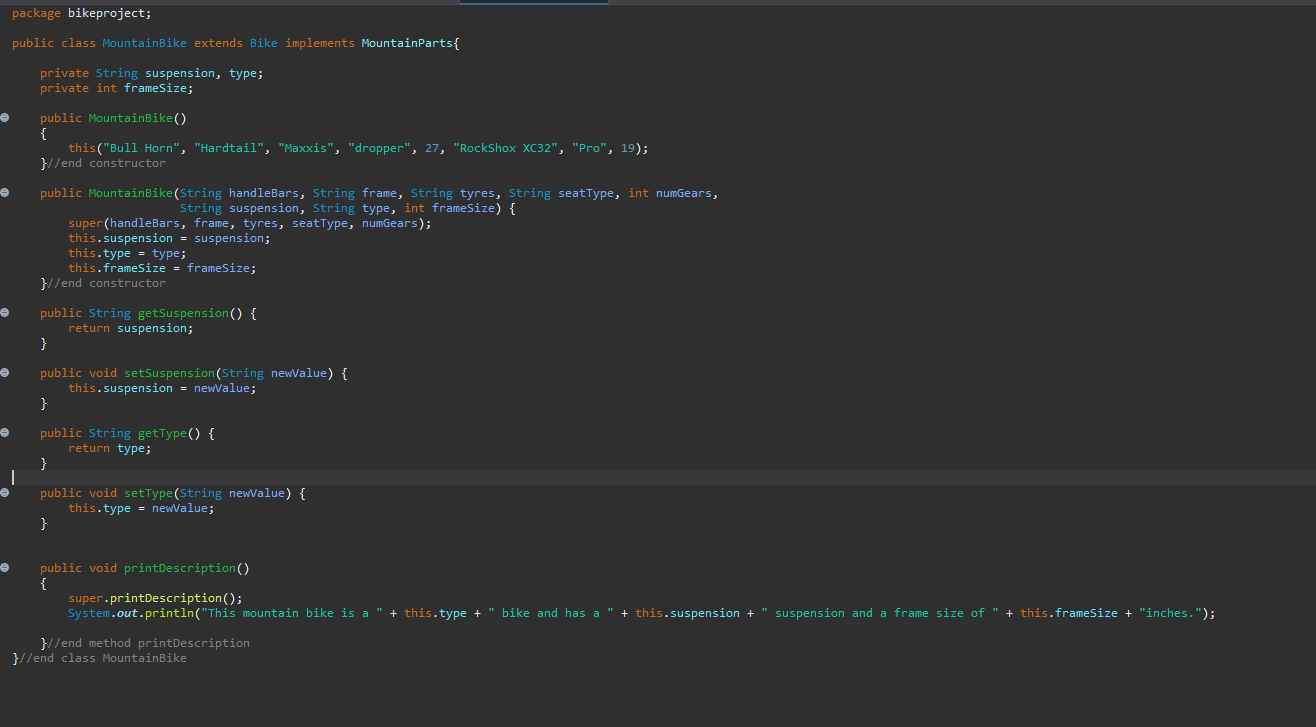


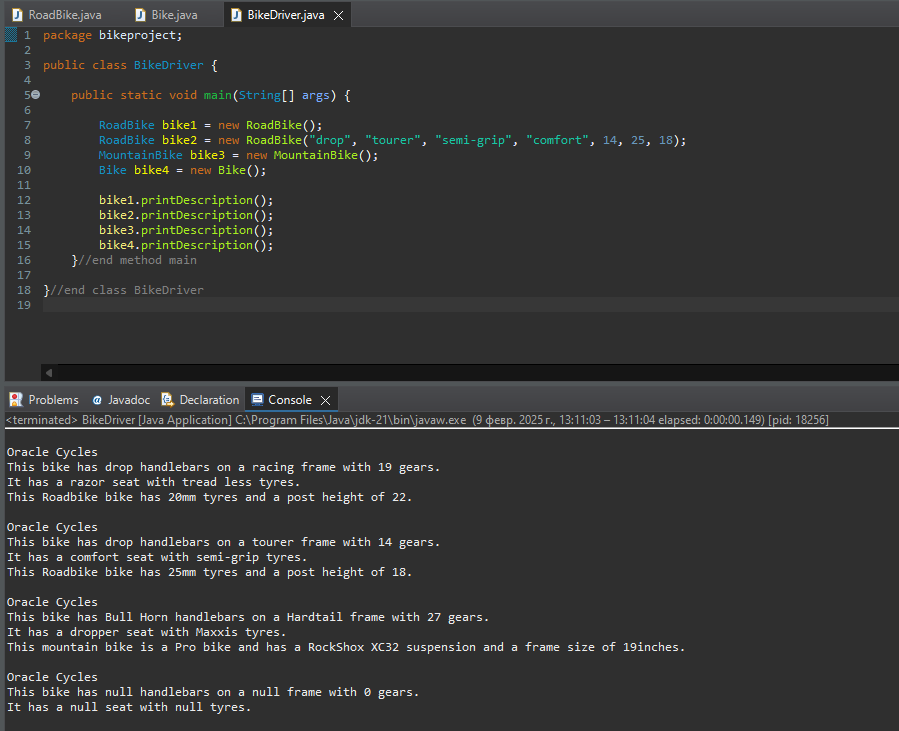


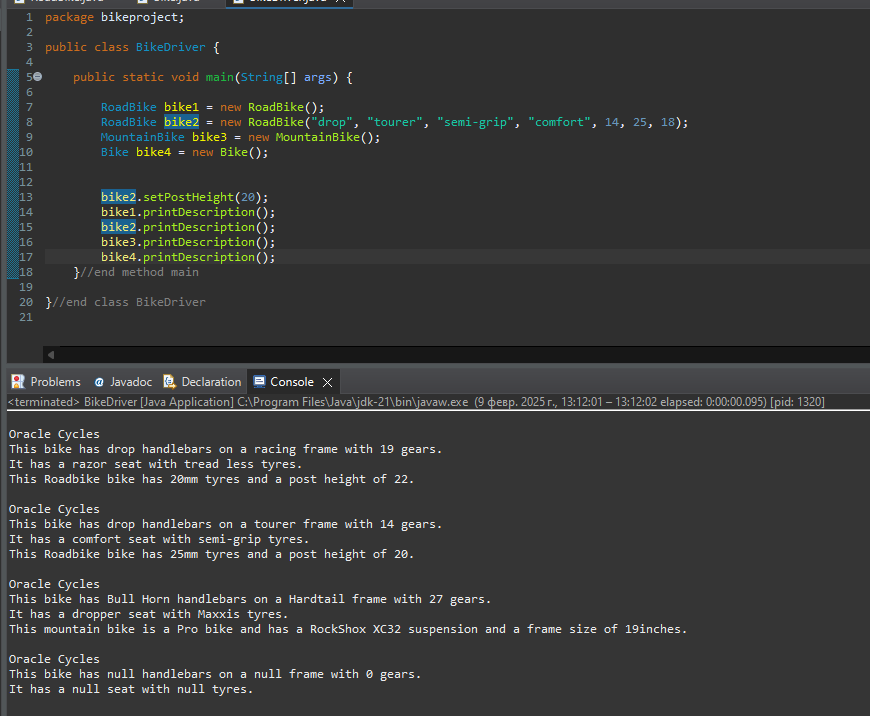


MounterBike

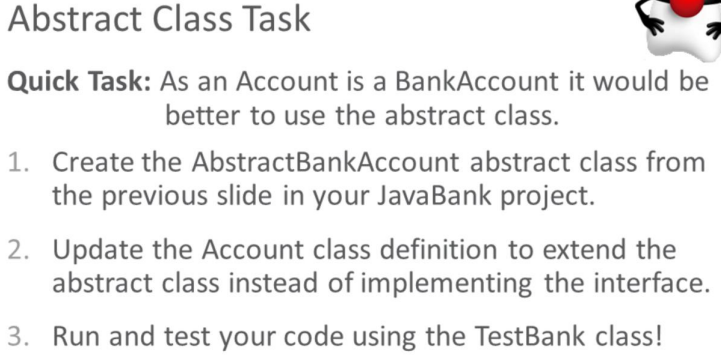


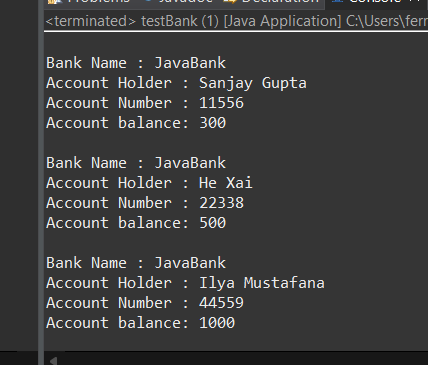
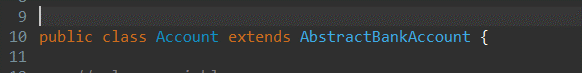
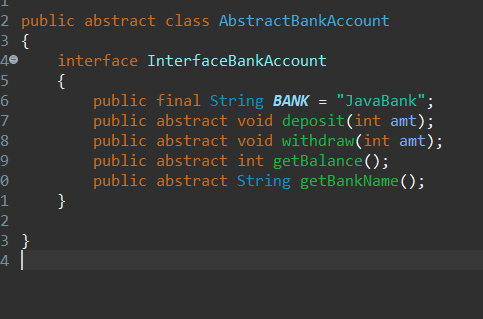




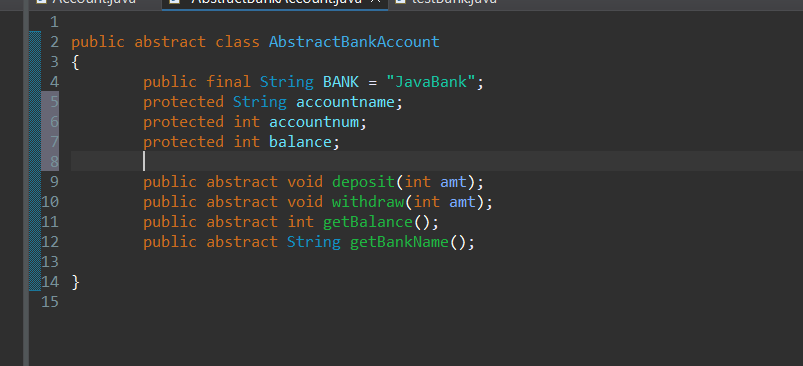


Проделайте упражнения из слайдов урока 2.3 (до слайда 36)

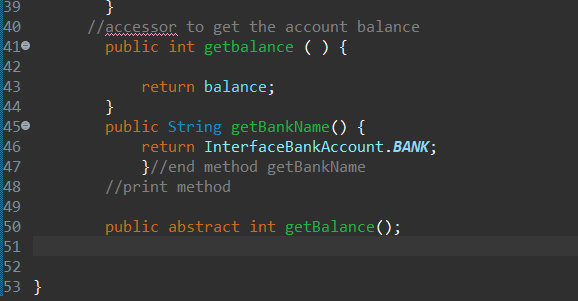


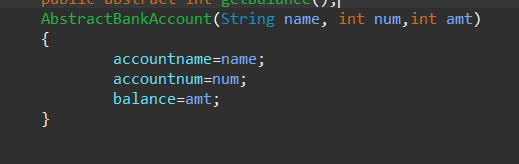


**Обновление классов Account и AbstractBankAccount**

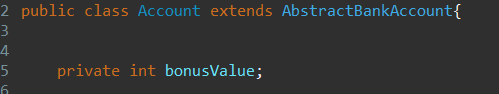


Переместите соответствующие методы (getter/setter) из класса Account в абстрактный класс BankAccount. Метод withdraw тоже должен быть перенесён, так как он выполняется одинаково как для Account, так и для CreditAccount.

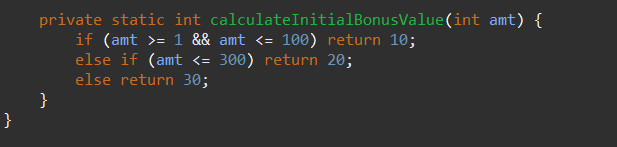




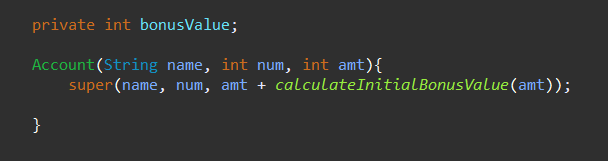
Добавьте в класс Account приватное поле экземпляра bonusValue, которое будет хранить целочисленное значение.



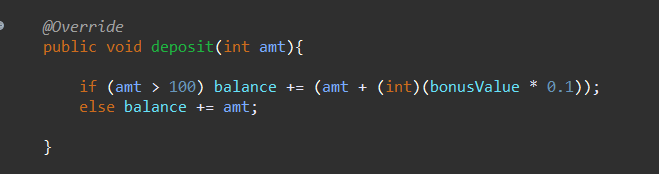
Создайте статический метод с именем calculateInitialBonusValue, который возвращает целочисленное значение и выполняет этот расчет.



Обновите конструктор в классе Account, чтобы он вызывал конструктор суперкласса AbstractBankAccount. Используйте name, num и результат суммы amt + calculateInitialBonusValue(amt) в качестве аргументов



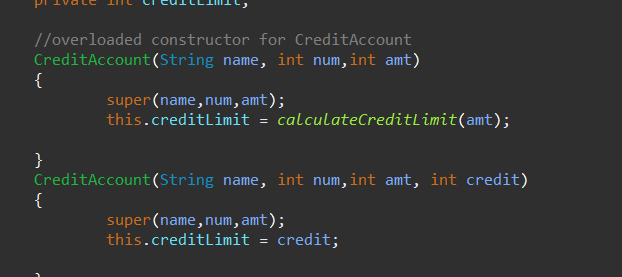
Если у вас обычный банковский счет (не кредитный), вы получите дополнительные 10% от вашего начального бонуса при каждом депозите суммой более $100. (Этот бонус доступен только для обычного счета. Кредитные счета не получают эту надбавку)



Сделайте так, чтобы класс CreditAccount наследовался от AbstractBankAccount вместо Account. 

Когда вы наследуете абстрактный класс, необходимо реализовать все абстрактные методы, которые в нем определены. Для AbstractBankAccount это означает, что в CreditAccount нужно реализовать метод deposit, который:

* Принимает целочисленный параметр (int amt)
* Добавляет его к текущему значению баланса (balance)

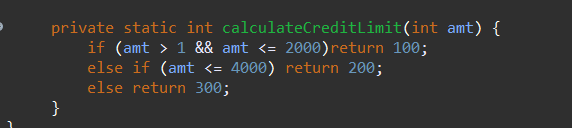


Уже существует конструктор для базовой кредитной учетной записи, где назначен кредитный лимит в размере 100 долларов США. JavaBank хочет внедрить новую схему, чтобы упростить обработку кредитных счетов: если вы открываете кредитный счет, вы получаете лучший кредитный лимит в зависимости от размера вашего первоначального депозита.

- от 1 до 2000 долларов вы получаете 100 долларов

- от 2001 до 4000 долларов вы получаете 200 долларов

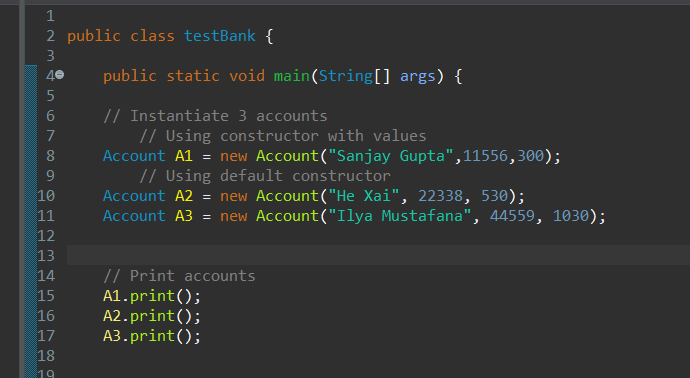
- свыше 4000 долларов вы получаете 300 долларов

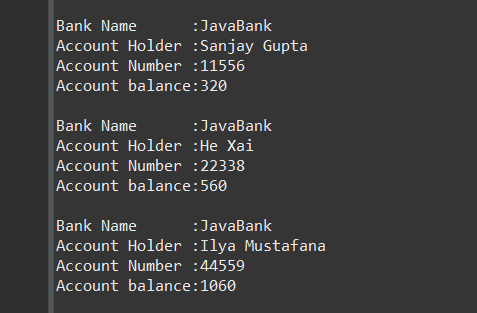


**Тестирование абстрактного класса 1**

1. Обновите класс TestBank, чтобы значения для учетных записей a2 и a3 предоставлялись через конструктор, а не через вызовы метода установки.

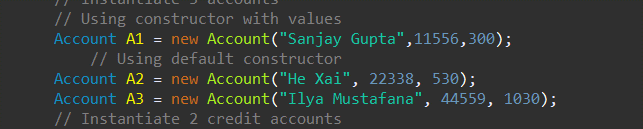
2. Удалите вызовы метода установки из класса TestBank!



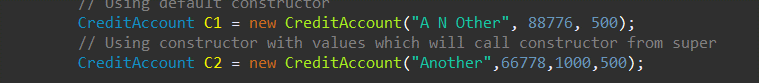


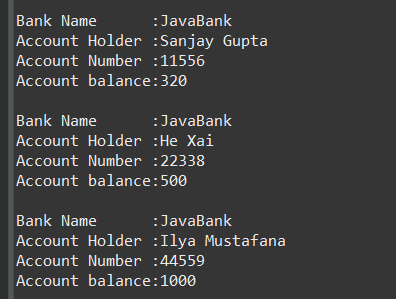
**Тестирование абстрактного класса 2**

1. Обновите класс TestCreditAccount, чтобы значения для учетных записей a2 и a3 предоставлялись через конструктор, а не через вызовы метода установки.

  
2. Обновите создание учетной записи c1, чтобы значения предоставлялись через конструктор, а не через вызовы метода установки.

Account больше не является суперклассом, поэтому обновите код создания экземпляра, чтобы создать объект creditAccount.





Примеры и конспект из книги Васильева

Конспект по главе 7: Абстрактные классы и интерфейсы

1. Абстрактные классы и методы

Абстрактный метод — метод, который объявлен, но не имеет реализации (тела). Описывается с ключевым словом abstract.

Абстрактный класс — класс, содержащий хотя бы один абстрактный метод. Также описывается с ключевым словом abstract.

На основе абстрактного класса нельзя создать объект.

Абстрактные классы используются как шаблоны для наследования. Подклассы должны реализовать все абстрактные методы, иначе они также будут абстрактными.

Объектная переменная абстрактного класса может ссылаться на объекты подклассов.

2. Интерфейсы

Интерфейс — похож на абстрактный класс, но содержит только объявления методов и статические константы.

Методы в интерфейсе по умолчанию public.

Интерфейсы реализуются в классах с помощью ключевого слова implements.

Класс может реализовывать несколько интерфейсов.

В Java 8 появилась возможность добавлять методы с реализацией по умолчанию (с ключевым словом default).

3. Реализация интерфейсов

Класс, реализующий интерфейс, должен реализовать все его методы (если они не имеют реализации по умолчанию).

Если метод в интерфейсе имеет реализацию по умолчанию, класс может не переопределять его.

4. Интерфейсные переменные

Интерфейсная переменная может ссылаться на объект любого класса, реализующего данный интерфейс.

Доступны только методы, объявленные в интерфейсе.

5. Методы с кодом по умолчанию

В Java 8 интерфейсы могут содержать методы с реализацией по умолчанию (с ключевым словом default).

Если класс не переопределяет такой метод, используется реализация по умолчанию из интерфейса.

6. Расширение интерфейсов

Интерфейсы могут наследовать другие интерфейсы с помощью ключевого слова extends.

Интерфейс-наследник получает все методы и константы из родительского интерфейса.

7. Наследование классов и реализация интерфейсов

Класс может наследовать суперкласс и одновременно реализовывать несколько интерфейсов.

Если методы с одинаковой сигнатурой есть в суперклассе и интерфейсах, их нужно явно переопределить в подклассе.

8. Резюме

Абстрактные классы используются как шаблоны для наследования. Они могут содержать абстрактные методы, которые должны быть реализованы в подклассах.

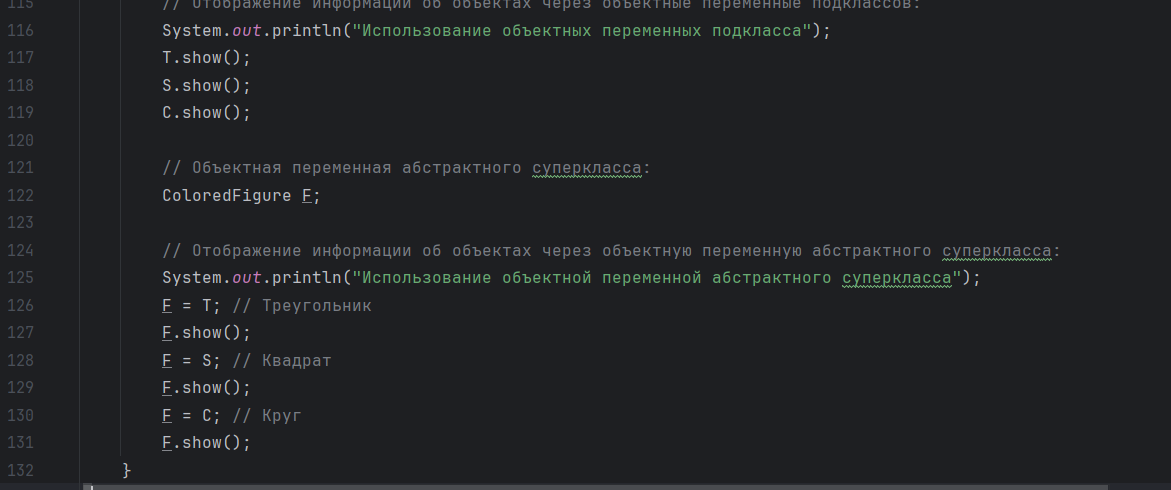
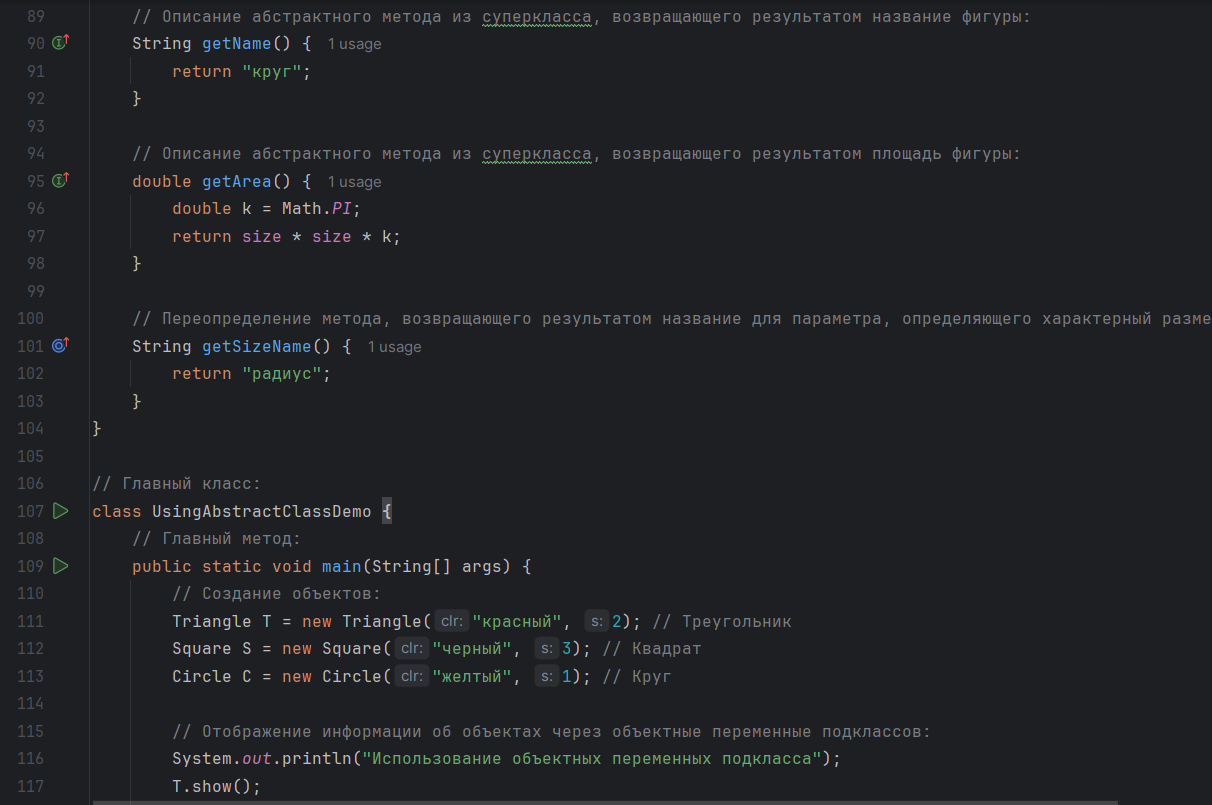
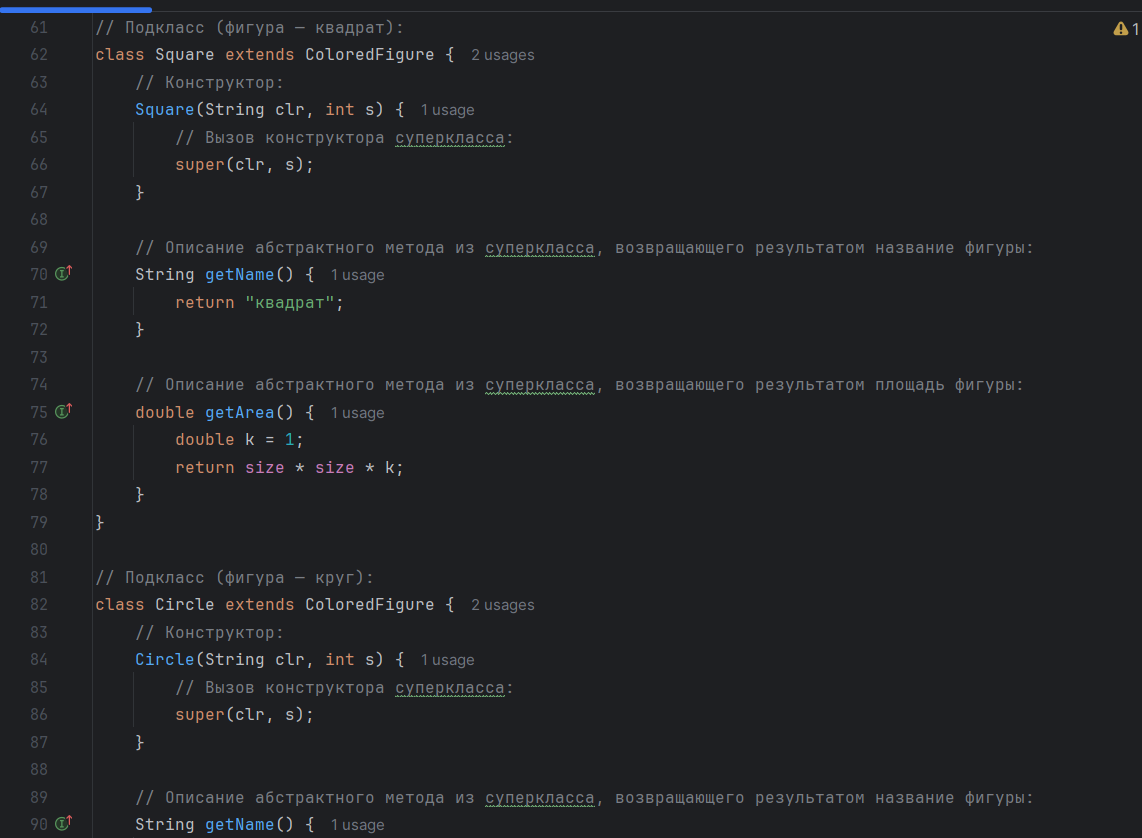
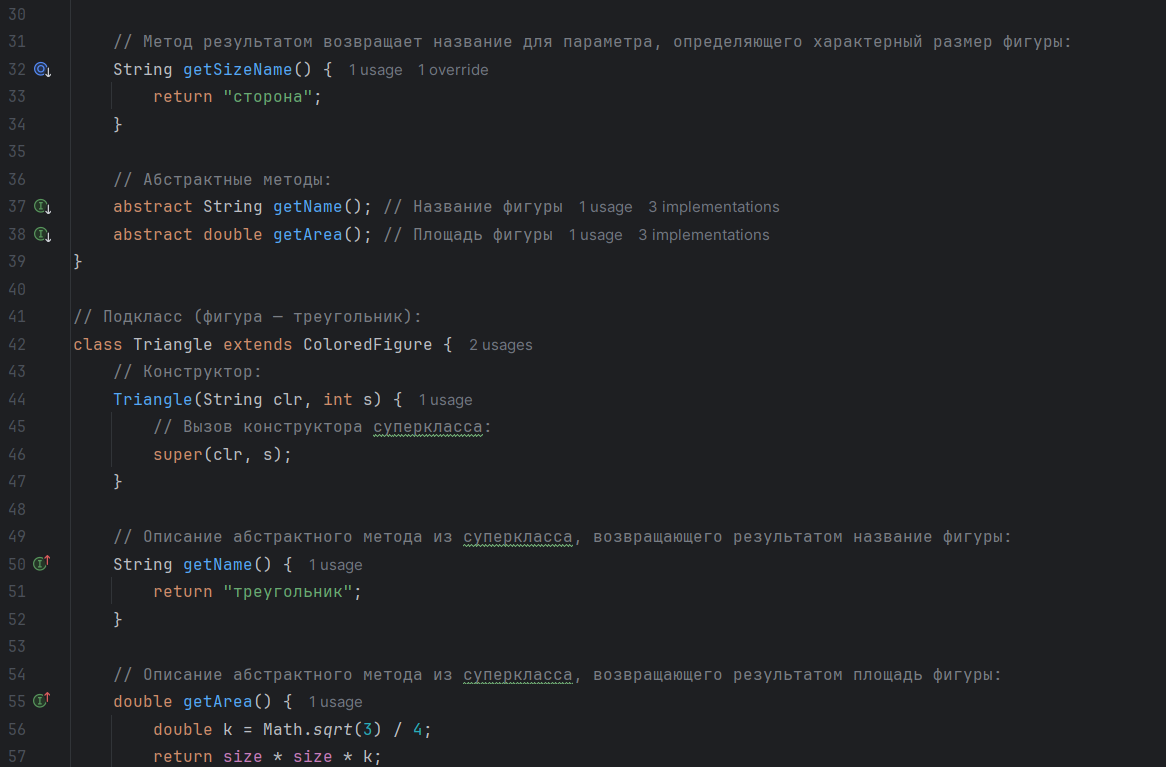
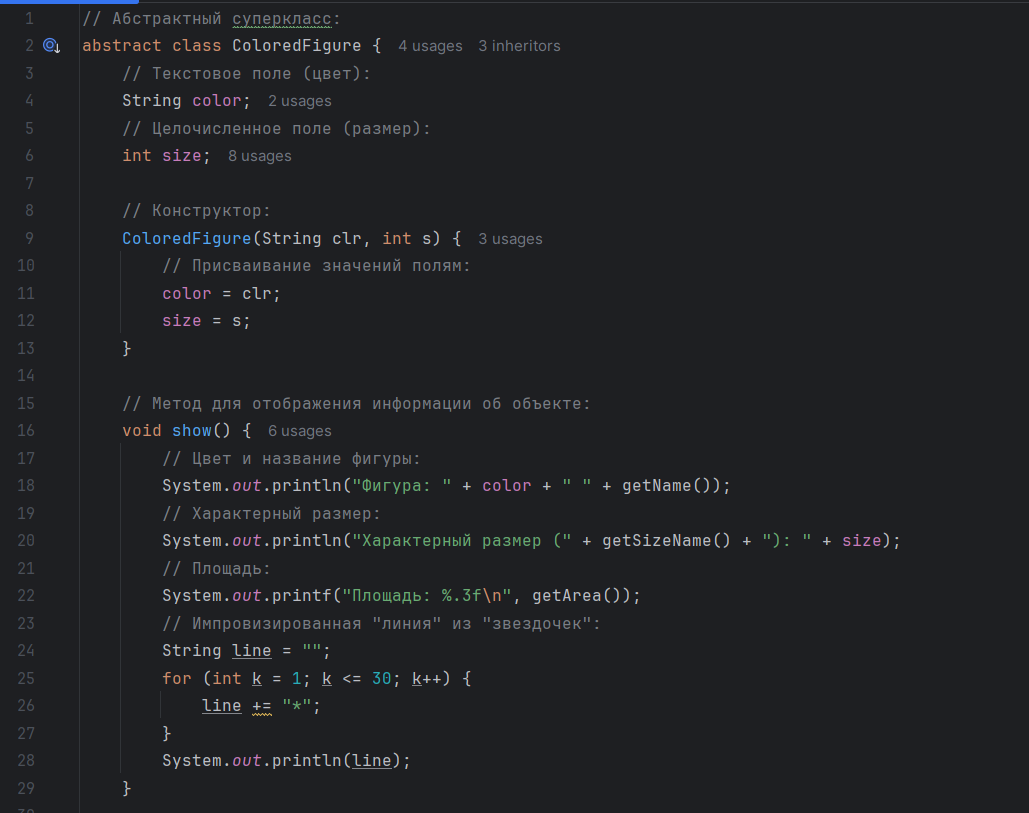
Интерфейсы — это контракты, которые классы должны выполнять. Они могут содержать методы с реализацией по умолчанию.

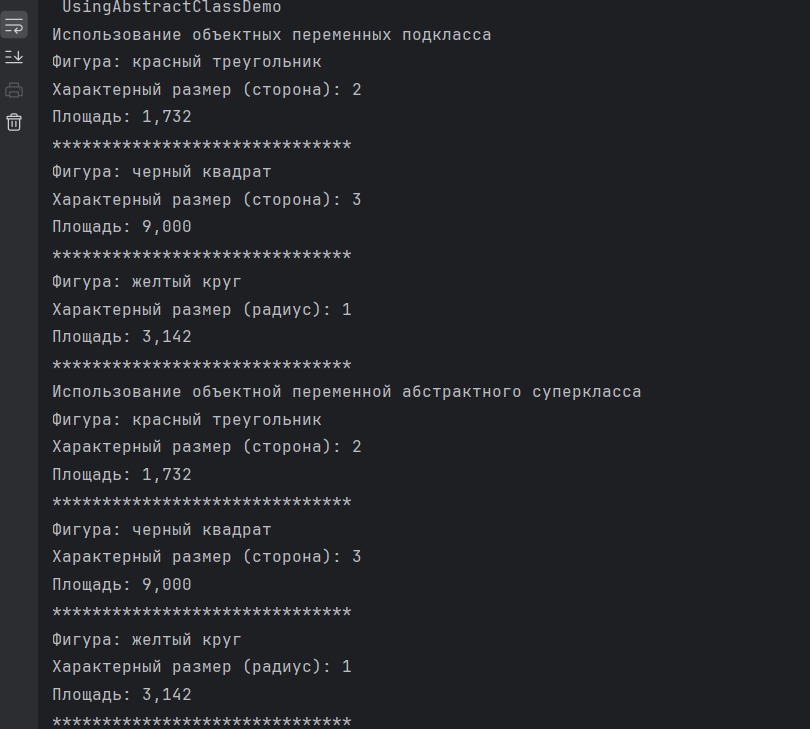
Класс может реализовывать несколько интерфейсов, но наследовать только один класс.

Интерфейсы могут наследовать другие интерфейсы, что позволяет создавать иерархии интерфейсов.

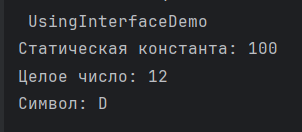
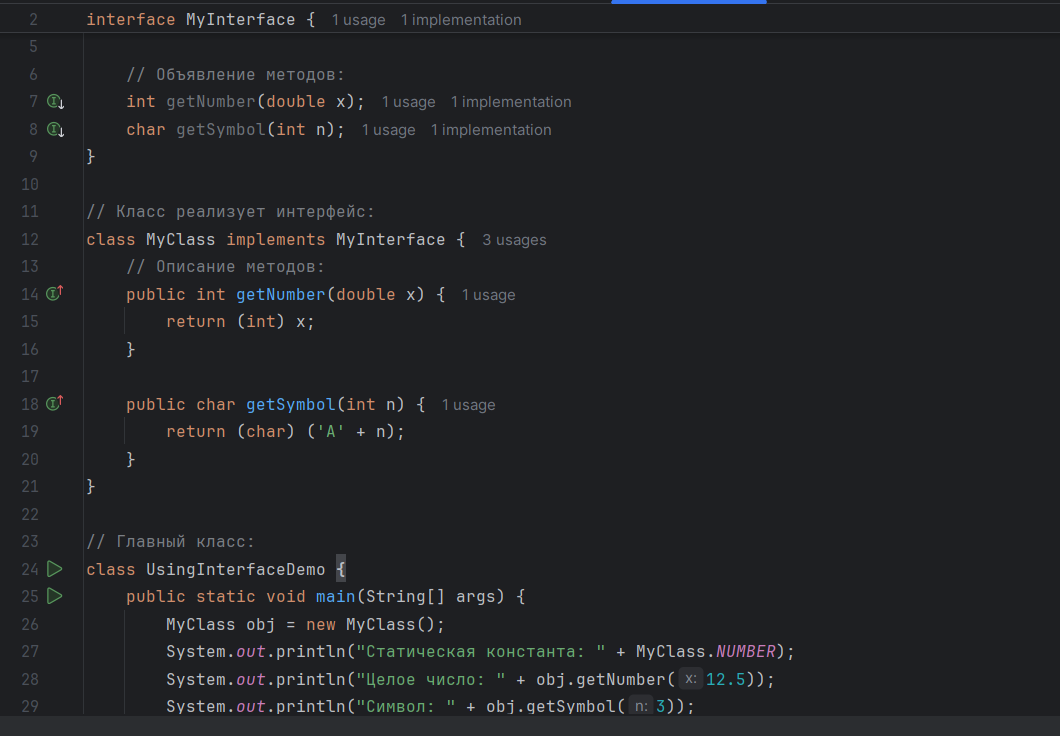
Методы с реализацией по умолчанию в интерфейсах позволяют добавлять новую функциональность без нарушения существующего кода.

Пример 1

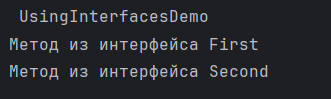
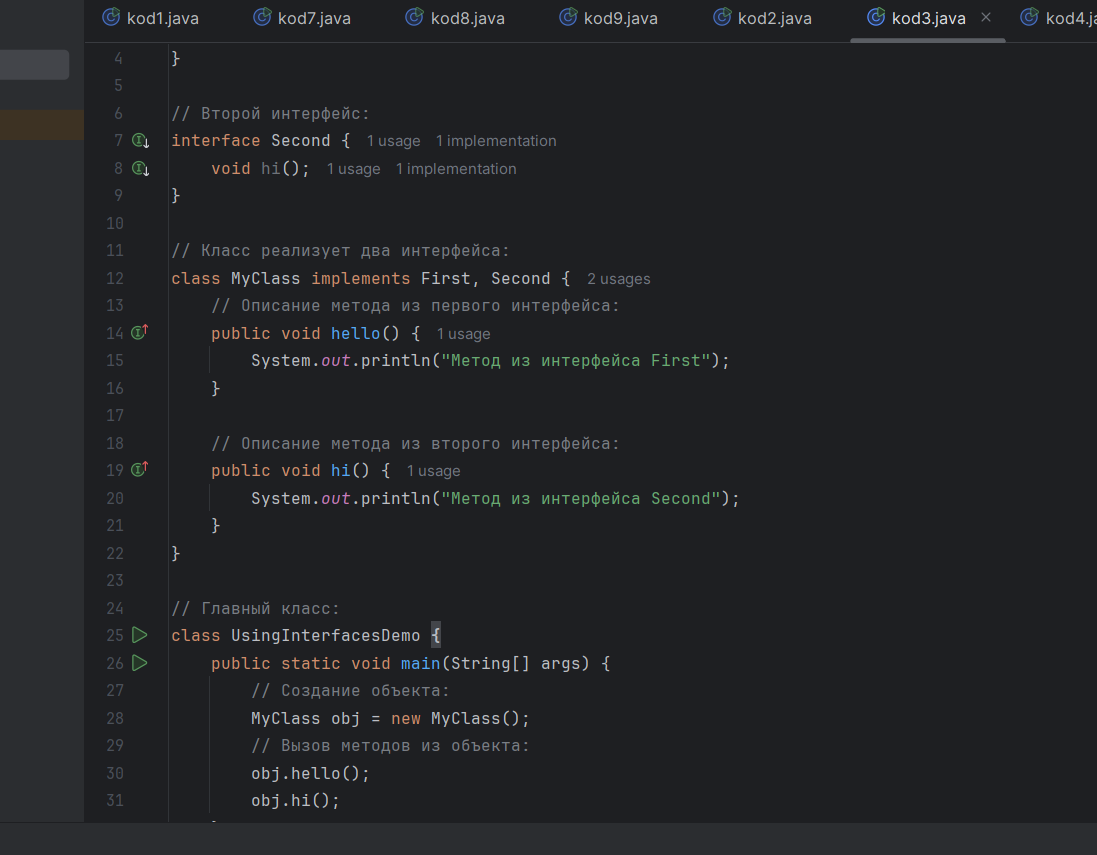




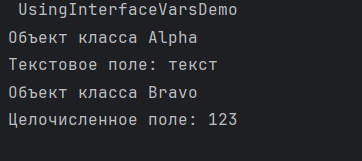
Пример 2



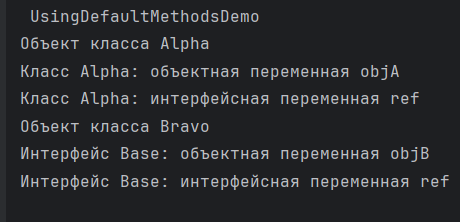
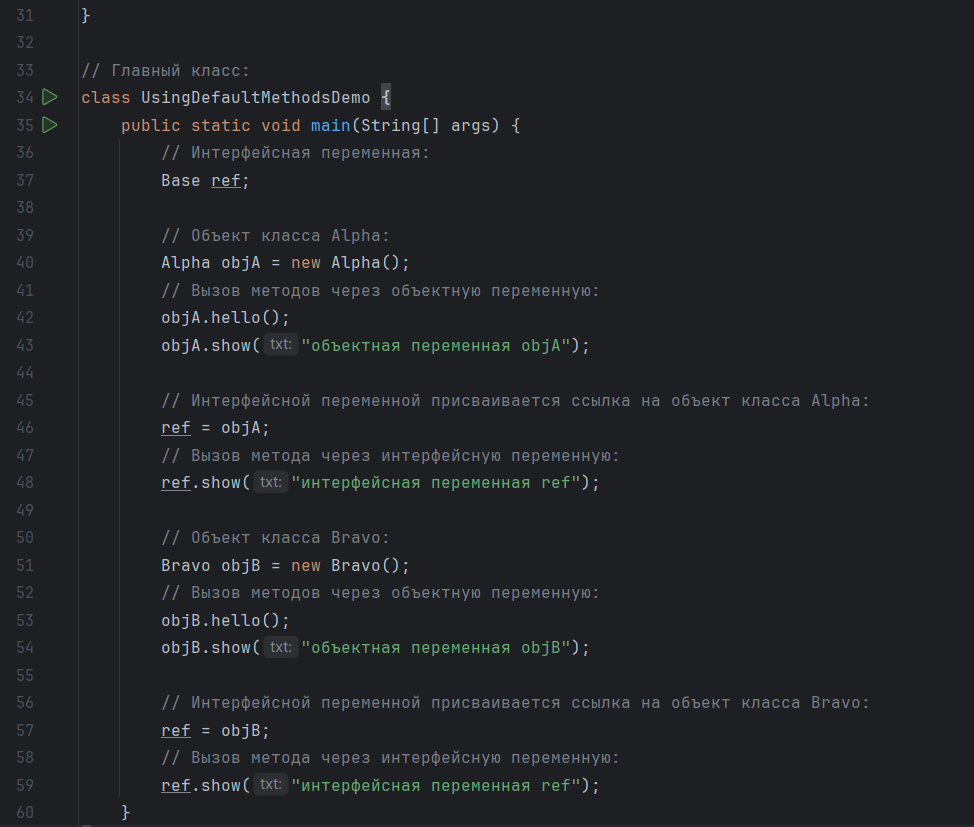
Пример 3



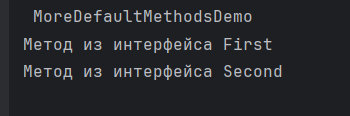
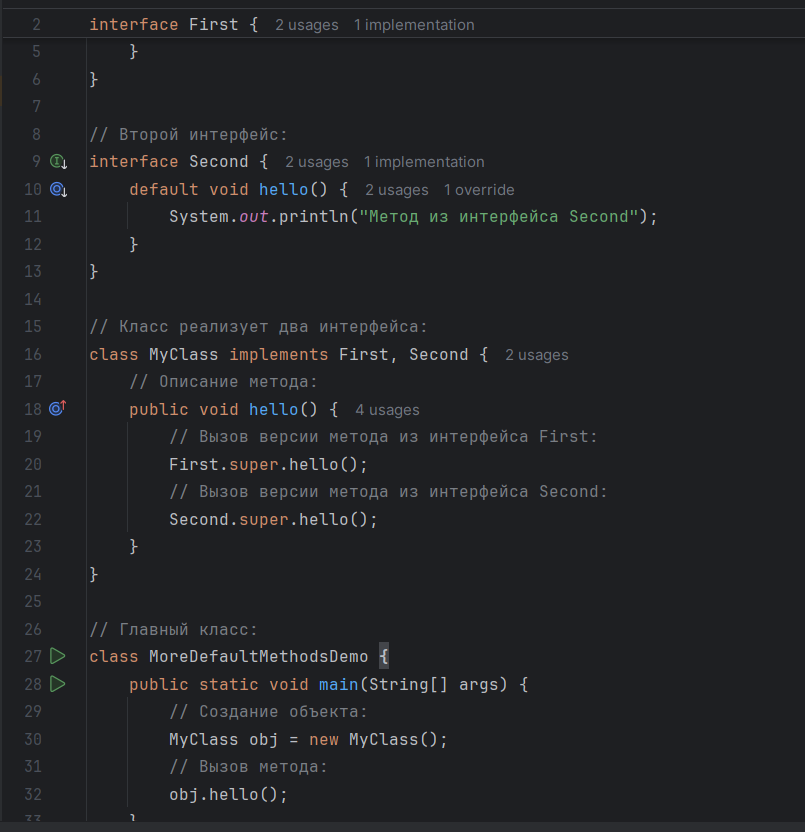
Пример 4



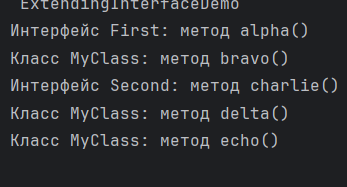
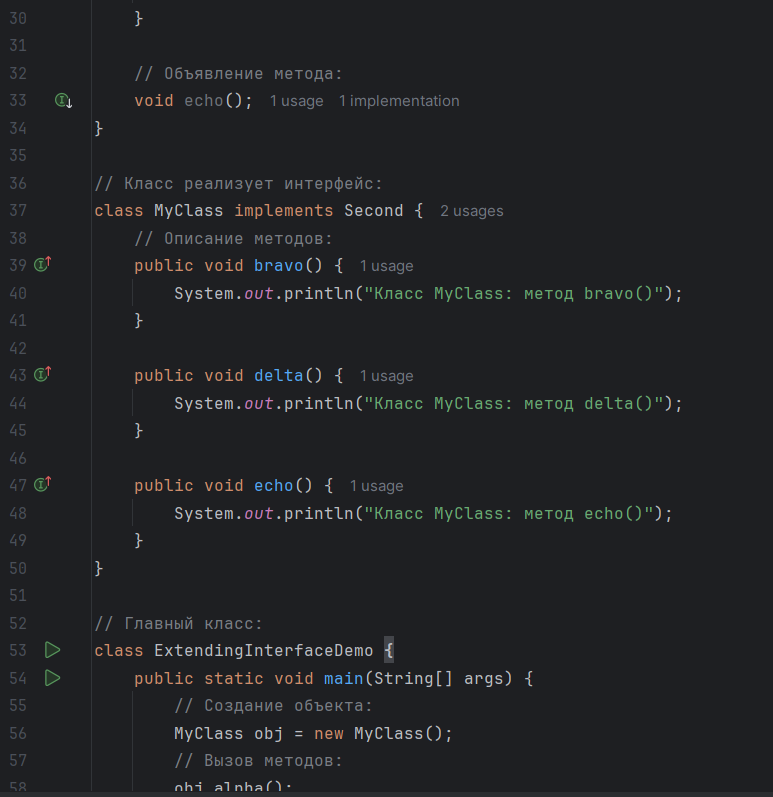
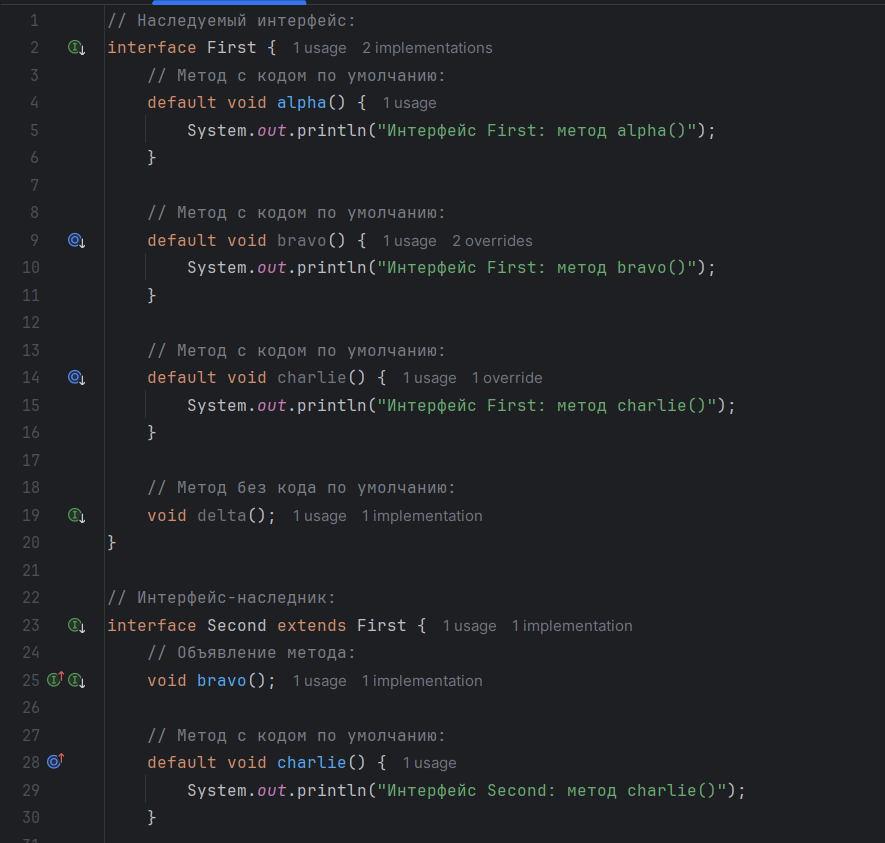
Пример 5



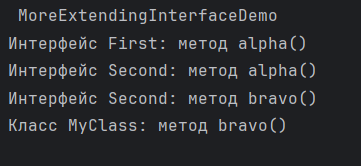
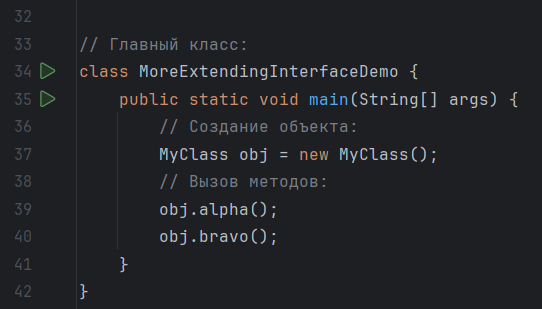
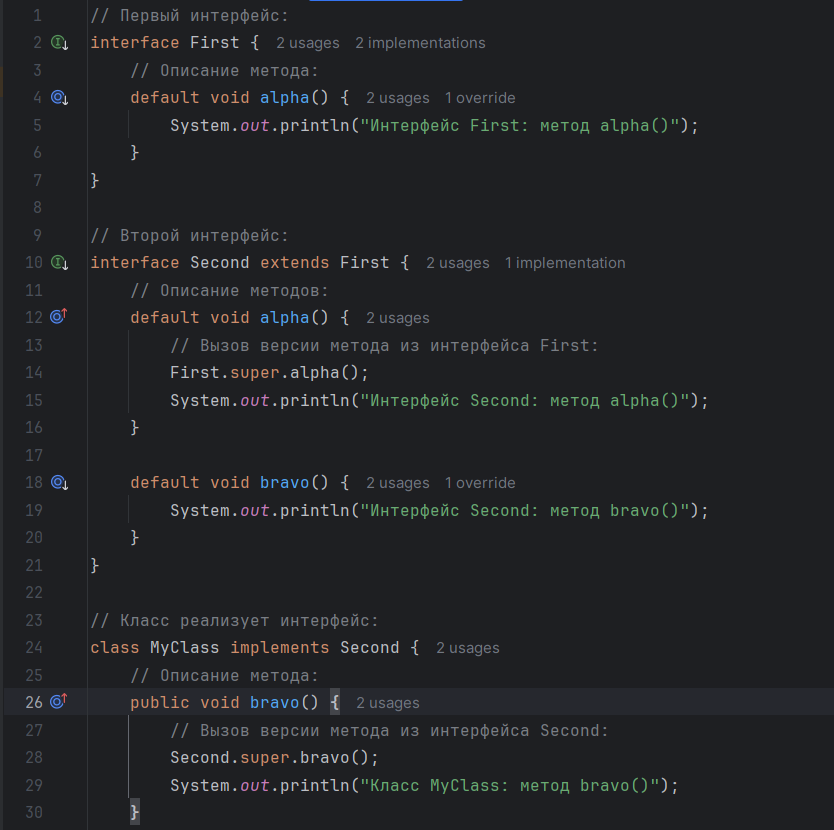
Пример 6



Пример 7



Пример 8



Пример 9