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 по лабораторной работе**

по дисциплине «Системный подход к разработке программного обеспечения»

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

«\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2023 г.

Санкт-Петербург

2023

**Васильев**

**Пример 1 UsingGenClassDemo**

Допустим, что перед нами стоит задача по описанию класса, у которого есть поле определенного типа, конструктор с одним аргументом и метод, которым отображается значение поля. Понятно, что каким бы ни был фактический тип поля, код класса достаточно универсальный. Разные версии класса для разных типов (простых) поля отличались бы только идентификатором типа в инструкции объявления поля и описании конструктора. Это как раз тот случай, когда удобно использовать обобщенный класс. Именно так и поступим. Соответствующий программный код представлен в листинге 9.1.

**Изображение выглядит как текст

Автоматически созданное описание**

**Пример 2 UsingGenericsDemo**

Следующий пример, который мы рассмотрим, содержит описание нескольких обобщенных классов и в некоторых из них используется более одного обобщенного параметра. Программный код примера представлен в листинге 9.2.

**Изображение выглядит как текст

Автоматически созданное описание**

**Пример 3 UsingStaticGenMethod**

Параметр (или параметры) используются в описании аргументов метода и/или типа результата метода. При вызове метода это дает возможность идентифицировать значения обобщенных параметров, которые следует использовать при выполнении кода метода. Небольшой пример, в котором объявляются и используются статические обобщенные методы, представлен в листинге 9.3.

**Изображение выглядит как текст

Автоматически созданное описание**

**Пример 4 UsingGenMethodsDemo**

Нестатический обобщенный метод создается практически точно так же, как и статический. Принципиальное отличие в том, что нестатический метод вызывается из объекта. Небольшой пример, в котором создается и используется нестатический метод, представлен в листинге 9.4.

**Изображение выглядит как текст

Автоматически созданное описание**

**Пример 5 ExtendingGenClassDemo**

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. Пример такого подхода представлен в программе в листинге 9.5.

**Изображение выглядит как текст

Автоматически созданное описание**

**Пример 6 GenTypeExtanding**

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. В некоторых случаях такой подход бывает достаточно удобен. Небольшая иллюстрация представлена в листинге 9.6.

**Изображение выглядит как текст

Автоматически созданное описание**

**Пример 7 GenInterfaceDemo**

Если на основе обобщенного интерфейса создается обобщенный класс, то обобщенный параметр типа указывается после имени класса и после имени реализуемого в классе интерфейса. Пример такой ситуации представлен в листинге 9.7.

**Изображение выглядит как текст

Автоматически созданное описание**

**Пример 8 MoreGenInterfaceDemo**

На основе обобщенного интерфейса можно создавать обычные классы. Небольшая программная вариация, отдаленно напоминающая предыдущий пример. Рассмотрим программный код в листинге 9.8.

**Изображение выглядит как текст

Автоматически созданное описание**

**Пример 9 UsingWildcardDemo**

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. Например, представим себе такую ситуацию: в программе определен обобщенный класс MyClass с одним обобщенным параметром (обозначим его как T). В классе объявлено поле value обобщенного типа T и конструктор с одним аргументом обобщенного типа. Аргумент конструктора присваивается значением полю value. В классе UsingWildcardDemo описывается главный метод main() и два статических не возвращающих результат метода: show() и display(). Оба метода выполняют практически одинаковую «работу», но описаны по-разному. Оба метода предназначены для отображения значения поля value объек та, созданного на основе обобщенного класса MyClass и переданного аргумен том методу. В главном методе программы приведены примеры создания объектов на основе обобщенного класса MyClass с последующей их передачей аргументами методам show() и display().

**Изображение выглядит как текст

Автоматически созданное описание**

**Пример 10 BoundedWildcardsDemo**

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. Небольшой пример использования таких обобщенных подстановок приведен в листинге 9.10.

**Изображение выглядит как текст

Автоматически созданное описание**

**Резюме**

• В обобщенных классах, методах и интерфейсах тип может передаваться через параметр. Параметры обобщенного типа для обобщенного класса или интерфейса указываются в угловых скобках после имени класса. Параметры обобщенного типа для обобщенного метода указываются в угловых скобках перед идентификатором типа результата метода. Параметры типа могут использоваться в теле обобщенного класса, интерфейса или метода.

• При создании объекта на основе обобщенного класса значения для параметров обобщенного типа, используемые при создании объекта, указываются в угловых скобках после основного имени обобщенного класса. Значениями параметров обобщенного типа могут быть только ссылочные типы (классы), и ими не могут быть базовые (примитивные) типы. Значения параметров обобщенного типа при вызове обобщенного метода определяются исходя из контекста команды вызова метода.

• На параметры обобщенного типа можно накладывать ограничение. С использованием ключевого слова extends значения для параметра обобщенного типа можно ограничить подклассами определенного класса.

• Класс может создаваться на основе обобщенного интерфейса.

• Обобщенная подстановка позволяет задекларировать использование обобщенного типа, не конкретизируя его значение. С использованием ключевых слов extends и super можно ограничить возможные значения для типа, определяемого через обобщенную подстановку: соответственно, возможные значения параметра типа ограничиваются подклассами определенного класса или суперклассами (по цепочке наследования) определенного класса.

• Существует ряд ограничений, которые следует иметь в виду при работе с обобщенными классами и методами. Так, например, в теле обобщенного класса нельзя создать объект на основе класса, определяемого обобщенным параметром (но можно использовать объектную переменную соответствующего класса).

**Блинов, Романчик**

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. При инициализации хотя бы одного перечисления происходит инициализация всех без исключения оставшихся элементов данного перечисления.

**Изображение выглядит как текст

Автоматически созданное описание**

**Изображение выглядит как текст

Автоматически созданное описание**

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.

Перечисление является классом, поэтому в его теле можно объявлять кроме методов также поля и конструкторы. Все конструкторы вызываются автоматически при инициализации любого из элементов. Конструктор не может быть объявлен со спецификаторами public и protected, так как не вызывается явно и перечисление не может быть суперклассом. Поля перечисления используются для сохранения дополнительной информации, связанной с его элементом.

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.

**Стивенс**

Алгоритмы — это наборы команд, способствующие эффективному программированию. Они объясняют, как сортировать записи, искать элементы, рассчитывать числовые значения (например, простые множители), находить кратчайший путь между двумя точками на карте, определять максимально возможный поток информации по сети и т. д. Хороший алгоритм способен решить задачу в считаные секунды, плохой потребует на это часы или же не решит ее вовсе.

Изучение алгоритмов поможет вам овладеть различными полезными методами и научиться выбирать те из них, что наилучшим образом подходят к вашему сценарию. Ведь для решения какой-либо задачи можно использовать разную методику, но только определенная окажется идеальной. Кроме того, действия, показавшие отличные результаты с одним набором данных, с другим могут просто не сработать.

Прежде чем приступить к изучению алгоритмов, рассмотрим несколько важных моментов. Для начала вы должны знать, что алгоритм — это набор команд, необходимых для решения той или иной задачи. Он определяет шаги, согласно которым она будет выполняться. Такое определение кажется довольно простым, однако никто не станет писать алгоритмы для выполнения примитивных задач или создавать инструкции, чтобы получить доступ к четвертому элементу массива (подразумевается, что вы уже владеете минимальными навыками программирования и знаете, как это делается). Как правило, алгоритмы пишутся только для сложных задач, например в том случае, когда нужно найти кратчайший путь через сеть из сотен улиц или отыскать наилучший вариант инвестиций для оптимизации прибыли. В этой главе объясняются некоторые основные положения алгоритмизации, о которых вам следует иметь полное представление, особенно если есть желание извлечь максимальную пользу от чтения книги. Возможно, вы захотите пропустить приведенную здесь информацию и сразу приступить к изучению специальных алгоритмов. Однако лучше хотя бы поверхностно ознакомиться с изложенным материалом. Подробно изучите подраздел «Асимптотическая сложность алгоритма» в разделе «Алгоритм и структура данных» текущей главы, поскольку понимание времени существенно для выбора нужного алгоритма. Очень важно, справится он с заданием за секунды, часы или не справится вообще.

Алгоритм и структура данных

Как сказано выше, алгоритм представляет собой набор команд для выполнения какой-либо задачи. При этом все данные, необходимые для ее решения, организуются особым образом в так называемую структуру. Это может быть массив, связный список, дерево, граф, сеть или что-то более замысловатое. Алгоритмы не могут существовать без структур данных. Например, алгоритм редакторского расстояния, который описывается в главе 15 и устанавливает схожесть двух строк, тесно связан с сетью и без нее не работает. Точно так же нет смысла строить структуру данных, если вы не планируете использовать ее вместе с алгоритмом.

Для описания алгоритмов в книге используются интуитивно понятные английские термины. Это сделано для того, чтобы рассматриваемые примеры можно было применять в большинстве языков программирования. Однако часто использование алгоритма связано с некоторыми нюансами, которые приводятся в псевдокоде. 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. В следующем фрагменте показан пример псевдокода для вычисления наибольшего общего делителя (НОД) двух целых чисел.

Изображение выглядит как текст

Автоматически созданное описание

Чтобы проверить эффективность алгоритма, программисты задаются вопросом: как изменится производительность кода, если скорректировать размерность задачи? Если удвоить количество переменных в алгоритме, не удвоится ли время его работы? А если возвести начальное количество переменных в четвертую степень, не станет ли полученный алгоритм выполняться несколько лет? В таком же ключе следует рассуждать о расходе оперативной памяти и других ресурсах, необходимых для работы алгоритма. Стоит озаботиться и производительностью алгоритма при различных условиях. Каков наихудший вариант? Сколь велика вероятность его возникновения? Если запустить алгоритм с использованием большого объема произвольных данных, то какова будет его средняя производительность? Чтобы получить представление о том, как сложность задачи влияет на производительность, программисты пользуются понятием «асимптотическая сложность». Об этом мы и поговорим ниже.

Изображение выглядит как текст

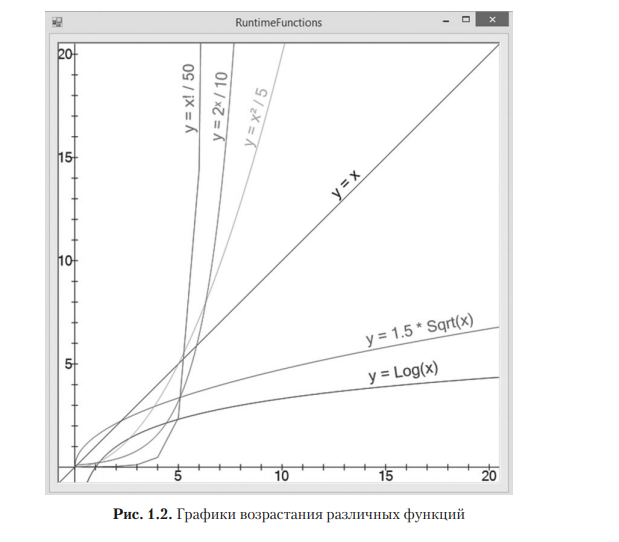
Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как стол

Автоматически созданное описание



**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.

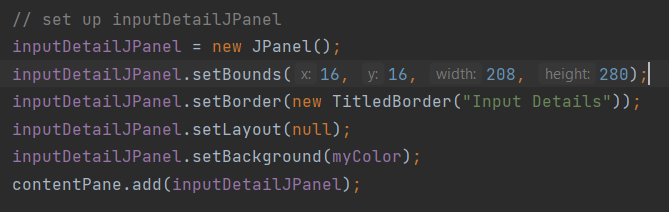
1. Go to the end of the createUserInterface method and identify the code that sets the values for the application’s window.
2. Update the size of the window from (670, 308) to increase the height to 340.

**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.

****

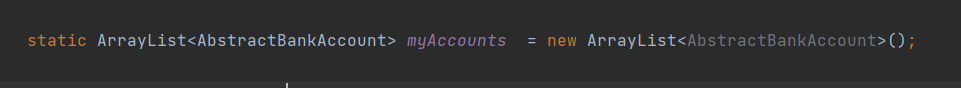
Update the size of the text area to better suit the new size of the application window.

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**