Санкт-Петербургский политехнический университет Петра Великого

Институт компьютерных наук и кибербезопасности

Высшая школа «Компьютерных технологий и информационных систем»

**ОТЧЕТ по лабораторной работе №4**

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

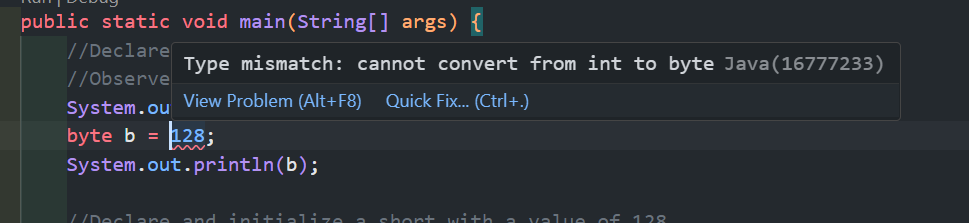
«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_ 2024г.

Санкт-Петербург, 2024

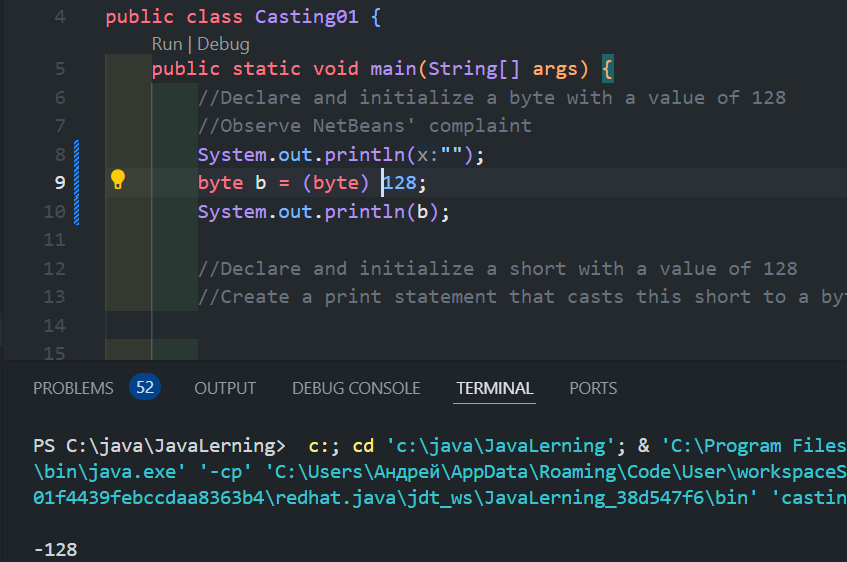
**Задание 1 (по уроку 3.4)**

1. **Casting01.**

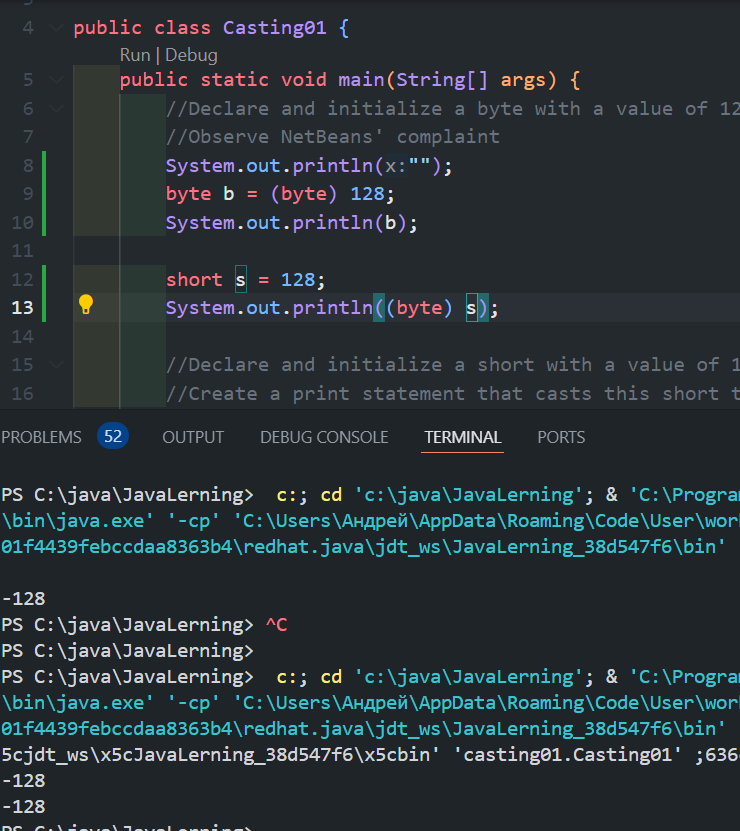
Обьявляем переменную b с типом byte и задаем ей значение 128. Так как тип byte может хранить числа от -128 до 127 компилятор выдает ошибку:



Если мы попробуем явно преобразовать число 128 к byte, программа заработает, но тогда в переменной будет храниться значение -128.

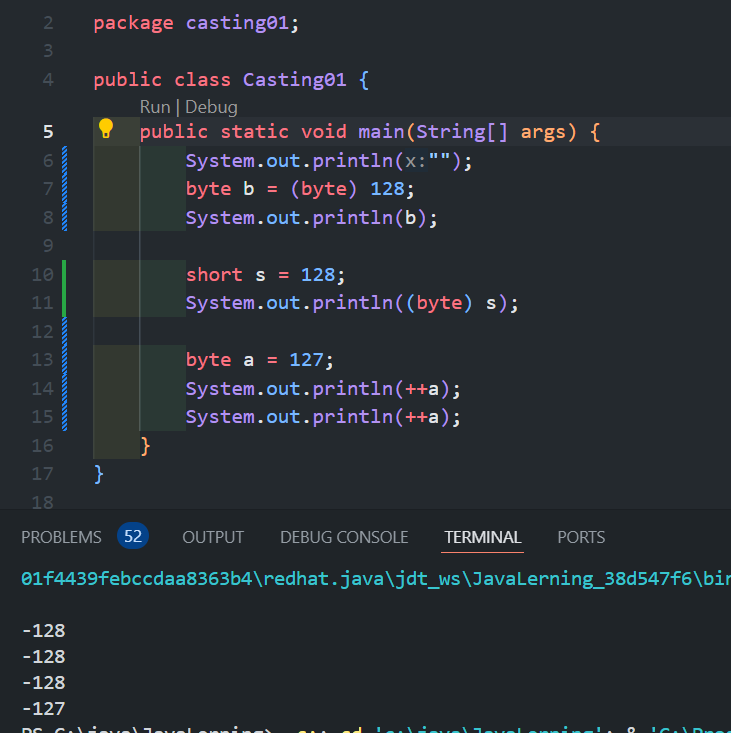


Объявите и инициализируйте переменную short со значением 128: Создайте оператор вывода, преобразующий short в byte.



Мы можем наблюдать, что программа запускается, но корректно вывести число 128 не получается, происходит переполнение.

Объявите и инициализируйте переменную byte со значением 127. Прибавьте 1 и выведите результат. Можно увидеть что произошло переполнение, и переменная стала равна -128. Ещё раз прибавьте 1 и выведите. Это привело к тому что переменная стала равна -127.



Если мы в short зададим значение 259, то соответствующий принт выведет 3. Это происходит, потому что мы преобразовываем short к int и у нас происходит переполнение памяти 259-256=3

1. **Casting02.**

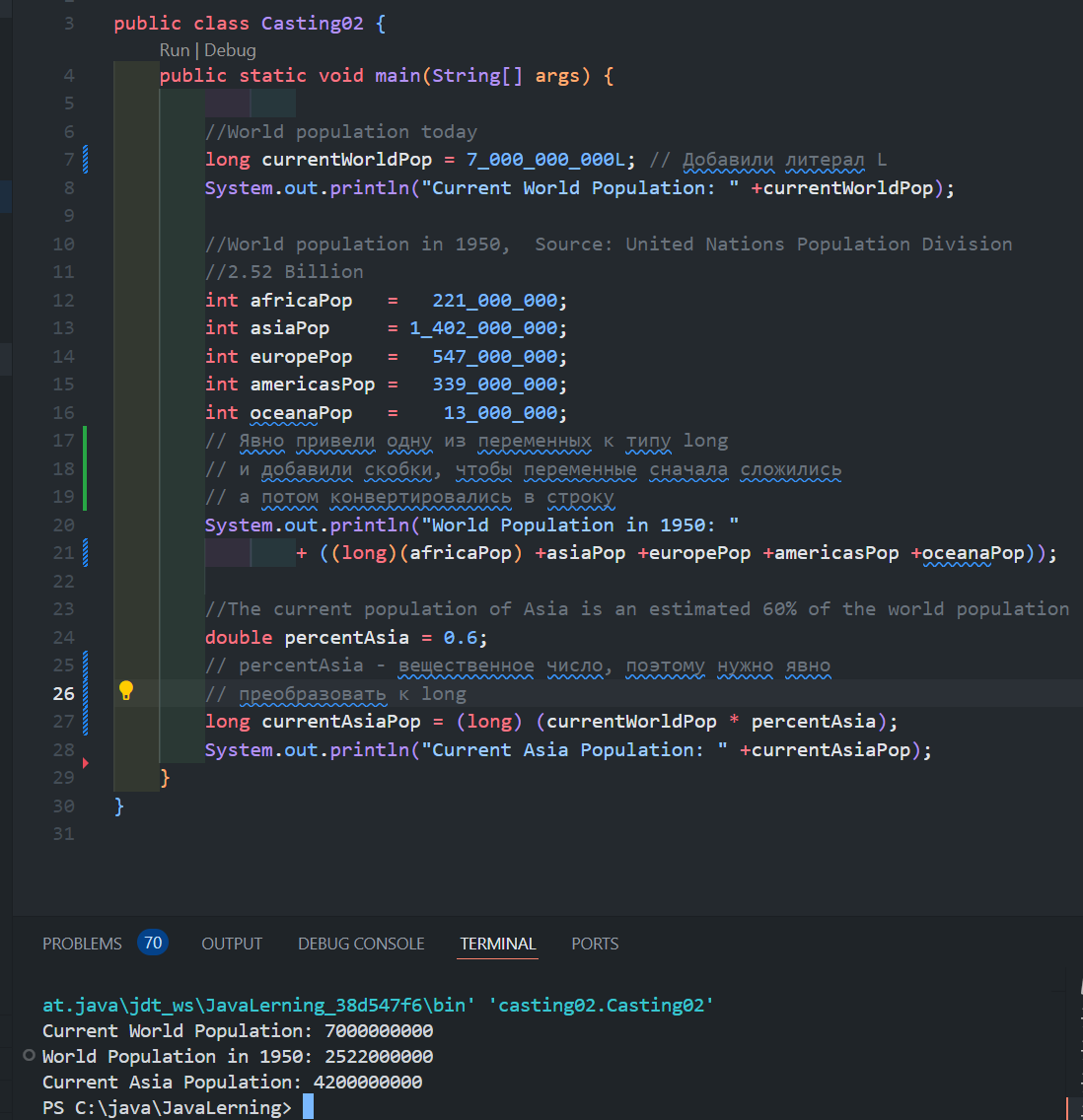
Импортируйте и откройте проект Casting02. В этой программе есть несколько ошибок. Эти ошибки можно устранить, применив следующие знания

– Знание типов данных

– Знание принципа повышения

– Знание принципа преобразования

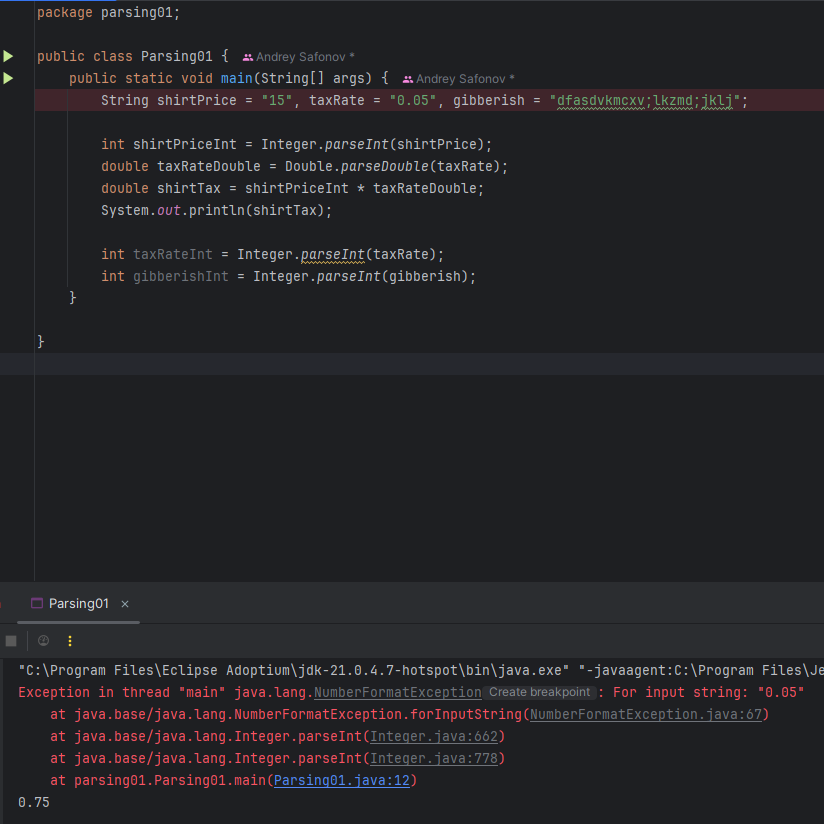
Исправленная версия программы



1. **Parsing01.**

* Импортируйте и откройте проект Parsing01.
* Объявите и инициализируйте 3 объекта String со предоставленными данными.
* Выполните синтаксический анализ и умножьте shirtPrice\*taxRate, чтобы получить значение налога.
* Выведите это значение.
* Попробуйте проанализировать taxRate как int.
* Прочтите сообщение об ошибке.
* Попробуйте проанализировать gibberish как int.

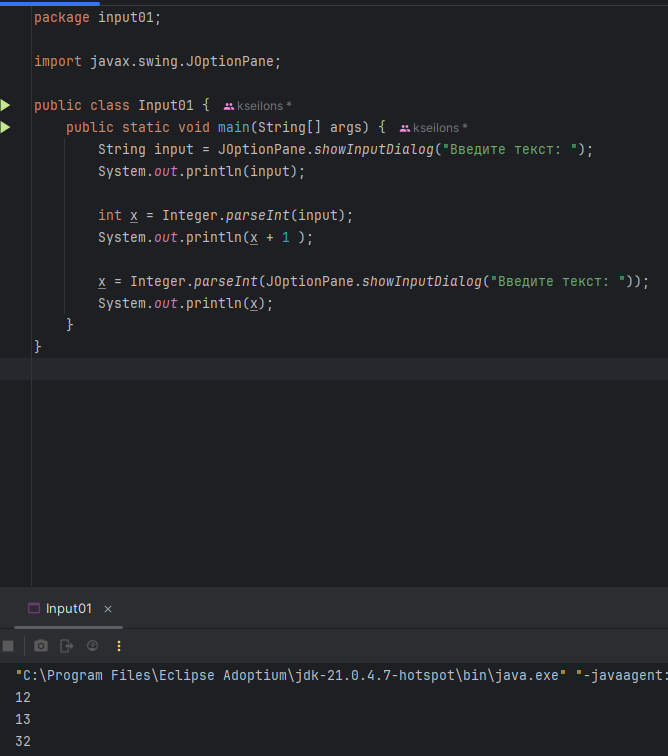
Прочтите сообщение об ошибке.



При попытке проанализировать taxRate как int и gibberish как int возникает ошибка «NumberFormatException» (Исключение числового формата) потому, что эти строки не имеют правильного формата для этих преобразований.

1. **Задания по уроку 3.5.**
2. **Input01.**

* Импортируйте и откройте проект Input01.
* Создайте JOptionPane:
* В NetBeans отобразится предупреждение.
* Следуя рекомендации NetBeans, импортируйте javax.swing.JOptionPane
* Сохраните введенные данные как объект String.
* Выведите переменную String.
* Выполните синтаксический анализ String как отдельной переменной int.
* Потребуется ввести значение, доступное для синтаксического анализа.
* Выведите значение +1.
* Попробуйте создать диалоговое окно, проанализировать его и инициализировать int в одной строке. Используйте только одну точку с запятой (;).

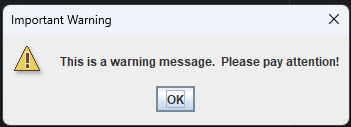


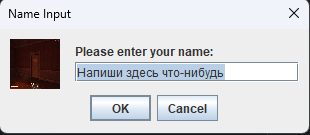
1. **Input02.**

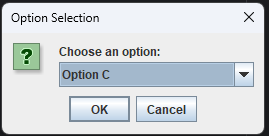
* Импортируйте и откройте проект Input02.
* Поэкспериментируйте с кодом и попробуйте изменить следующее …
* Заголовок сообщения
* Сообщение
* Любой введенный текст по умолчанию
* Значок диалогового окна
* Анализируйте, выполняйте операции и выводите входные данные.

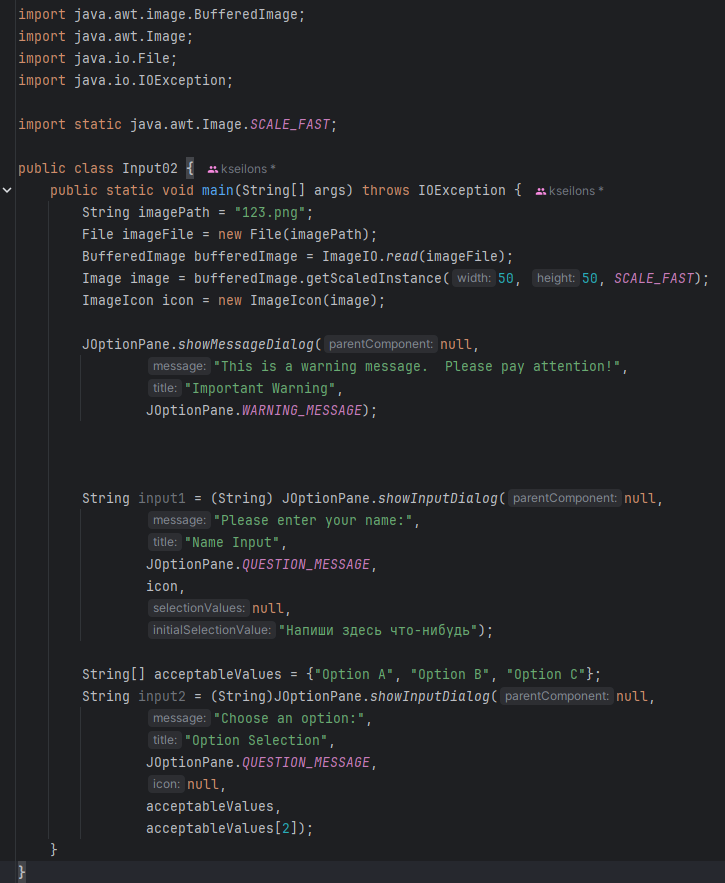
При изменении цифры от 0 до 3 меняется значок: 0 – красный крест, 1 – фиолетовый круг, 2 – желтый треугольник с восклицательным знаком, 3 – зеленый квадрат с вопросительным знаком.

Также можно менять сообщение(первые кавычки), заголовок(вторые кавычки), введенный текст(третьи кавычки) и изменять значения выбора:



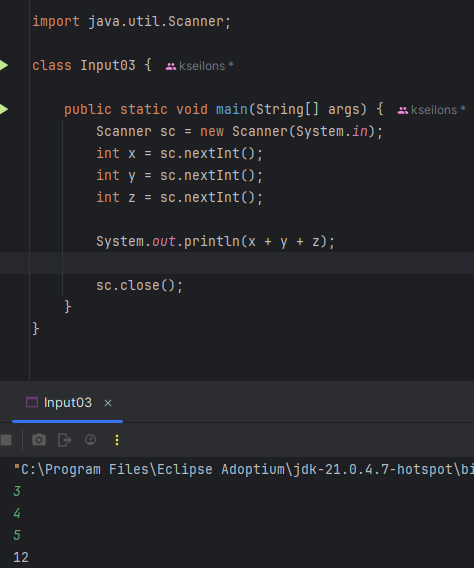




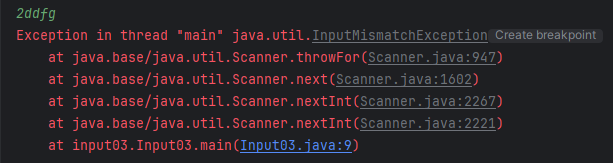


1. **Input03.**

* Импортируйте и откройте проект Input03.
* Создайте класс Scanner:
* В NetBeans отобразится предупреждение.
* Следуя рекомендации NetBeans, импортируйте java.util.Scanner
* Не забудьте закрыть Scanner.
* С помощью класса Scanner и System.in напишите программу, которая будет выполнять следующие задачи …
* Вычислять и выводить сумму трех целых чисел, введенных пользователем.
* Попытайтесь использовать менее трех маркеров.
* Попытайтесь использовать маркер, который нельзя проанализировать как int.



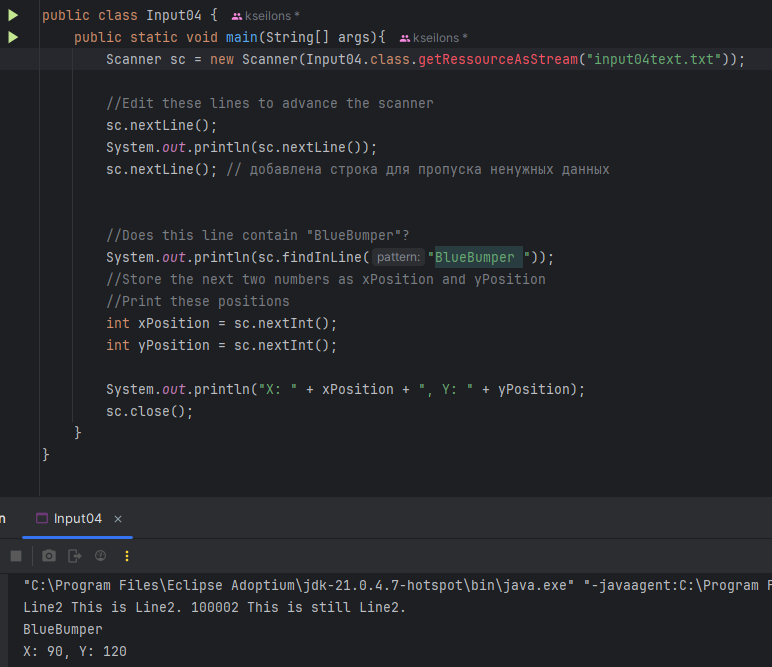
При попытке ввести маркер, который нельзя проанализировать как int, программа прекратила работу и вывела ошибку:



1. **Input04.**

**Часть 1.**

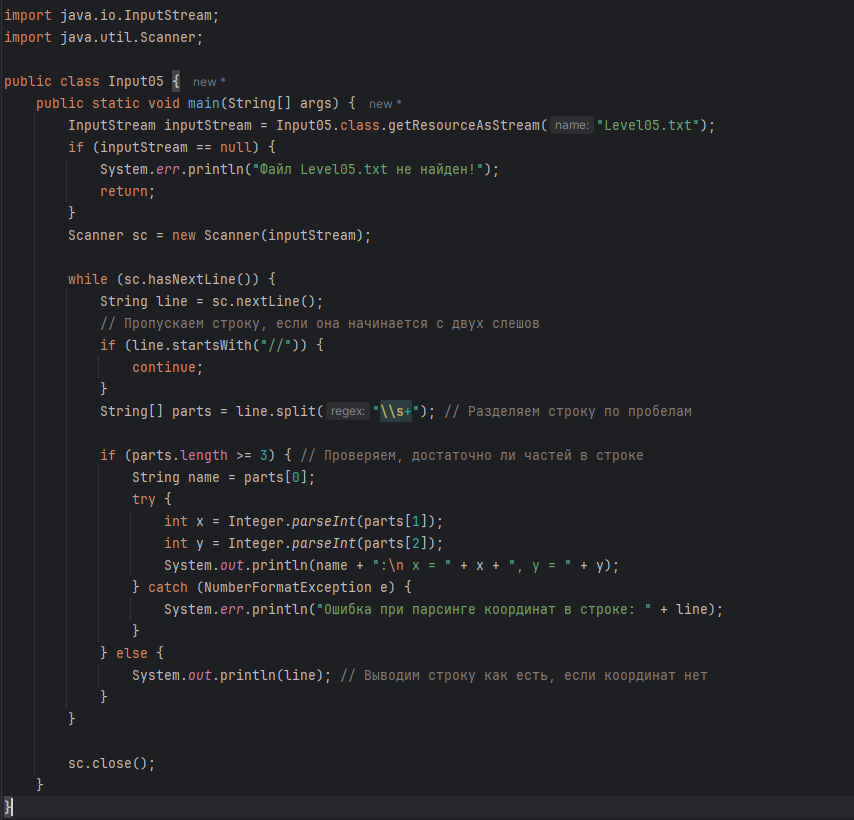
* Импортируйте и откройте проект Input04.
* Выполните код и изучите результат.
* Пролистайте все строки, пока не дойдете до "BlueBumper".
* Два числовых значения после "BlueBumper" — это объекты xPositon и yPosition. Сохраните эти координаты как значения integer и выведите их.
* При необходимости изучите файл input04text.txt.



**Часть 2.**

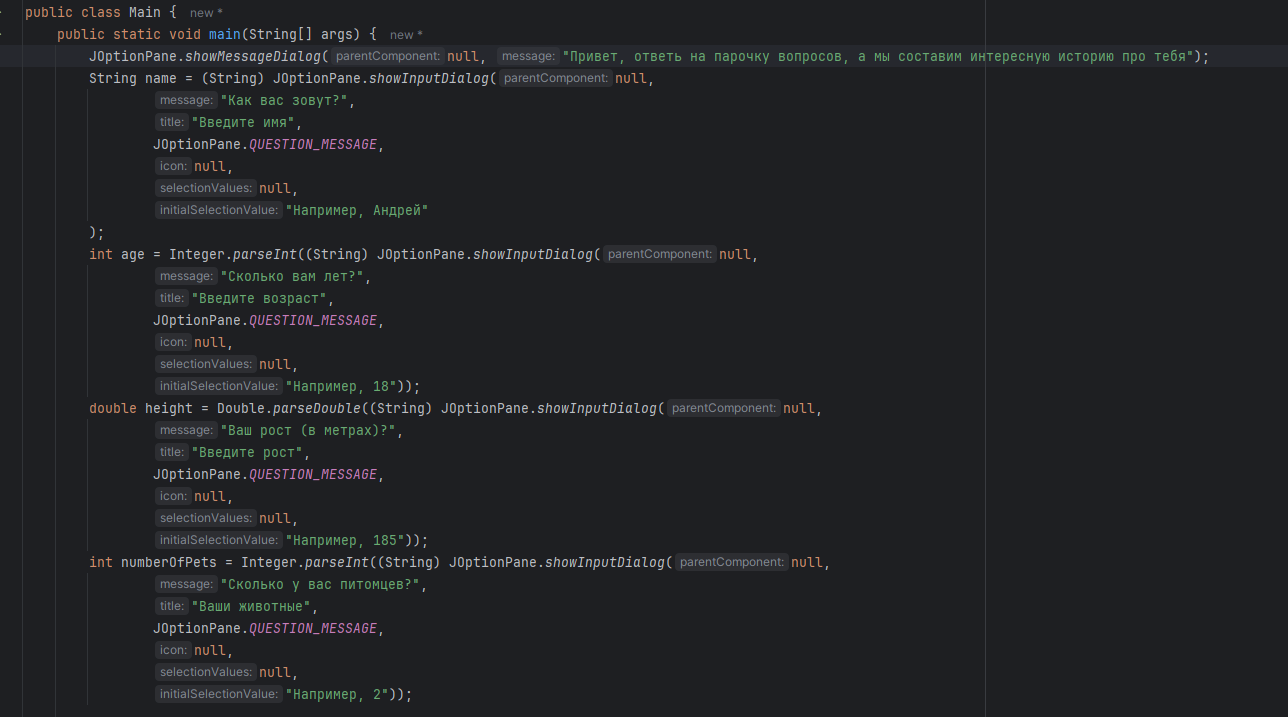
Если вам интересно, изучите файл Level05.txt:

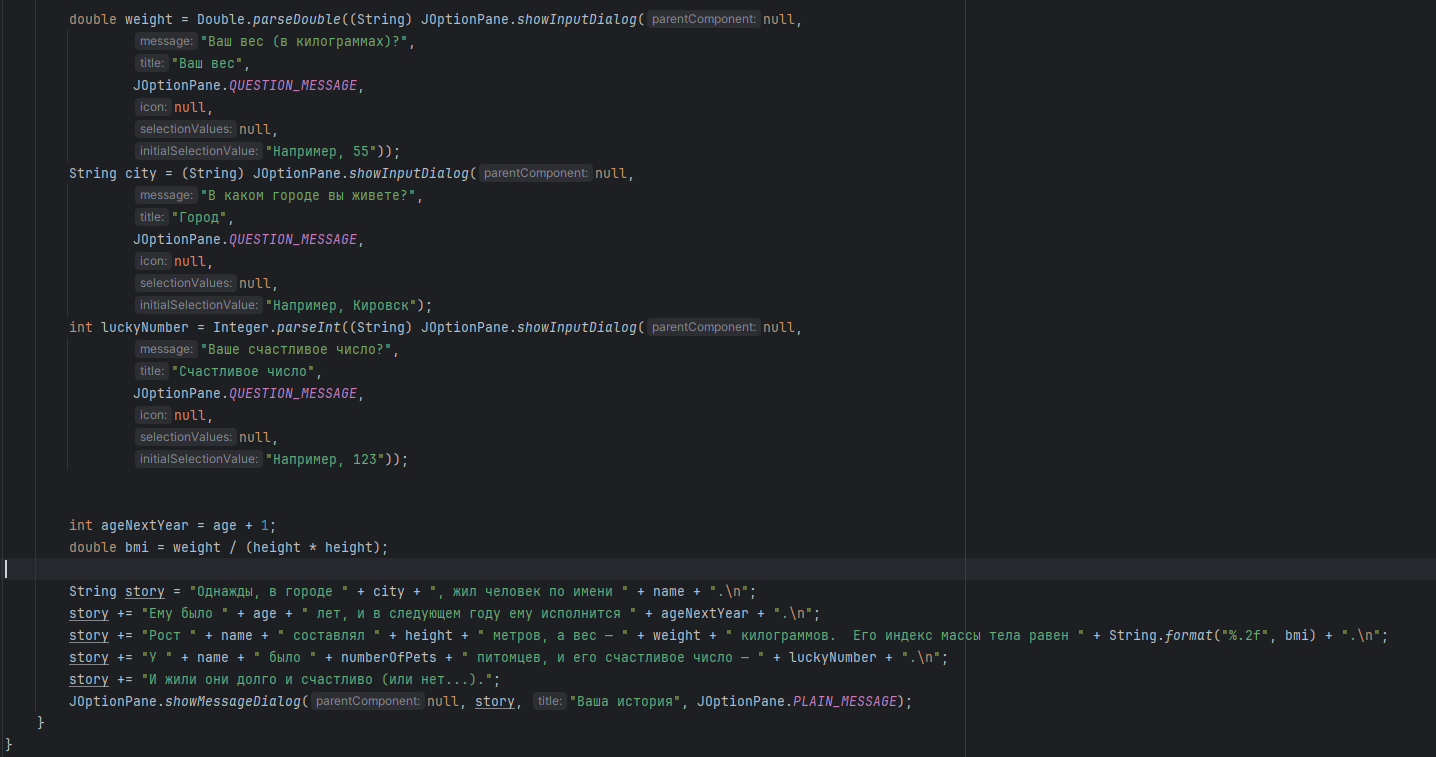
* В этом файле хранятся данные уровней игры Java Puzzle Ball.
* Операции чтения и анализа данных уровней немного сложнее, чем в этом упражнении.
* Но если вы выполнили это упражнение, вы приблизились к пониманию того, как это происходит.



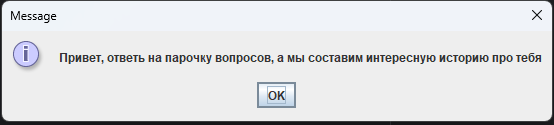
1. **Упражнения – Раздел 3.**

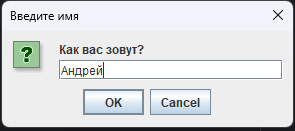
Цель — создать программу, подобную JavaLibs. Написать рассказ, отдельные части конечного текста которого изменяются в зависимости от ввода пользователя. Предложить пользователю различные варианты ввода.

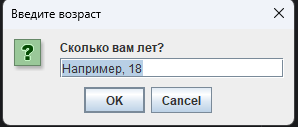


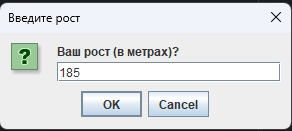


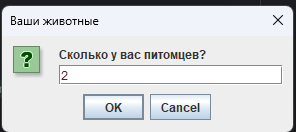
Работа программы:

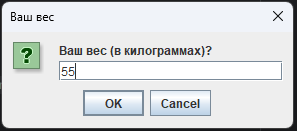


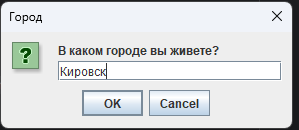


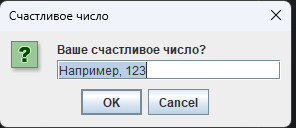




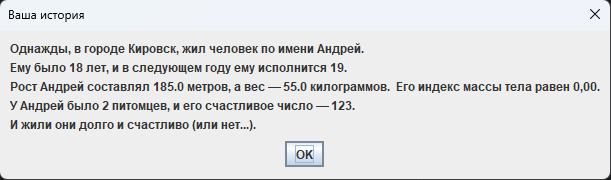






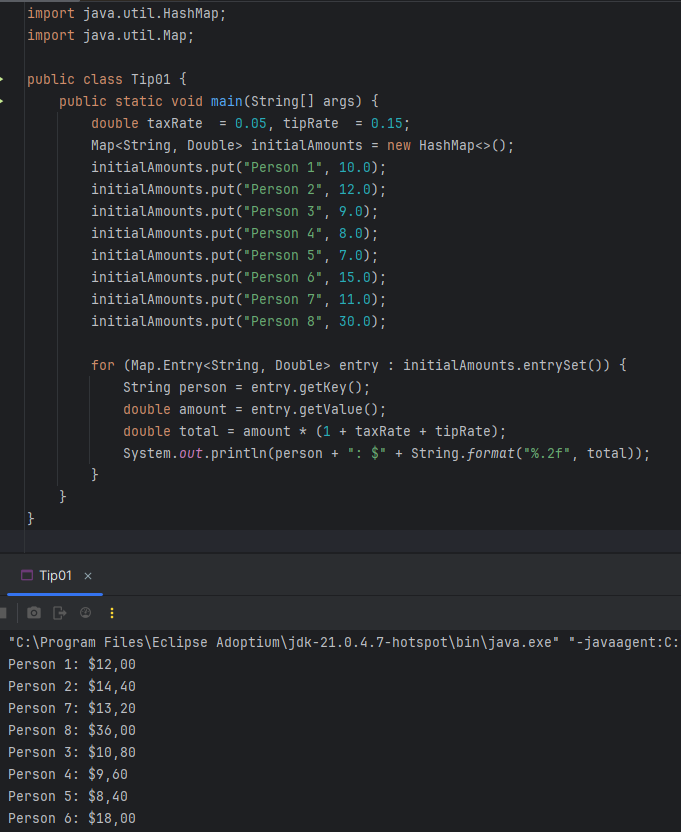


Вывод:



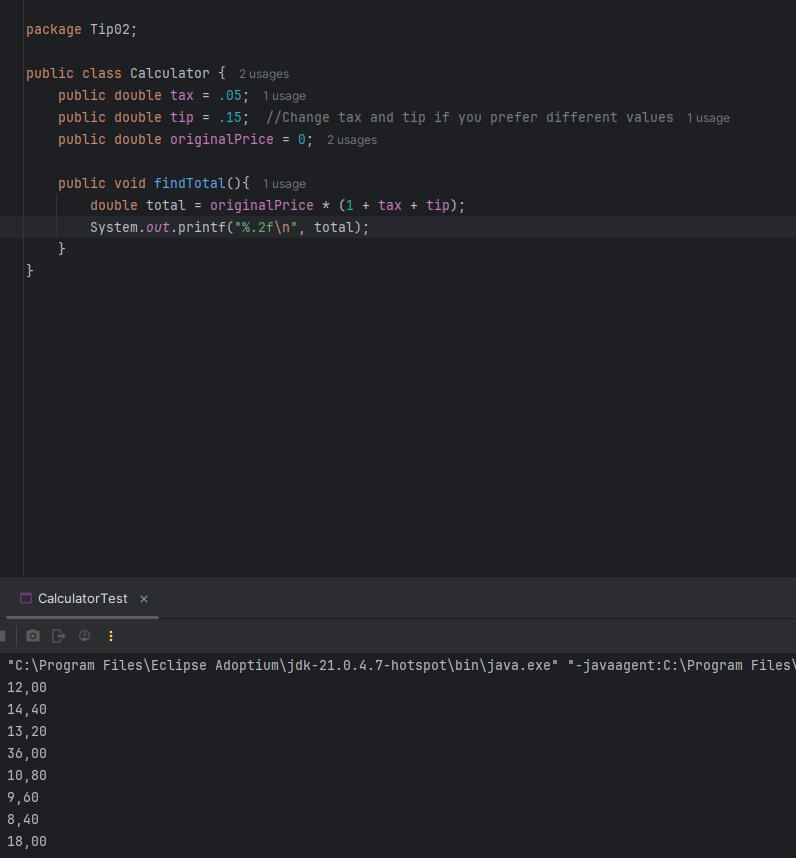
1. **Задания по уроку 4.1**.
2. **Tip01.**

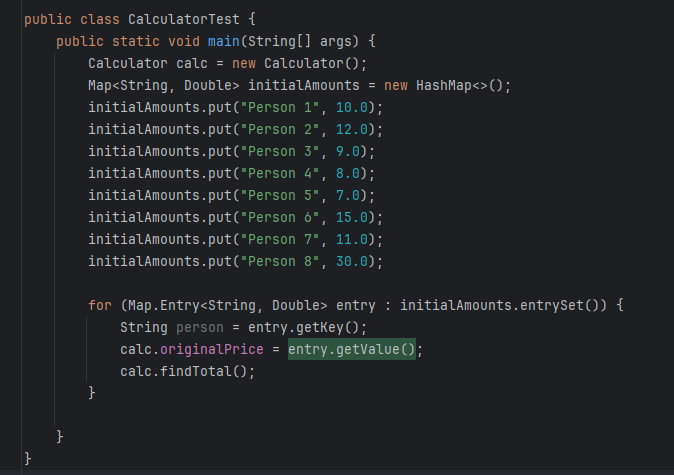
У Алекса день рождения. Вы собрались группой из восьми человек в местном ресторане, чтобы отметить праздник. Когда вам принесли счет, никто толком не знал, сколько должен заплатить. Вам известна только общая сумма без учета налога (5%) и чаевых (15%). Но вам повезло! Вы взяли с собой ноутбук, и вас попросили написать программу, которая рассчитает общую сумму для каждого человека.



1. **Tip02.**

* Импортируйте и откройте проект Tip02.
* Дополните метод findTotal(), который должен выполнять следующие функции:
* Вычислять итоговое значение с учетом полей tax, tip и originalPrice
* Выводить итог для каждого человека
* В главном методе:
* Создайте экземпляр объекта Calculator с именем calc.
* Посмотрите, что произойдет в NetBeans, если ввести слово "calc".
* Получите доступ к полям и методам этого объекта, чтобы вывести итог для каждого гостя на дне рождения.
* Измените tip и tax, если предпочитаете использовать другие значения.



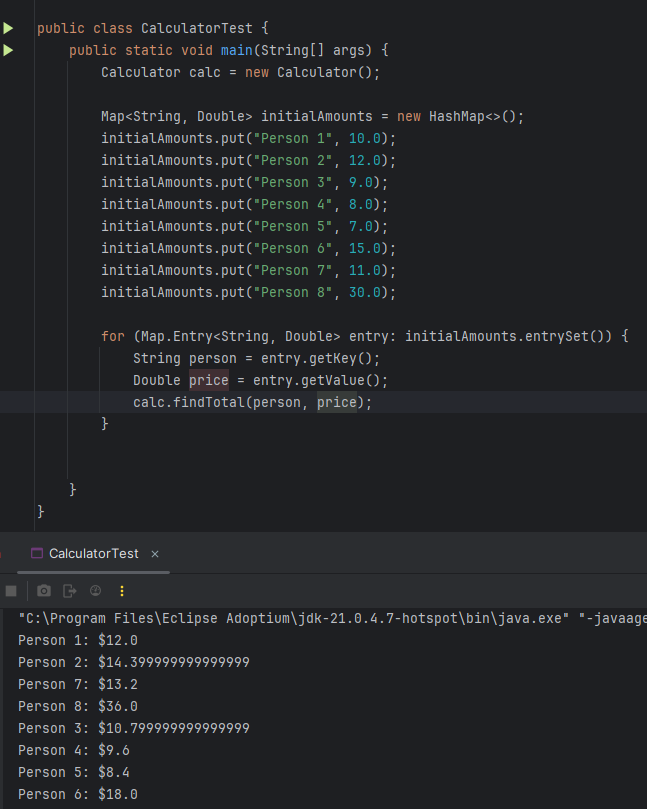


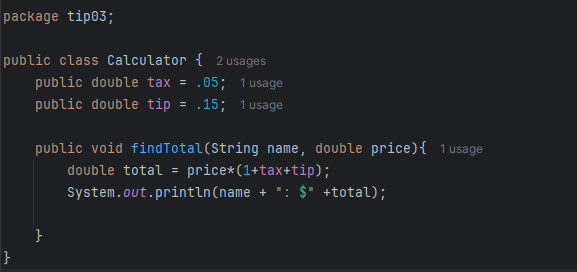
1. **Tip03.**

* Импортируйте и откройте проект Tip03.
* В главном методе:
* Используйте экземпляр объекта Calculator и передайте аргументы в метод findTotal(), чтобы вывести итог для каждого человека.
* Подсказка. Изучите метод findTotal() в классе Calculator и выясните, сколько аргументов в него можно передать.

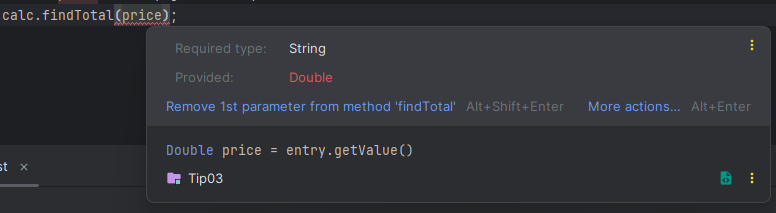
К кому относится каждое итоговое значение?

* Измените метод findTotal(), чтобы передать в него дополнительный аргумент String name. Объедините оператор вывода, чтобы включить аргумент name.
* В NetBeans появится сообщение об ошибке в главном методе. Проверьте вызовы метода findTotal().



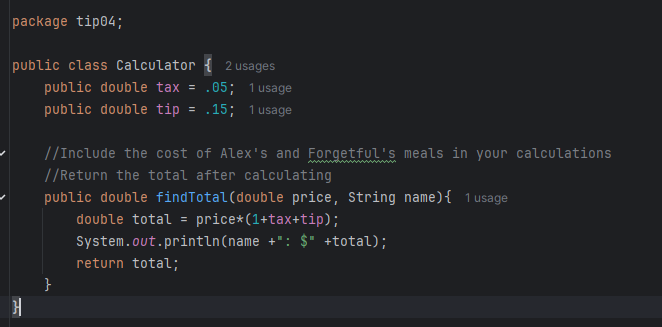


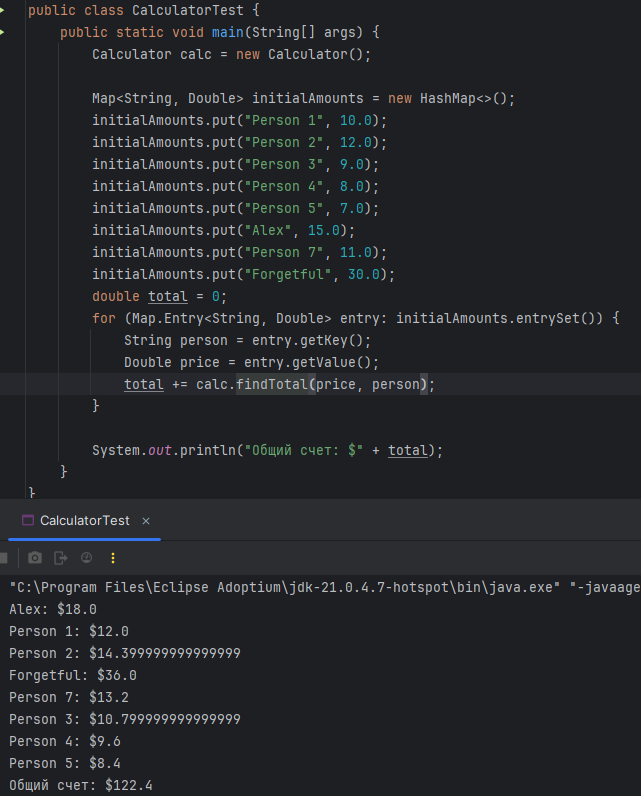
Появятся ошибки, т.к. мы подаем один аргумент вместо двух.



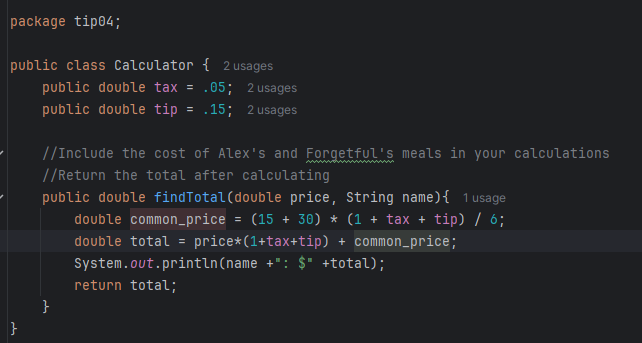
1. **Tip04.**

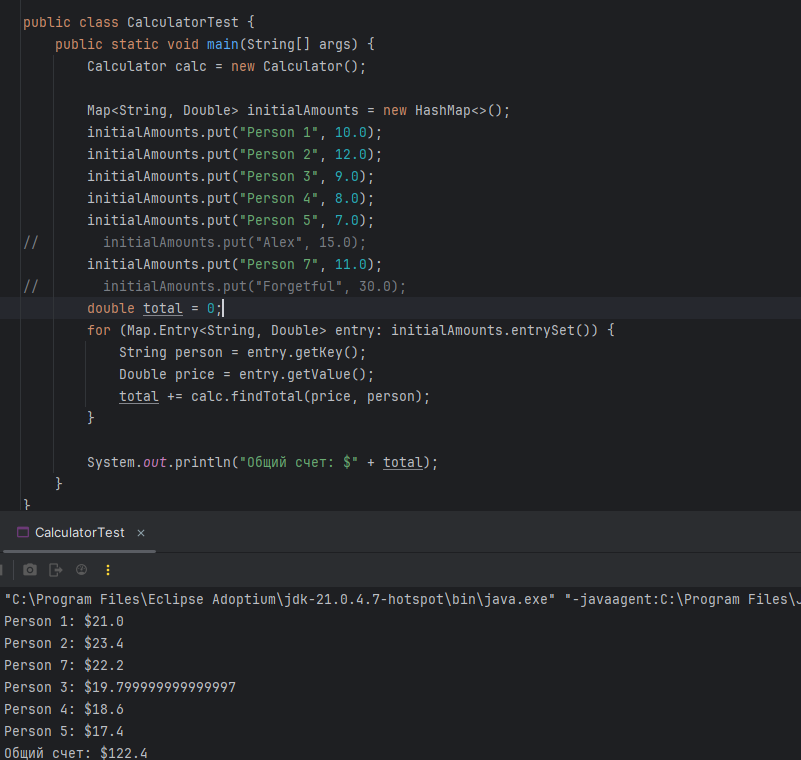
* Отредактируйте решение Tip03 или импортируйте проект Tip04.
* Вычислите и выведите итоговое значение счета для всех гостей с учетом налога и чаевых.
* Для этого потребуется отредактировать метод findTotal(), чтобы он возвращал вычисленное значение.
* Гость 8 забыл бумажник. А ужин для Алекса был подарком на его день рождения. Измените метод findTotal(), чтобы разделить стоимость остальных блюд поровну между остальными гостями.
* Пересчитайте итог для всех гостей. Это значение не должно измениться.





Выполним вторую часть задания:



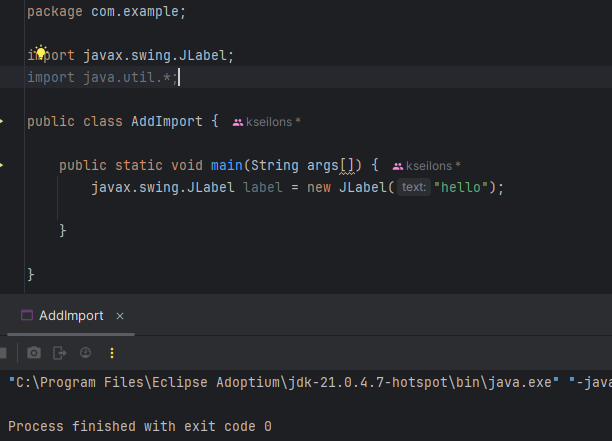


Общая сумма не поменялась.

У третьего гостя сумма в таком виде потому, что не все десятичные числа можно представить в типе double в виде конечного двоичного числа.

1. **Задания по уроку 4.2.**

* Импортируйте и откройте проект ImportEx.
* Изучите AddImport.java.
* Выполните следующие действия.
* Замените полное имя, используемое для доступа к компоненту Jlabel, оператором import.
* Чтобы импортировать классы из пакета util, замените несколько операторов import одним оператором import.



# Конспект.

Кроме базовых типов данных широко используются соответствующие им классы оболочки (wrapper – classes): Boolean, Character, Integer, Byte, Short, Long, Float, Double. Объекты этих классов могут хранить те же значения, что и соответствующие им базовые типы, но преобразуются в надлежащие классы, у которых появляется возможность использовать методы этих классов.

Другие символы, которые также называются операторами в языке Java, в том числе:

* точка "." указывает на имена пакетов и вызывает методы;
* круглые скобки "()" отделяют список параметров, разделенных запятыми, от имени метода;
* new создает экземпляр объекта (если за ним следует имя конструктора объекта). Конструктор - специальный метод класса, который используется для создания объекта. Класс - это некоторый шаблон, на основании которого создаются экземпляры класса - объекты.

Язык Java содержит операторы, которые позволяют программе принимать решения. Операторы отношения (>, >=, <=, ==, !=) используются, чтобы проверить отношение между двумя операндами, и возвращают boolean значение "истина" (true) или "ложь" (false) в зависимости от результата сравнения значений отношения.

Рассмотрим отложенные операторы. Оператор >> означает в языке Java сдвиг вправо. Он перемещает все биты своего левого операнда вправо на число позиций, заданное правым операндом. Когда биты левого операнда выдвигаются за самую правую позицию слова, они теряются. При сдвиге вправо освобождающиеся старшие (левые) разряды сдвигаемого числа заполняются предыдущим содержимым знакового разряда. Такое поведение называют расширением знакового разряда. Оператор << выполняет сдвиг влево всех битов своего левого операнда на число позиций, заданное правым операндом. При этом часть битов в левых разрядах выходит за границы и теряется, а соответствующие правые позиции заполняются нулями. Здесь используется автоматическое повышение типа всего выражения до int, в том случае, если в выражении присутствуют операнды типа int или целые типы меньшего размера. Если же хотя бы один из операндов в выражении имеет тип long, то и тип всего выражения повышается до long.

В методе printf сначала задается шаблон, согласно которому будет отформатирована строка, а потом передаются объекты для форматирования. Вот как выглядит метод printf с параметрами: System.out.printf(String format, Object... args); Формат обычно заключается в двойные кавычки, и определяет содержание вывода и тип выводимых данных с помощью следующих символов, начинающихся со знака "%" (Далее для простоты текста вместо System.out.printf() пишется printf()): %s - для типа String, например, printf("Hello %s!", "World"); // выводится "Hello World!" %n в составе формата выполняет переход на другую строку в выводе. Для типов byte, short, int, long: %d - вывод в десятеричном формате. %x - вывод в шестнадцатеричном формате. Здесь также можно указывать ширину поля вывода, например, так: %7d - в десятеричном формате и минимальной шириной поля 7 знаков. Оператор printf("%7d", 1); // выводит " 1"

Для типов float, double. %f - Десятичное число с точкой. %e - Десятичное число с точкой и экспонентой. Например, %.10f выводит с точностью 10 знаков после запятой. В отличие от println, printf использует не конкатенацию строк, а список переменных через запятую в порядке, определенном строкой формата.

Язык Java поддерживает условную операцию, которая также называется тернарная и записывается следующим образом: ? : < выражение2>; Если истинно, то результатом будет , в противном случае . Например, int x = a < b ? a : b; объявляет переменную x и присваивает вычисленный минимум значений из a и b.

Блок, или составной оператор — это произвольный состав простых операторов языка Java, заключенных в фигурные скобки. Блоки (как и методы) определяют область видимости своих переменных.