Федеральное государственное автономное образовательное учреждение

высшего образования

Санкт-Петербургский политехнический университет Петра Великого

Институт компьютерных наук и технологий

Высшая школа «Киберфизические системы и управление»

**Отчет №4**

по дисциплине «Системный подход к разработке программного обеспечения»

|  |  |  |
| --- | --- | --- |
| Выполнил:  студент гр. 3530902/00201 | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Перевалов В. В. |
|  | <*подпись*> |  |
|  |  |  |
|  |  |  |
| Руководитель:  Кандидат т.н. | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Нестеров С. А. |
|  | <*подпись*> |  |

«\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2023 г.

Санкт-Петербург

2023

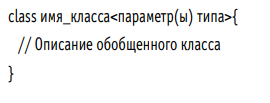
**Ход работы**

**А. Н. Васильев**

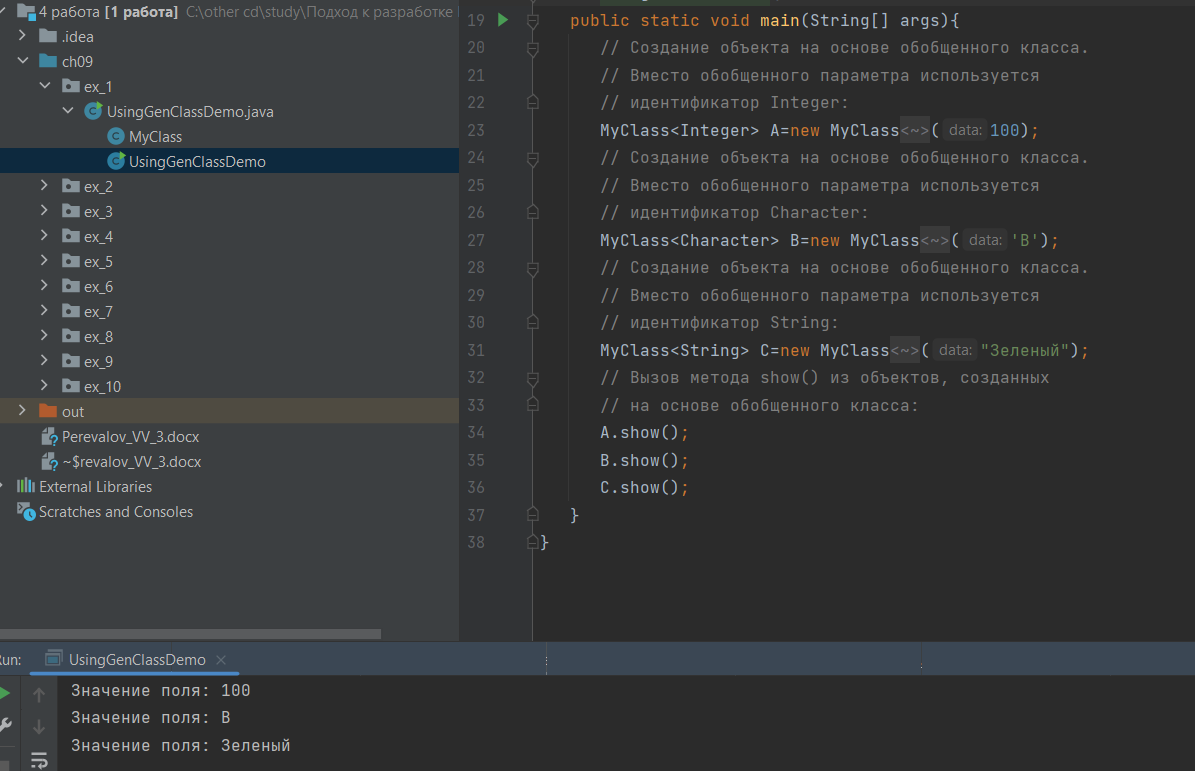
**Глава 9: Обобщенные типы данных**

В обобщенных классах тип данных выступает параметром. Обычно обобщенные классы используют в тех случаях, когда с помощью класса реализуется некоторая «структура» или «конструкция», которая имеет универсальные характеристики и слабо зависит от конкретного типа данных, с которыми приходится оперировать. Это же замечание относится и к методам: нередко приходится решать задачи, в которых алгоритм обработки данных мало зависит от типа данных.

Используется следующий шаблон описания обобщенного класса:

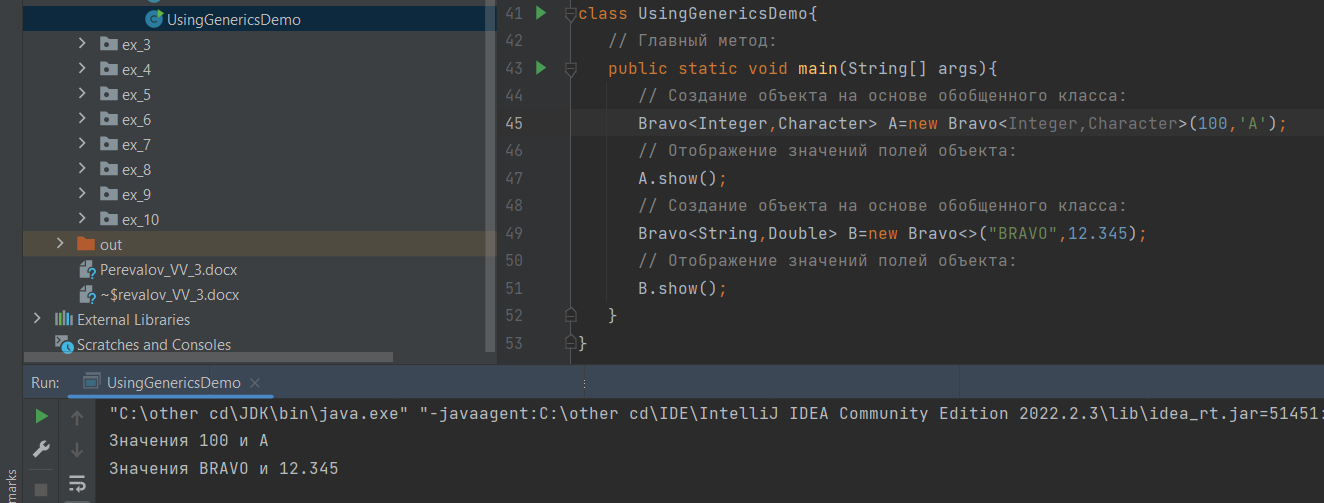


**Пример 1**

****

В главном методе программы создается три разных объекта. Все они создаются на основе обобщенного класса MyClass, но в качестве обобщенного параметра каждый раз используются разные значения. Так, команда MyClass<Integer> A=new MyClass<Integer>(100) означает, что при создании объекта вместо параметра X используется имя класса Integer, а аргументом конструктору передается значение 100. Соответственно, поле data объекта A получает значение 100, и оно же отображается при вызове из объекта A метода show() (команда A.show()). Аналогично создаются объекты B и C, но только вместо параметра X используются соответственно идентификаторы Character и String (и, разумеется, конструктору передаются разные значения).

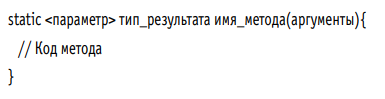
**Пример 2**

****

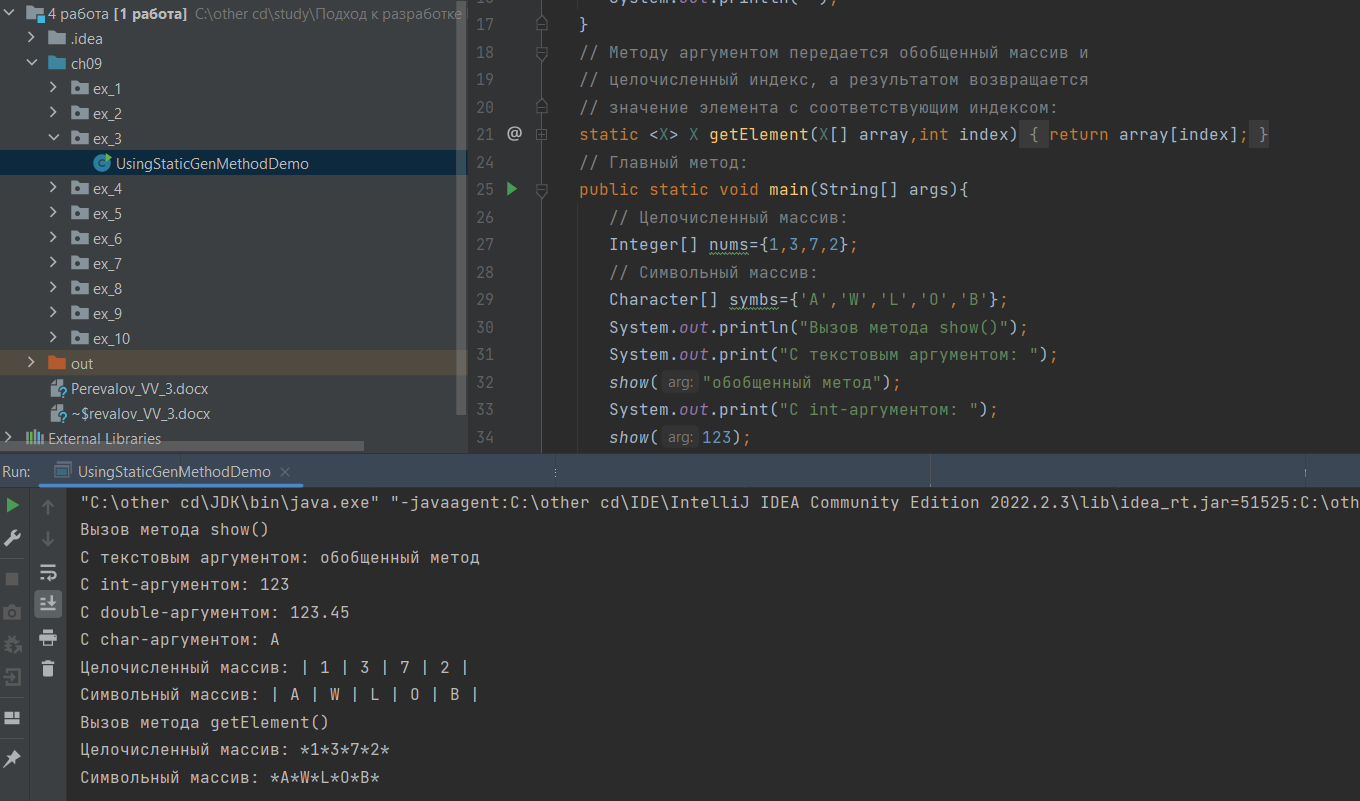
В классе Bravo объявлены два поля. Поле obj является переменной обобщенного класса Alpha с обобщенным параметром X. Поле объявлено инструкцией Alpha<X> obj. Также в классе Bravo объявляется поле second обобщенного типа Y. У конструктора, описанного в классе, два аргумента обобщенного типа: один типа X, и другой типа Y. В теле конструктора командой obj=new Alpha<X>(first) создается объект обобщенного класса Alpha с параметром X, и ссылка на объект записывается в поле obj. После этого присваивается значение полю second. Наконец, в классе Bravo описан метод show(), с помощью которого отображаются значения поле объекта, из которого вызывается метод.

**Обобщенные методы**

Помимо обобщенных классов, в Java можно создавать обобщенные методы . В описании обобщенного метода тип определяется через обобщенный параметр (или параметры), а при вызове метода значения обобщенных параметров определяются на основе типов аргументов, переданных методу. Обычно (но не обязательно) обобщенные методы являются статическими.

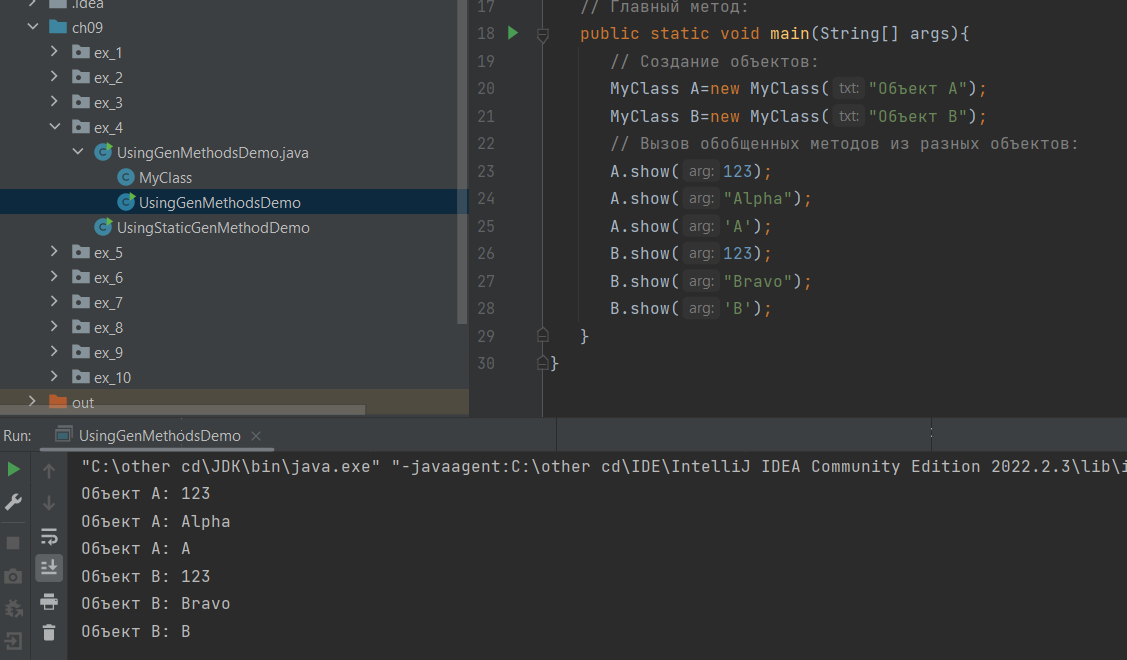


**Пример 3**

****

В первой версии метода show() ему передается аргумент обобщенного типа X. При вызове метода в окне вывода отображается значение аргумента. Второй версии метода show() передается массив, элементы которого относятся к обобщенному типу. При вызове метода отображаются значения элементов массива. У метода getElement() два аргумента: массив с элементами обобщенного типа и целочисленное значение, определяющее индекс элемента массива. Результатом метод возвращает значение элемента массива (первый аргумент) с данным индексом (второй аргумент).

**Пример 4**

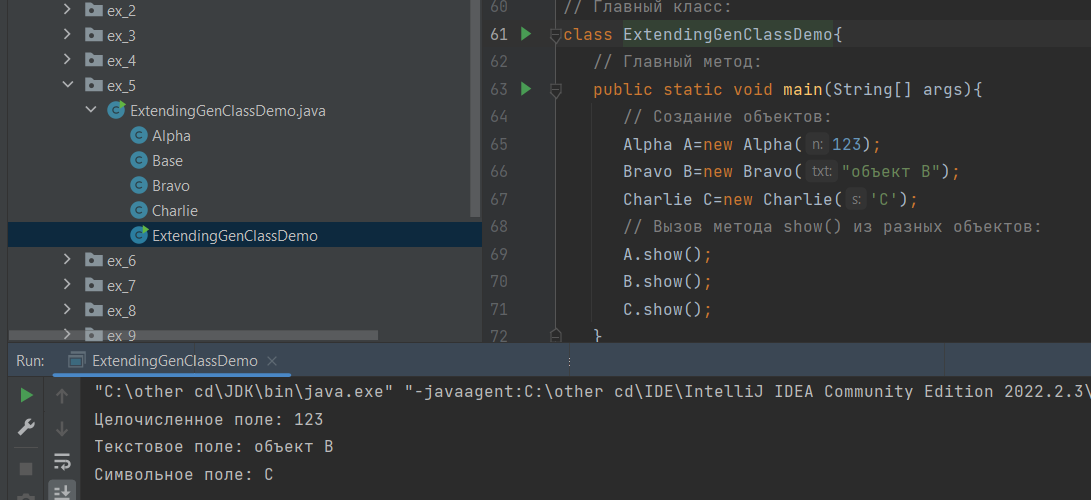
****

В данном случае описывается класс MyClass, который содержит текстовое поле name, конструктор с текстовым аргументом, а также обобщенный метод show(). Методу передается аргументом значение обобщенного типа (обозначен как X), а при выполнении метода отображается значение поля name и, через двоеточие, значение аргумента метода.

**Пример 5**

**Суперкласс на основе обобщенного класса**

Конкретные реализации обобщенного суперкласса могут использоваться в качестве суперкласса при наследовании. Проще говоря, при создании подкласса в качестве суперкласса можно указать обобщенный класс — но не как таковой, а с конкретными значениями для обобщенных параметров.

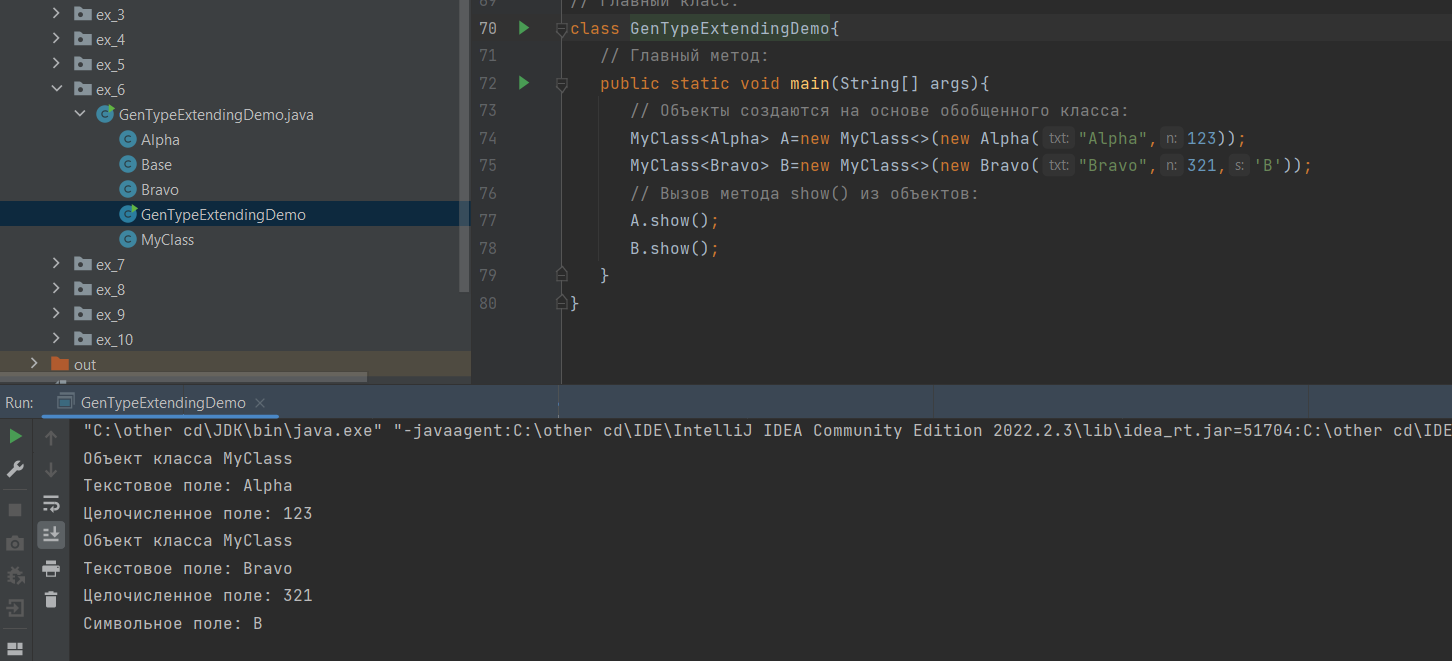


Мы использовали очень простую схему: описали обобщенный класс Base с обобщенным параметром X, а в классе описали поле data типа X, метод show(), при вызове которого отображается значение поля data, а также у класса есть конструктор с одним аргументом обобщенного типа. На основе класса Base путем наследования создаются классы Alpha, Bravo и Charlie. При этом в качестве обобщенного параметра указываются соответственно типы Integer, String и Character соответственно. Также в каждом и классов переопределяется метод show(). При этом с помощью инструкции super.show() вызывается версия метода из обобщенного суперкласса Base.

**Ограничение наследования для обобщенного типа**

При использовании обобщенных классов и методов в инструкции объявления обобщенного типа может использоваться выражение вида X extends Суперкласс , означающее, что обобщенный тип X должен быть таким, что является подклассом (прямым или опосредованным) класса Суперкласс.

**Пример 6**

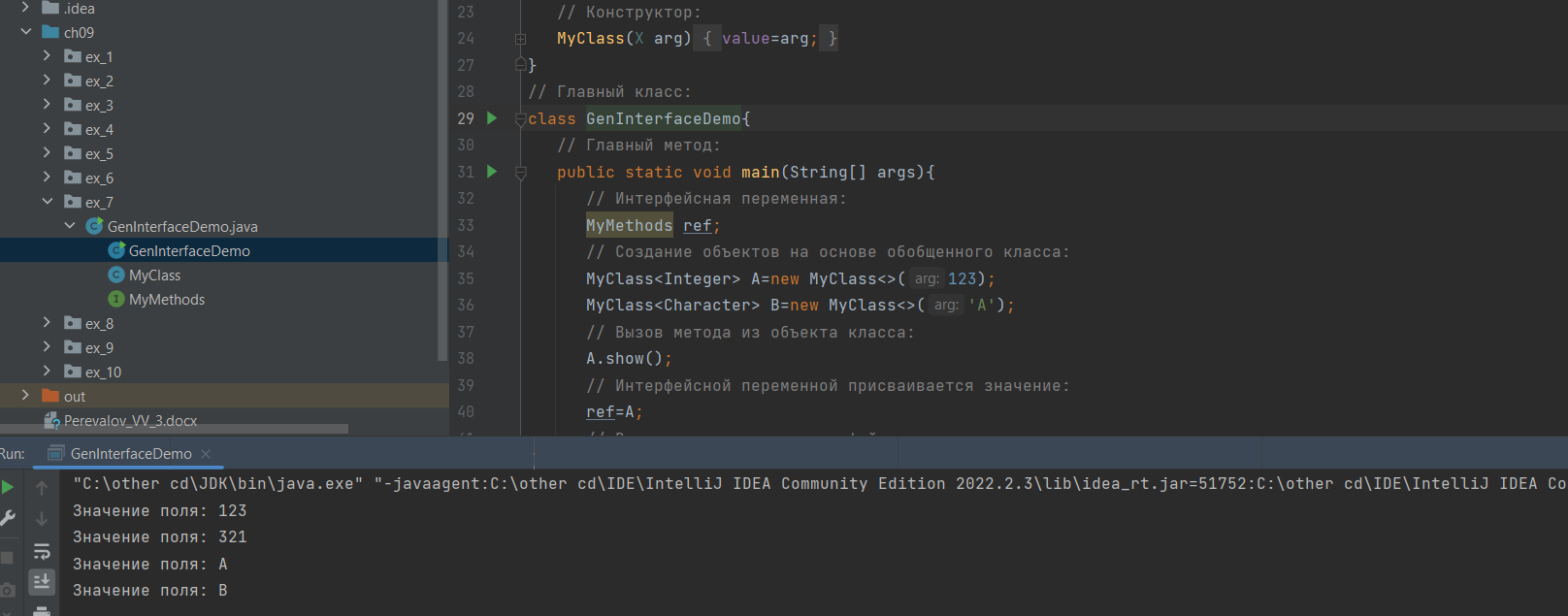


Обобщенный класс MyClass описывается с инструкцией <X extends Base>, означающей, что значением обобщенного параметра типа X при создании объекта класса MyClass может быть лишь класс, который является прямым или опосредованным (через несколько классов) наследником класса Base. Классы Alpha и Bravo для этой цели подходят.

**Обобщенные интерфейсы**

Обобщенным может быть не только класс или метод, но и интерфейс Объявляется обобщенный интерфейс аналогично к обобщенному классу: после имени интерфейса в угловых скобках указываются названия для обобщенных типов.

**Пример 7**



В главном методе программы объявляется интерфейсная переменная ref интерфейса MyMethods, а также создаются переменные A и B на основе обобщенного класса MyClass соответственно со значениями Integer и Character для обобщенного параметра типа. Для каждого из объектов выполняется такая процедура:

• из объекта вызывается метод show();

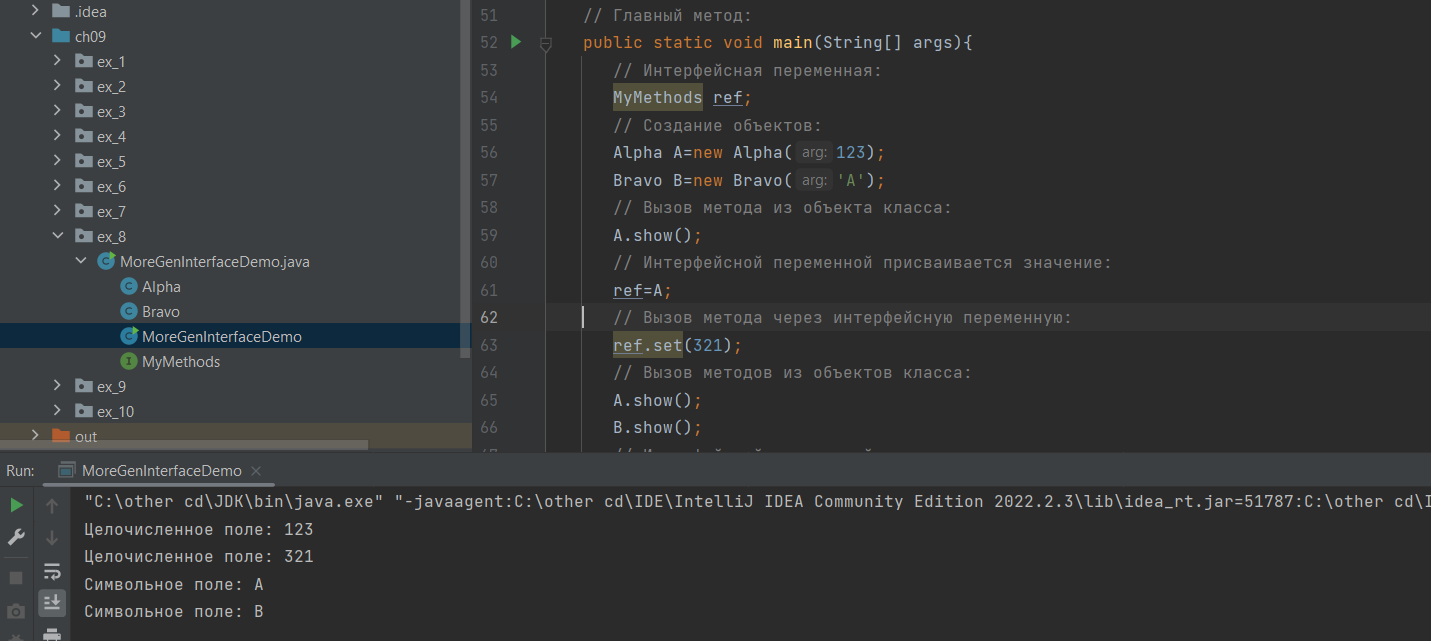
• переменной ref присваивается ссылка на соответствующий объект;

• через интерфейсную переменную ref вызывается метод set(), благодаря чему закрытому полю объекта присваивается новое значение;

• через объектную переменную снова вызывается метод show().

**Создание обычного класса на основе обобщенного интерфейса**

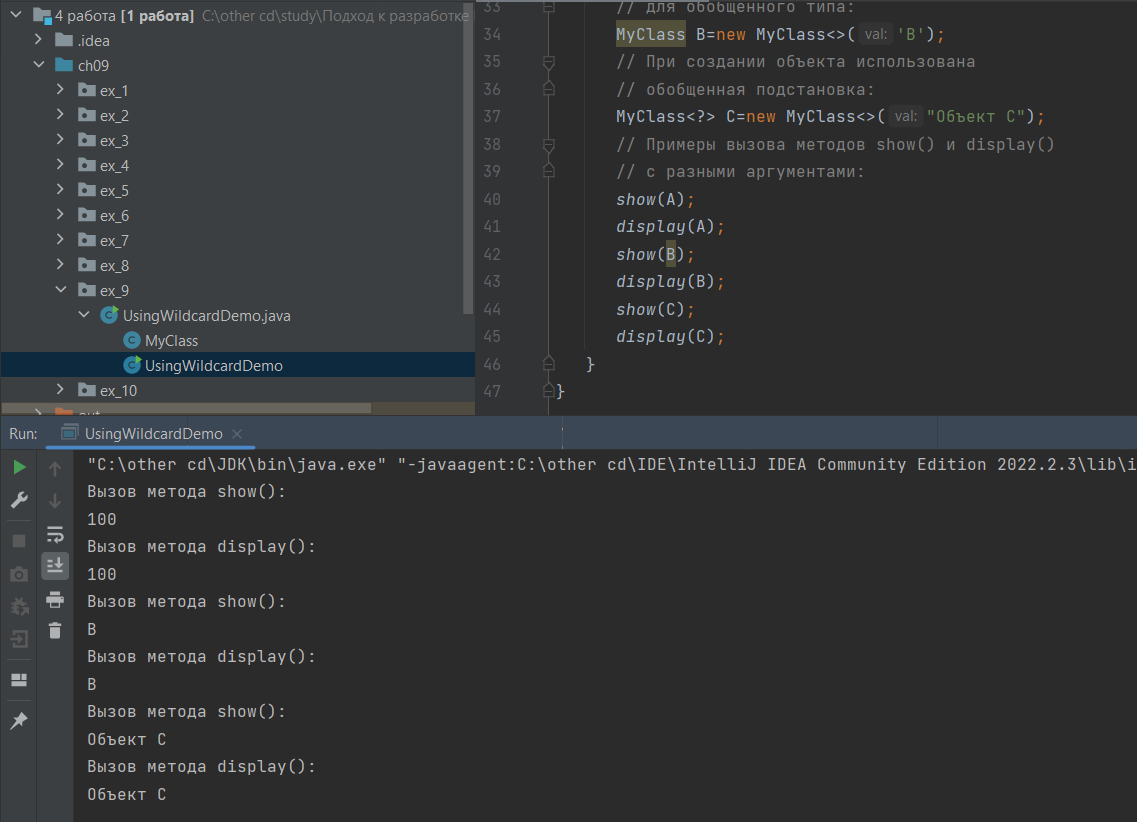
**Пример 8**



Как видим, результат выполнения программы такой же, как и в предыдущем случае.

**Обобщенные подстановки**Идея обобщенной подстановки в принципе проста: используя в качестве определения обобщенного параметра инструкцию вида можно сообщить компилятору, что в соответствующем месте используется некоторый неизвестный обобщенный тип.

**Пример 9**

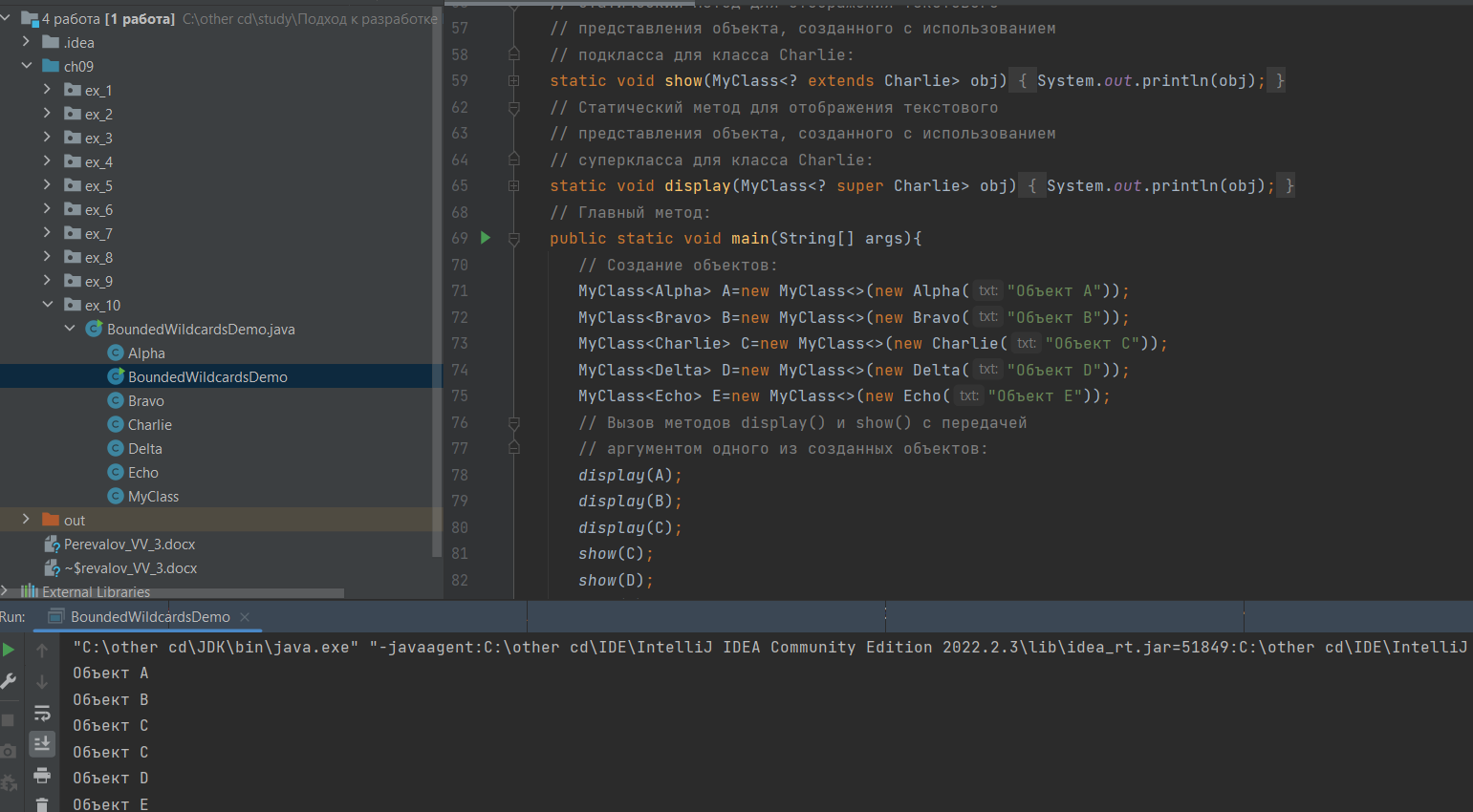


Итак, метод show() описан как обобщенный. Параметр обобщенного типа обозначен как T. Тип аргумента метода определен выражением MyClass. Сам аргумент обозначен как obj. В теле метода отображается текстовое сообщение и значение поля value объекта obj. Таким образом, обобщенный тип T здесь в явном виде не используется. Это «намек» на то, что можно использовать обобщенную подстановку. Такой подход использован при описании метода display(). Аргумент obj этого метода описан с типом MyClass. Данная инструкция означает, что аргумент является объектом обобщенного класса MyClass с некоторым значением для параметра обобщенного типа, но это значение не конкретизируется (не вычисляется). Можно сказать и так: какой-то тип использован, но какой именно — не важно. Все остальное в методе display() практически такое же, как в методе show().

**Обобщенные подстановки с ограничениями**

При использовании обобщенных подстановок можно использовать огра ничения : с помощью ключевого слова extends определяют подстановки, обозначающие подклассы для данного класса, а с помощью ключевого слова super определяют подстановки для суперклассов данного класса. В частности, выражение вида означает, что для параметра обобщенного типа используется неустановленный тип, являющийся подклассом (прямым или опосредованным) класса SuperClass. Выражение вида означает, что в качестве параметра обобщенного типа может быть класс, являющийся (прямо или через цепочку наследования) суперклассом для класса SubClass.

**Пример 10**

****

В главном методе программы с использованием классов Alpha, Bravo, Charlie, Delta и Echo и обобщенного класса MyClass создаются объекты A, B, C, D и E, а затем данные объекты передаются аргументами методам display() и show(). С учетом указанных выше ограничений, методу display() аргументом можно передавать только объекты A, B и C, а аргументом методу show() можно передавать только объекты C, D и E.

**И. Н. Блинов, В.С, Романчик**

**Перечисления**

Типобезопасные перечисления (typesafe enums) в Java представляют собой классы и являются подклассами абстрактного класса **java.lang.Enum**. Вместо слова class при описании перечисления используется слово **enum**. При этом объекты перечисления инициализируются прямым объявлением **без помощи оператора new**.

Перечисление как подкласс класса Enum может содержать поля, конструкторы и методы, реализовывать интерфейсы. Каждый элемент enum может использовать методы:

**static enumType[] values()** — возвращает массив, содержащий все элементы перечисления в порядке их объявления;

**static > T valueOf(Class enumType, String arg**) — создает элемент перечисления, соответствующий заданному типу и значению передаваемой строки;

**static enumType valueOf(String arg)** — создает элемент перечисления, соответствующий значению передаваемой строки;

**int ordinal()**— возвращает позицию элемента перечисления;

**String name()** — возвращает имя элемента;

**int compareTo(T obj)** — сравнивает элементы на больше-меньше либо равно.

Без throw данный код не будет компилироваться, так как компилятор не исключает появления неизвестного элемента. Данная инструкция позволяет указать на возможную ошибку при появлении необъявленной фигуры. Поэтому и при введении нового элемента желательно добавлять соответствующий ему case.

Конструктор не может быть объявлен со спецификаторами public и protected, так как не вызывается явно и перечисление не может быть суперклассом.

Метод toString() реализован в классе Enum для вывода элемента в виде строки. Если переопределить метод toString() в конкретной реализации перечисления, то можно выводить не только значение элемента, но и значения его полей, то есть предоставить полную информацию об объекте, как и определяется контрактом метода. Однако на перечисления накладывается целый ряд ограничений.

Им запрещено:

• быть суперклассами;

• быть подклассами;

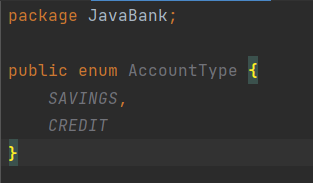
• быть абстрактными;

• создавать экземпляры, используя ключевое слово new

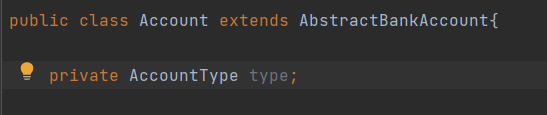
**Задания по занятию 3\_1**

6-9

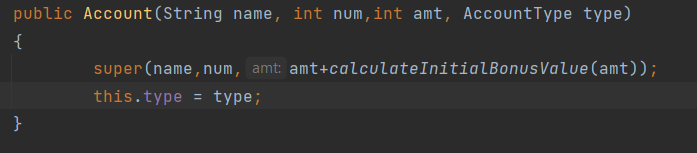
Создаем перечисление:



Добавляем поле в класс аккаунта:



Обновим конструктор:

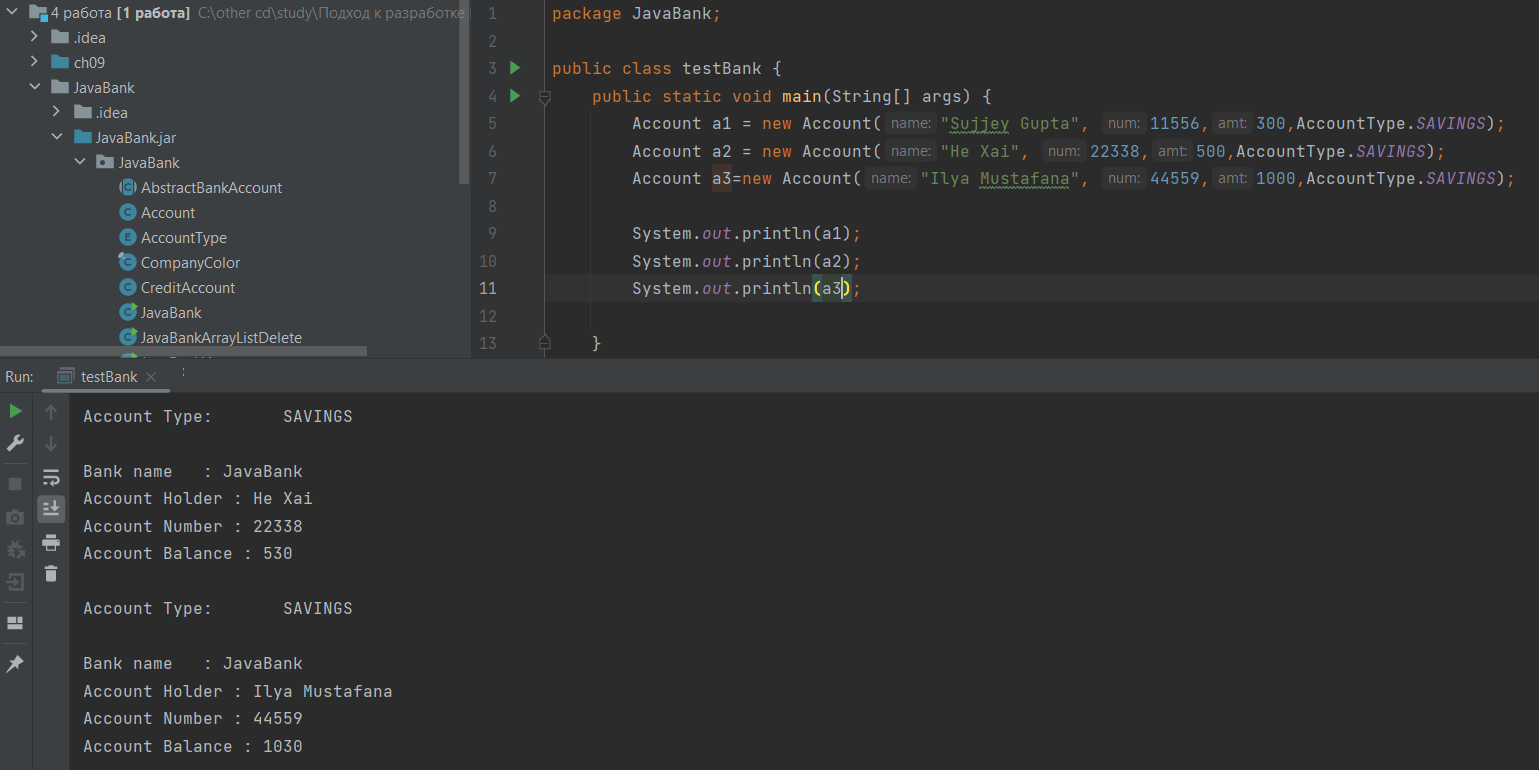


Напишем toString :

Изображение выглядит как текст

Автоматически созданное описание

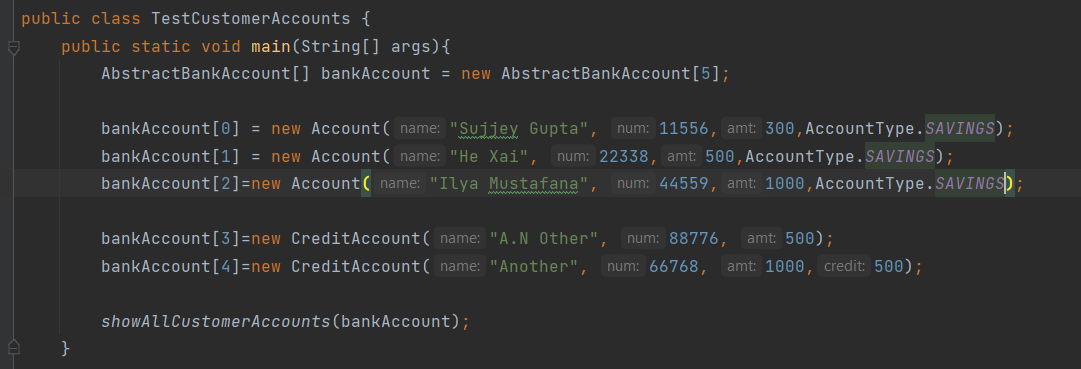
Изменим testBank и протестируем:



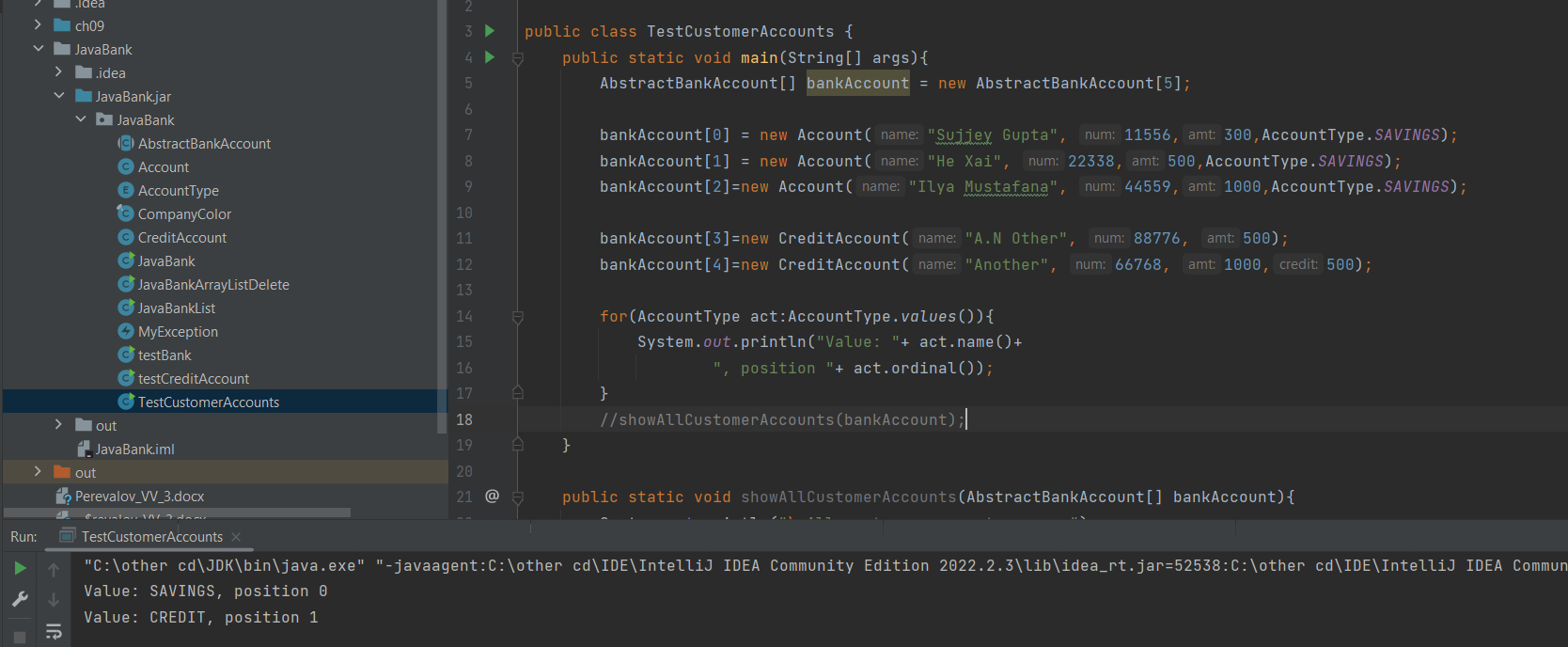
Адаптируем остальные файлы для тестирования:

Изображение выглядит как текст

Автоматически созданное описание



Проитерируемся по перечислению:



**14**

Модифицируем AccountType:

**Изображение выглядит как текст

Автоматически созданное описание**

Введем второе поле:

Изображение выглядит как текст

Автоматически созданное описание

Напишем геттер:

Изображение выглядит как текст

Автоматически созданное описание

**Слайд 29**

Напишем класс Cell с двумя типами данных:

Изображение выглядит как текст

Автоматически созданное описание

Напишем класс CellDriver:

Изображение выглядит как текст

Автоматически созданное описание

Слайд 39

Напишем метод, который может распечатать массив произвольных элементов:

Изображение выглядит как текст

Автоматически созданное описание

Напишем тест и запустим его:

Изображение выглядит как текст

Автоматически созданное описание

Все работает корректно.

**43-45**

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

**47-48**

Напишем метод, позволяющие посчитать сумму элементов массива, содержащего числовые данные:

Изображение выглядит как текст

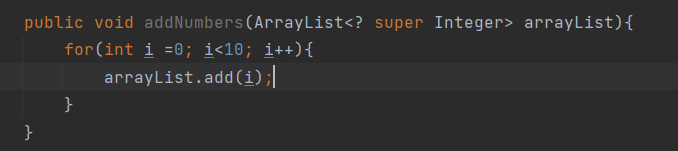
Автоматически созданное описание

Протестируем:

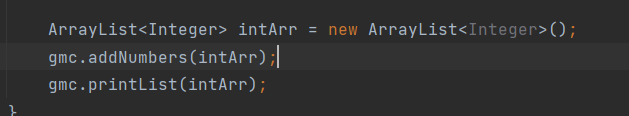
Изображение выглядит как текст, снимок экрана, внутренний

Автоматически созданное описание

**50-51**

****

Протестируем:



**Практика 3\_1**

Vocabulary

|  |  |
| --- | --- |
| Generic | This is a special type of class that associates one or more non-specified Java types |
| <T> | A type interface diamond is what 2 characters |
| Enumerated type | A datatype that contains a fixed set of constants |

**Задание 1**

Add a private JComboBox named accountTypes above the MAXACCOUNTS statement. Also create a private field named actType of type AccountType that will store the type of chosen account, this should be set to the default savings account.

**Изображение выглядит как текст

Автоматически созданное описание**

Under the code that sets up the displayJButton but before the code that sets up the displayJLabel add the following code:

**Изображение выглядит как текст

Автоматически созданное описание**

Run the application, click proceed when the error message is displayed! You will see that the combo box is not displayed properly as some of the interface components require to be updated.

Update the size of the window.

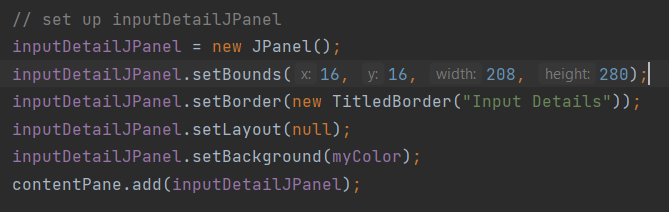
1. Go to the end of the createUserInterface method and identify the code that sets the values for the application’s window.
2. Update the size of the window from (670, 308) to increase the height to 340.

**Изображение выглядит как текст

Автоматически созданное описание**

Update the size of the panel that contains the interactive components.

1. Identify the code that sets up the inputDetailJPanel.
2. ii. Currently the bounds settings (X, Y, Width, Height) are (16, 16, 208, 250). You need to increase the Height of the window by 30 pixels.

****

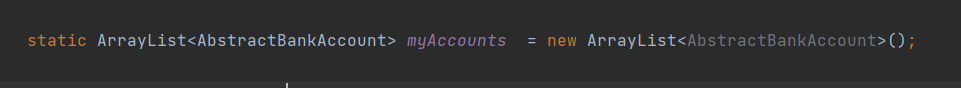
Update the size of the text area to better suit the new size of the application window.

1. Identify the code that sets up the displayJTextArea.
2. **Изображение выглядит как текст

   Автоматически созданное описание** ii. Increase the Height of the window to a value of 245.

**Задание 2**

Update the myAccounts array declaration so that it instantiates AbstractBankAccount objects and not Account objects.

****

Update the if statement that adds accounts to the system (After the finally block in the createAccountJButtonActionPerformed method) to include an if statement that checks the account type.

**Изображение выглядит как текст

Автоматически созданное описание**

**Задание 3**

Create an Enum class named BikeUses that includes the following values:

**Изображение выглядит как текст

Автоматически созданное описание**

Update the interfaces to use the enum for the terrain.

1. Update the MountainParts interface by commenting out the original line that sets the terrain and replace it with the Enum declaration using the off\_road value

**Изображение выглядит как текст

Автоматически созданное описание**

1. Do the same with the RoadParts interface using the Enum value track for the terrain.
2. **Изображение выглядит как текст

   Автоматически созданное описаниеИзображение выглядит как текст

   Автоматически созданное описаниеИзображение выглядит как текст

   Автоматически созданное описание**Update the toString() method in both bike subclasses to also display the terrain information so tha the output matches:

**4 Задание**

**Изображение выглядит как текст

Автоматически созданное описание**

**Изображение выглядит как текст

Автоматически созданное описаниеИзображение выглядит как текст

Автоматически созданное описание**

**ОСНОВЫ АЛГОРИТМИЗАЦИИ**

Алгоритм — это набор команд, необходимых для решения той или иной задачи. Он определяет шаги, согласно которым она будет выполняться.

Псевдокод — это текст, похожий на язык программирования, но не являющийся таковым. Он описывает структуры и детали, которые вам понадобятся для применения алгоритма в полноценном коде вне зависимости от используемого языка программирования.

Асимптотическая сложность (производительность) определяется функцией, которая указывает, насколько ухудшается работа алгоритма с усложнением поставленной задачи. Такую функцию записывают в круглых скобках, предваряя прописной буквой О.

Существуют пять основных правил для расчета асимптотической сложности алгоритма.

1. Если для математической функции f алгоритму необходимо выполнить определенные действия f(N) раз, то для этого ему понадобится сделать O(f(N)) шагов.

2. Если алгоритм выполняет одну операцию, состоящую из O(f(N)) шагов, а затем вторую операцию, включающую O(g(N)) шагов, то общая производительность алгоритма для функций f и g составит O(f(N) + g(N)).

3. Если алгоритму необходимо сделать O(f(N) + g(N)) шагов и область значений N функции f(N) больше, чем у g(N), то асимптотическую сложность можно упростить до выражения O(f(N)).

4. Если алгоритму внутри каждого шага O(f(N)) одной операции приходится выполнять еще O(g(N)) шагов другой операции, то общая производительность алгоритма составит O(f(N) g(N)).

5. Постоянными множителями (константами) можно пренебречь. Если C является константой, то O(C f(N)) или O(f(C N)) можно записать как O(f(N)).

Log (N)

Алгоритм с производительностью O(log (N)) делит количество рассматриваемых им элементов на фиксированный коэффициент при каждом шаге.

Полное завершенное двоичное дерево высотой H включает примерно 2H вершины. Если исходить из обратного, то N вершин составят дерево высотой log2 (N). Поскольку алгоритм движется по дереву сверху вниз в худшем (и близком к нему) случае, а само дерево имеет высоту приблизительно log2 (N), работа алгоритма займет время, равное O(log2 (N)).

Sqrt(N)

Алгоритмы с производительностью O(sqrt (N)), где sqrt — функция извлечения квадратного корня. Эта функция возрастает очень медленно, хотя и несколько быстрее, чем log (N).

N

Функция N возрастает быстрее, чем log (N) и sqrt (N), но все же не так быстро, поэтому большинство подобных алгоритмов демонстрирует на практике хорошую производительность.

N log N

Предположим, что алгоритм перебирает все элементы в поставленной задаче, а затем в отдельном цикле выполняет с элементом какую-то операцию O(log (N)). 37 Основы алгоритмизации В этом случае производительность алгоритма определяется выражением O(N log N) или O(N log N). Как вариант, задача может выглядеть так: алгоритм выполняет операцию O(log (N)) и на каждом ее шаге что-то делает с каждым элементом.

Предположим, у вас есть отсортированное дерево, содержащее N элементов (как описано выше) и такой же размерности массив. Вам надо узнать, какие из элементов массива присутствуют в дереве. Один из способов выяснить это — осуществить циклическое прохождение по величинам массива, задействовав описанный ранее метод поиска по дереву. В процессе работы алгоритм проверит N элементов и выполнит log (N) шагов для каждого из них, так что общая производительность будет O(N log N).

N^2

Алгоритм, который изначально перебирает все входные данные, а затем пересматривает их еще раз для каждого единичного значения, имеет производительность O(N^2).

Говорят, алгоритм имеет многочленное время работы, если оно включает в себя многочлен N: например O(N), O(N^2 ), O(N^6 ) и даже O(N^4000). В любом случае подобные задачи можно решить. А вот экспоненциальное и факториальное время работы, описанное ниже, возрастает очень быстро, поэтому алгоритмы с указанной производительностью будут применимы только для небольшого количества входных данных.

2^N

Экспоненциальные функции, такие как 2^N, возрастают молниеносно и поэтому полезны для решения ограниченного круга задач. Обычно посредством алгоритмов с подобным временем работы ищется оптимальный набор входных данных.

При решении задач с экспоненциальным временем работы часто пользуются эвристиками — алгоритмами, которые дают хорошие результаты, но не гарантируют, что они будут наилучшими.

N!

Функция N! (читается как «N факториал») рекомендуется для работы с целыми числами больше 0 и определяется формулой N! = 1 2 3 ... N. Она возрастает намного быстрее, чем экспоненциальная функция 2N.

В алгоритмах с факториальным временем работы, как правило, ищется оптимальное распределение входных данных. Например, у торгового представителя имеется список городов. Его задача — составить маршрут таким образом, чтобы посетить каждый населенный пункт один раз и вернуться в отправную точку, преодолев минимальное расстояние.

Изображение выглядит как стол

Автоматически созданное описание