|  |
| --- |
| **ТЕХНОЛОГИИ ПРОГРАММИРОВАНИЯ** |
|  |
|  |
|  |
| **Практическое занятие №8.**  **Тема: Проектирование и использование перечислений в Java-программах** |
|  |
|  |
|  |

**Цель:** получение практических навыков проектирования перечислений и их использования в Java-программах.

**Методические рекомендации**

**Перечисления. Общие сведения**

Перечисления создаются с помощью ключевого слова **enum**. В качестве примера ниже показано простое перечисление цветов.

**enum Color {**

**Red, Yellow, Black, Green, White**

**}**

Идентификаторы **Red, Yellow** и так далее называются константами типа «перечисление». Каждая из них явно объявлена как открытый статический конечный член класса **Color**. Более того, они относятся к типу того перечисления, в котором объявлены (в данном случае - **Color**). В языке **Java** такие константы называются самотипизированными.

Объявив перечисление, можно создавать переменные данного типа. Но, несмотря на то, что перечисления определяют тип класса, получать экземпляры класса типа **enum** с помощью оператора **new** нельзя. Вместо этого переменная перечисления объявляется и применяется практически так же, как и переменные примитивных типов.

В приведенном ниже примере объявляется переменная **col** перечислимого типа **Color**.

**Color col**;

Переменная **col** относится к типу **Color** , и поэтому ей можно присвоить только те значения, которые определены в перечислении. В следующем примере переменной **col** присваивается значение **Black**:

**col = Color.Black;**

Две константы перечислимого типа можно проверять на равенство с помощью операции отношения **==**. Например, в следующем условном операторе переменная **col** сравнивается с константой **Color. Black**:

**if(col == Color. Black) …**

Значения перечислимого типа можно также использовать в управляющем операторе **switch**. Разумеется, для этого во всех выражениях ветвей **case** должны использоваться константы из того же самого перечисления, что и в самом операторе **switch**. Например, следующий оператор **switch** составлен совершенно правильно:

**switch(col ) {**

**case Red:**

**// . . .**

**Case Yellow:**

**// . . .**

В выражениях ветвей **case** имена констант указываются без уточнения имени их перечислимого типа, например **Black** , а не **Color. Black**. Дело в том, что тип перечисления в операторе **switсh** уже неявно задает тип перечисления для выражений в ветвях **case**.

Когда выводится константа перечислимого типа, например, методом **println( )**, то отображается ее имя. Например, в следующей строке кода:

**System.out.println**(**Color.Black**)

Выводится имя **Black**.

Приведенный ниже пример программы подытоживает все сказанное выше, демонстрируя применение перечисления **Color**.

**enum Color {**

**Red, Yellow, Black, Green, White**

**}**

**public class JavaEnum {**

**public static void main(String[] args) {**

**Color col;**

**col = Color.Red;**

//вывести значение перечислимого типа

**System.out. println( "Знaчeниe col: " + col);**

**col = Color.Yellow;**

// сравнить два значения перечислимого типа

**if(col == Color.Yellow)**

**System.out.println("Пepeмeннaя col содержит Yellow.");**

**switch(col){**

**case Red:**

**System.out.println("Цвет красный.");**

**break;**

**case Yellow:**

**System.out.println("Цвет желтый.");**

**break;**

**case Black:**

**System.out.println("Цвет чёрный.");**

**break;**

**case Green:**

**System.out.println("Цвет зелёный.");**

**break;**

**case White:**

**System.out.println("Цвет белый.");**

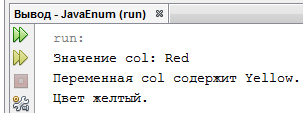
**break;**

**}**

**}**

**}**

Эта программа выводит следующий результат:



Как пояснялось ранее, перечисление в **Java** относится к типу класса. Создать экземпляр перечисления с помощью оператора **new** нельзя, но в остальном перечисление обладает всеми возможностями, которые имеются у других классов. Тот факт, что перечисление определяет класс, наделяет перечисление огромным потенциалом, которого лишены перечисления в других языках программирования. В частности, перечисления допускают применение конструкторов, добавление переменных экземпляров и методов и даже реализацию интерфейсов.

Важно понимать, что ***константа перечислимого типа является объектом класса своего перечисления***. Так, если для перечисления определяется конструктор, он вызывается всякий раз, когда создается константа перечислимого типа. Кроме того, у каждой константы перечислимого типа имеется своя копия любой из переменных экземпляра, объявленных в перечислении.

**Методы и переменные перечисления и экземпляра**

Перечисления автоматически включают в себя два предопределенных метода: **values() и valueOf().** Их общая форма:

**public static тип\_перечисления[] values()**

**public static тип\_перечисления valueOf(String строка).**

Метод **values()**возвращает массив, содержащий список констант перечисления. А метод **valueOf( )** возвращает константу перечисления, значение которой соответствует символьной строке, переданной в качестве аргумента **строка**. В обоих случаях **тип\_перечисления** обозначает тип конкретного перечисления.

Так, если вызвать метод **Color.valuesOf("** **Green** **")** для упоминавшегося ранее перечисления **Color**, то он возвратит значение константы перечислимого типа **Green**.

Пример.

**enum Apple{**

**Jonathan(l0), GoldenDel(9),**

**RedDel(l2),Winesap(15), Cortland(8);**

**private int price;** // цена яблока каждого сорта

// Конструктор

**Apple(int р) {price = р;}**

**int getPrice () {return price;}**

**}**

**class JavaEnum {**

**public static void main (String[] args) {**

**Apple ар;**

// вывести цену на яблоко сорта Winesap

**System.out. println("Яблoкo сорта Winesap стоит " + Apple.Winesap.getPrice() + "центов.");**

// применение метода valuesOf()

**ар = Apple.valueOf("Cortland");**

**System.out.println("Пepeмeннaя ар содержит" + ар);**

// вывести цены на все сорта яблок

**System.out.println("Цeны на все сорта яблок:");**

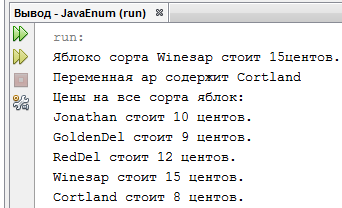
**for(Apple а:Apple.values())**

**System.out.println(a+"стоит"+a.getPrice()+"центов.");**

**}**

**}**

Эта программа выводит следующий результат:

****

В данной версии перечисления **Apple** добавлено следующее.

Во-первых, переменная экземпляра **price**, которая служит для хранения цены яблока каждого сорта. Во-вторых, конструктор **Apple**( ), которому передается цена на яблоко. И в-третьих, метод **getPrice()**, возвращающий значение цены.

Когда в методе **main**( ) объявляется переменная **ар**, конструктор **Apple**( ) вызывается один раз для каждой объявленной константы. Аргументы конструктору передаются в скобках после каждой перечисляемой константы:

**Jonathan(l0), GoldenDel(9),**

**RedDel(l2),Winesap(15),Cortland(8);**

У каждой константы перечислимого типа имеется своя копия переменной экземпляра **price**, поэтому для получения цены на определенный сорт яблок достаточно вызвать метод **getPrice()**. Например, цена на сорт яблок **Winesap** получается в результате следующего вызова в методе **main**( ): **Apple.Winesap.getPrice().**

Цены на все сорта яблок получаются при переборе перечисления в цикле **for**.

Также можно определять методы для отдельных констант, например (переопределяется абстрактный метод **action()**):

**enum Operation{**

**SUM{**

**public int action(int x, int y){ return x + y;}**

**},**

**SUBTRACT{**

**public int action(int x, int y){ return x - y;}**

**},**

**MULTIPLY{**

**public int action(int x, int y){ return x \* y;}**

**};**

**public abstract int action(int x, int y);**

}

**class JavaEnum {**

**public static void main(String[] args) {**

**Operation op = Operation.SUM;**

**System.out.println("Сумма= "+ op.action(10, 4));** // 14

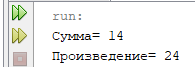
**op = Operation.MULTIPLY;**

**System.out.println("Произведение="+op.action(6, 4));**// 24

**}**

**}**

Вид экрана:

****

При использовании перечисления программисту запрещается применять наследование. Однако все перечисления автоматически наследуют переменные и методы от класса **Enum**. В этом классе определен ряд методов, доступных для использования всеми перечислениями. Вот два из них: **ordinal()** и **compareТо().**

Метод **ordinal()**возвращает порядковое значение вызывающей константы. Отсчет порядковых значений начинается с нуля. Ниже приведена общая форма объявления метода **ordinal().**

**final int** **ordinal()**

Для сравнения порядковых значений двух констант в одном и том же перечислении можно воспользоваться методом **compareТо()**. Ниже приведена общая форма объявления этого метода.

**final int compareТо(перечислимый\_тип е).**

Здесь в качестве параметра **е** задается константа, сравниваемая с вызывающей константой, а перед ней указывается *перечислимый\_тип* , к которому эта константа относится.

Ниже приведен пример программы, демонстрирующий применение данных методов.

// Перечисление разных видов транспорта

**enum Transport {**

**CAR, TRUCK, AIRPLANE, TRAIN, BOAT**

**}**

**class EnumPrym {**

**public static void main(String args[]) {**

**Transport tp, tp2, tp3;**

**System.out.println("Неге are all Transport**

**constants and their ordinal values: ");**

// Получение порядковых значений констант

**for(Transport t : Transport.values())**

**System.out.println(t + " " + t.ordinal());**

**tp = Transport.AIRPLANE;**

**tp2 = Transport.TRAIN;**

**tp3 = Transport.AIRPLANE;**

**System.out.println ();**

// Сравнение порядковых значений констант

**if(tp.сотрагеТо(tp2) < 0)**

**System.out.println(tp + " comes before " + tp2);**

**if(tp.compareTo(tp2) > 0)**

**System.out.println(tp2 + " comes before " + tp);**

**if(tp.compareTo(tp3) == 0)**

**System.out.println(tp + " equals " + tp3);**

**}**

**}**

На экране:

**Неге are all Transport constants and their ordinal values:**

**CAR 0**

**TRUCK 1**

**AIRPLANE 2**

**TRAIN 3**

**BOAT 4**

**AIRPLANE comes before TRAIN**

**AIRPLANE equals AIRPLANE**

**Практикум**

**Задание.** Спроектировать перечисление (**enum**) согласно индивидуальному варианту (см. приложение). Создать по три объекта каждого типа, задавая их характеристики случайным образом. Рассчитать вычисляемый показатель и вывести на экран характеристики экземпляра, которому он принадлежит.

Отчет оформляется по общеустановленным правилам в электронном виде со следующим содержанием:

1. титульный лист
2. тема и цель занятия,
3. задание на практическое занятие,
4. текст программы с комментариями,
5. результаты работы программы (вид экрана) и
6. выводы с анализом использования средств Java в приложении.

ПРИЛОЖЕНИЕ

**Варианты заданий**

|  |  |  |  |
| --- | --- | --- | --- |
| **№**  **п/п** | **Перечисление** | **Типы объектов (констант)** | **Вычисляемый показатель** |
|  | Компьютер | Настольный компьютер, ноутбук | Экземпляр с наибольшей оперативной памятью |
|  | Товар | Посуда, продукты питания | Сумма покупки |
|  | Грузовое транспортное средство | Самосвал, трейлер | Максимальный пробег на полном бензобаке |
|  | Представитель университета | Преподаватель, студент | Количество студентов, обучающихся у конкретного преподавателя |
|  | Транспортное средство | Легковой автомобиль, грузовой автомобиль | Максимальная емкость бензобака |
|  | Компьютер | Настольный компьютер, ноутбук | Самый дешевый компьютер |
|  | Программный продукт | Операционная система, текстовый редактор | Последняя версия |
|  | Документ | Паспорт, студенческий билет | Количество документов на заданную фамилию |
|  | Периферийное устройство компьютера | Принтер, монитор | Минимальная цена устройства |
|  | Строительный товар | Сыпучие материалы, инструмент | Сумма покупки |
|  | Страховой полис | Полис обязательного медицинского страхования, страхования жилища | Количество полисов на заданную фамилию |
|  | Периферийное устройство компьютера | Клавиатура, сканер | Средняя цена |
|  | Недвижимость | Таунхаус, квартира в многоквартирном доме | Максимальная жилая площадь |
|  | Часы | Электронные часы, механические часы | Самый дорогой экземпляр |
|  | Товар | Телевизор, холодильник | Количество товаров заданной фирмы |
|  | Магнитная карта для проезда на транспорте | Карта общего назначения для проезда в метро, льготная транспортная карта учащегося | Количество карт без поездок |
|  | Транспортное средство | Легковой автомобиль, грузовой автомобиль | Количество транспортных средств, выпущенных после заданного года |
|  | Канцелярские товары | Бумага, авторучка | Количество товаров заданной фирмы |
|  | Программный продукт | Операционная система, текстовый редактор | Количество программ, выпущенных заданной фирмой |
|  | Документ | Паспорт, студенческий билет | Документ, выданный раньше всех других |
|  | Путешествие (тур) | Морской круиз, отдых на море | Самый дешевый тур на 7 и более дней |
|  | Товар | Одежда, обувь | Самый дорогой товар |
|  | Страховой полис | Полис обязательного медицинского страхования, страхования жилища | Полис с максимальным сроком действия |
|  | СУБД | Реляционная, иерархическая | Количество СУБД заданного производителя |
|  | Предприятие малого бизнеса | Магазин, парикмахерская | Название предприятия с максимальным числом сотрудников |
|  | Представление | Театр, кино | Минимальное число зрителей в зале |
|  | Запоминающее устройство | Флэш-карта, карта памяти | Экземпляр с минимальным размером |
|  | Принтер | Струйный, лазерный | Представитель с наибольшей производительностью |
|  | Книга | Электронная, на бумаге | Количество книг одного автора |
|  | Телефон | Смартфон, кнопочный | Самая новая модель |