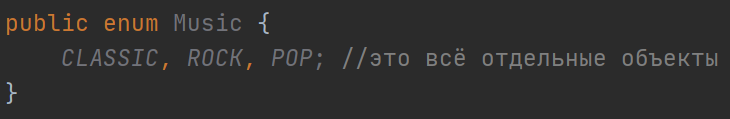
# Другой вариант рассказать об этом же

Общее: Enum – это класс. Он позволяет хранить набор констант, объединенных каким – то смыслом. У каждой константы свой номер. При создании указываем, что это не класс, а enum. В каком то смысле элементы перечисления – это объекты.

Примеры: жанры музыки, названия профессий, размеры кофе.

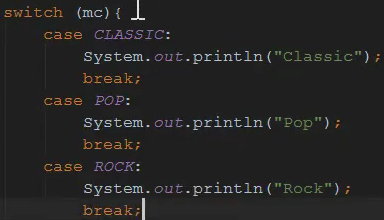




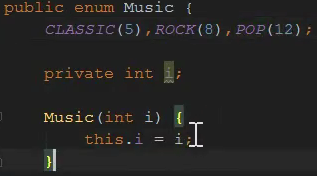




Удобно пользоваться case в сочетании с enum:



Т. к. наши элементы – это экземпляры, то они собраны пустым конструктором. Если мы добавляем элемент и создадим новый конструктор, то придется написать вот так:



Крч **каждый элемент хранит поля.** Когда мы объявляем поля и конструктор, то эти поля относятся к элементам enum. Методы тоже можно иметь :

Энам – пронумерованный список, просто в нем переменные static final (набор констант).

public class Example {

enum Coffee {SMALL, MEDIUM, LARGE};

}

Можем определить энам как в классе, так и вне класса.

Что ты можем делать:

Coffee coffee = Coffee.LARGE;

System.out.print(coffee);

В целом это тот же класс, поэтому у enum могут быть и свои поля, и конструкторы:

enum Exp {SMALL, NESMALL, BIG, MEGABIG;

int milliliters;

float tries;

Exp (int milliliters) {

this.milliliters = milliliters;

}

//методы тоже разрешены

}

Можно переопределить метод для 1 конкретного объекта:

enum Exp {SMALL, NESMALL, BIG {

String getCoffee() {

Return “C”}

, MEGABIG;

int milliliters;

}

Создание нового объекта enum выполняется без ключевого слова new:

Music mc = Music.ROCK;

Music.values() вернет массив всех Music

Music mc2 = Music.valueof(Music.class,”ROCK”);

mc.name()

mc.ordinal() – возвращает индекс значения элемента mc (считая с 0)

Если сделать в enum определенный конструктор, то элементы перечисления будут хранить в себе значение (которое мы им зададим)