Практическая работа №4. Перечисления и их использование в Джава программах.

Цель данной практической работы – познакомиться с новым ссылочным типом данных перечислением, научиться разрабатывать перечисления и использовать их в своих программах.

Теоретические сведения

Перечисления это один из объектных типов в Джава. Они являются безопасными типами, поскольку переменная тип перечисление может принимать значение только константу из перечисления. Рассмотрим пример объявления перечисления в языке Джва:

```
public enum Level {
    HIGH,
    MEDIUM,
    LOW
}
```

Обратите внимание на enum - ключевое слово перед именем перечисления, которое используется вместо class или interface. Ключевое слово enum сигнализирует компилятору Java, что это определение типа является перечислением. Вы можете ссылаться на константы в приведенном выше перечислении следующим образом:

```
Level level = Level.HIGH;
```

В этом случае переменная level принимает значение HIGH.

Обратите внимание, что level переменная имеет тип, Level который является типом перечисления Java, определенным в приведенном выше примере. Level.

Переменная может принимать одно значение из констант перечисления Level, то есть в качестве значения может принимать значения HIGH, MEDIUM, или LOW.

Использование перечислений в операторах ветвления if () else

Поскольку все перечисления в Джава являются константами, то вам часто придется сравнивать переменную, указывающую на константу перечисления, с возможными константами в типе перечисления. Вот пример использования перечисления в операторе if () else:

```
Level level = ...
//Присваиваем некоторое значение из перечисления
Level
if( level == Level.HIGH) {
} else if( level == Level.MEDIUM) {
} else if( level == Level.LOW) {
}
```

В этом коде переменную level сравнивается с каждой из возможных констант перечисления в перечислении Level. Если оно соответствует одному из значений в перечислении, то проверка этого значения в первом операторе if приведет к повышению производительности, так как в среднем выполняется меньше сравнений.

Использование перечислений в операторах множественного выбора switch

Если ваши типы перечисления Java содержат много констант и вам нужно проверять переменную на соответствие нескольким значениям из перечисления, то воспользуйтесь оператором идеей switch.

Вы можете использовать перечисления в операторах switch следующим образом:

```
Level level = ...
// присвоить ему некоторую константу Level

switch (level) {
    case HIGH: ...;break;
    case MEDIUM: ...; break;
    case LOW: ...; break;
}
```

Замените многоточие ... в коде выше на то значение, которое вам нужно для выполнения кода, если переменная level истинно, то выполнится соответствующая ветка switch, т.е. в выражении переменная совпадет со значением константы соответствует заданному значению константы присвоенному level. Код, который соответствует определенной ветви switch может быть просто блок выражений или например может выполняться вызов метода и т.д.

Итерация перечислений

Вы можете получить массив всех возможных значений типа перечисления Джава, вызвав его статический метод values(). Все типы перечислений автоматически получают статический метод values() с помощью компилятора Джава. Вот пример итерации всех значений перечисления:

```
for (Level level : Level.values()) {
    System.out.println(level);
}
```

Запуск кода выше распечатает все значения перечисления.

Вот результат выполнения этого кода:

```
HIGH
MEDIUM
LOW
```

Обратите внимание, как печатаются имена самих констант. Это одно из отличий работы с перечислениями, перечисления Java отличаются от static final констант.

Использование метода toString () с перечислениями

Класс перечисления автоматически получает метод toString()при компиляции. Метод toString() возвращает строковое значение для данного экземпляра перечисления.

Пример:

```
Строка levelText = Level.HIGH.toString ();
```

Значением переменной levelText после выполнения вышеуказанного оператора будет текст HIGH.

Печать перечислений

Вы можете вывести на значение перечисления обычным способом, с помощью метода println(), например таким образом:

```
System.out.println (Level.HIGH);
```

Затем будет вызван метод toString(), поэтому значение, которое будет распечатано, будет именем экземпляра перечисления - текст. Другими словами, в приведенном выше примере будет напечатан текст HIGH.

Использование метода valueOf () с перечислениями

Класс перечисление автоматически получает статический метод valueOf() в классе при компиляции. Метод valueOf() может быть

использован для получения экземпляра класса перечислимого для заданного значения String. Вот пример:

```
Level level = Level.valueOf ("BHIGH");
```

Переменная типа Level будет указывать на Level.HIGH после выполнения этой строки.

Поля перечислений

Вы можете добавлять поля данных в перечисление Джава, также как в классы. Таким образом, каждое константа перечисления получает эти поля. Значения полей должны быть переданы конструктору перечисления при определении констант. Пример приведен на листинге 4.1.

Лстинг 4.1 – Пример перечисления Level

```
public enum Level {
    HIGH(3),  //вызов конструктора со значением
3
    MEDIUM(2),     //  вызов  конструктора  со
значением 2
    LOW(1)   // вызов конструктора со значением
1
/* нужна точка с запятой, если после констант
следуют поля и методы*.
    ;
 //определение полей перечисления
    private final int levelCode;
    private Level(int levelCode) {
        this.levelCode = levelCode;
    }
}
```

Обратите внимание, как у перечисления в приведенном выше примере есть конструктор, который принимает параметр типа int. Конструктор перечисления устанавливает поле int. Когда определены постоянные значения перечисления, int значение передается конструктору перечисления. Конструктор для перечисления должен быть объявлен с модификатором private. Вы не можете использовать конструкторы с модификаторами public или protected для перечислений в Джава. Если вы

не укажете модификатор доступа в конструкторе перечисления, то он будет объявлен неявным образом с модификатором private.

Поля перечислений

Вы также можете добавлять методы в перечисление Джава. Пример приведен на листинге 4.2

Лстинг 4.2 – Пример перечисления Level с полями и методами

```
public enum Level {
  HIGH (3), // вызывает конструктор со значением
3
  MEDIUM (2),// вызывает конструктор со значением
2
  LOW (1) // вызывает конструктор со значением 1
    ; //здесь нужна точка с запятой
//поле для перечисления
  private final int levelCode;
//конструктор
  private Level (int levelCode) {
  this.levelCode = levelCode;
  }
// геттер для поля levelCode
    public int getLevelCode () {
        return this.levelCode;
    }
}
```

Вы вызываете метод перечисления Java через ссылку на одно из постоянных значений. Вот пример вызова метода перечисления Java:

```
Level level = Level.HIGH;
System.out.println (level.getLevelCode ());
```

Этот код распечатает значение, 3 которое является значением levelCode для поля HIGH константы перечисления

Кроме конструкторов, геттеров и сеттеров вы можете добавлять свои собственные методы в классы перечисления, которые будут производить вычисления на основе значений полей константы перечисления. Если ваши поля не объявлены как final, вы даже можете изменить значения полей (хотя

это может быть не очень хорошая идея, учитывая, что перечисления должны быть константами).

Задания на практическую работу №4

Задание 1. Времена года

Создать перечисление, содержащее названия времен года.

1) Создать переменную, содержащую ваше любимое время года и распечатать всю информацию о нем.
2) Создать метод, который принимает на вход переменную созданного вами enum типа. Если значение равно Лето, выводим на консоль “Я люблю лето” и так далее. Используем оператор switch.
3) Перечисление должно содержать переменную, содержащую среднюю температуру в каждом времени года.
4) Добавить конструктор, принимающий на вход среднюю температуру.
5) Создать метод getDescription, возвращающий строку “Холодное время года”. Переопределить метод getDescription - для константы Лето метод должен возвращать “Теплое время года”.
6) В цикле распечатать все времена года, среднюю температуру и описание времени года

Задание 2. Ателье

Создать перечисление, содержащее размеры одежды (XXS, XS, S, M, L). Перечисление содержит метод getDescription, возвращающий строку “Взрослый размер”. Переопределить метод getDescription - для константы XXS метод должен возвращать строку “Детский размер”. Также перечисление должно содержать числовое значение euroSize(32, 34, 36, 38, 40), соответствующее каждому размеру. Создать конструктор, принимающий на вход euroSize.

1) Создать интерфейсы MenClothing (мужская одежда) с методом dressMan() (одеть мужчину) и WomenClothing (женская одежда) с методом dressWomen() (одеть женщину).

2) Создать абстрактный класс Clothes (одежда), содержащий в качестве переменных класса - размер одежды, стоимость, цвет.
3) Создать классы наследники класса Clothes – класс TShirt (футболка) (реализует интерфейсы MenClothing и WomenClothing), класс Pants (штаны) (реализует интерфейсы MenClothing и WomenClothing), класс Skirt (реализует интерфейсы WomenClothing), класс Tie (галстук) (реализует интерфейсы MenClothing).
4) Создать массив, содержащий все типы одежды. Создать класс Atelier (Ателье), содержащий методы dressWomen, dressMan, на вход которых будет поступать массив, содержащий все типы одежды (подумайте какой тип будет у массива). Переопределите метод dressWomen() для вывода на консоль всей информации о женской одежде. То же самое сделайте для метода dressMan().

Задание 3. Интернет-магазин

Создать мини приложение - интернет-магазин. Должны быть реализованы следующие возможности:

1) Аутентификация пользователя. Пользователь вводит логин и пароль с клавиатуры.
2) Просмотр списка каталогов товаров.
3) Просмотр списка товаров определенного каталога.
4) Выбор товара в корзину.
5) Покупка товаров, находящихся в корзине.

Для выполнения заданий необходимо создать перечисление согласно заданию, можете добавить свои операции или изменить что-то по своему усмотрению.

Задание 4

Создать класс, описывающий сущность компьютер (Computer). Для описания составных частей компьютера использовать отдельные классы (Processor, Memory, Monitor). Описать необходимые свойства и методы для каждого класса. Для названий марок компьютера используйте перечисления (enum)