<https://javarush.ru/groups/posts/1963-kak-ispoljhzovatjh-klass-enum>

enum в переводе с английского – перечисление

Версия появления - java 5

определения:

Это класс. Но он специально «заточенный» для создания некоторого ограниченного круга значений.

Это специальный тип данных, который предоставляет заранее заданный набор значений на выбор.

это Java-класс, который предоставляет ограниченный набор значений-объектов.

Рассмотрим применение enum-ов:

Представим, что у нас есть кисть, которой мы планируем нарисовать картину. Нашей кисти необходимы краски разных цветов.

Создадим соответствующий класс:

public class Brush {  
 private String color;  
  
 public Brush(String color) {  
 this.color = color;  
 }  
  
 public String getColor() {  
 return color;  
 }  
  
 public void setColor(String color) {  
 this.color = color;  
 }  
  
 @Override  
 public String toString() {  
 return "Brush{" +  
 "color='" + color + '\'' +  
 '}';  
 }  
}

Создадим точку входа (метод main) и попробуем воспользоваться нашей кистью:

public class App {  
  
 public static void main(String[] args) {  
 Brush brush = new Brush("красный");  
 System.*out*.println(brush);  
 }  
}

Вывод в консоль:

Brush{color='красный'}

Все выглядит неплохо. Но представим, что нашу кисть взял другой разработчик и “не прочел инструкции”. В итоге мы можем получить следующее:

Brush brush = new Brush("мама мыла раму");  
System.*out*.println(brush);

Вывод в консоль:

Brush{color='мама мыла раму'}

Уфы, наша кисть не умеет сразу рисовать задуманное (((

Нам надо ограничить значения, которые кисть может принимать на вход, только цветами.

Попробуем справиться без enum-a:

Воспользуемся решением, предложенным Джошуа Блох, в книге «Effective Java»

Создадим класс ColorClass, с полем color и геттером для это поля.

public class ColorClass {  
 private String color;  
  
 public ColorClass(String color) {  
 this.color = color;  
 }  
  
 public String getColor() {  
 return color;  
 }  
}

Вспомним определение enum и реализуем одно из условий : “ предоставляет заранее заданный набор значений ”

Сделать не сложно. Просто дополним класс ColorClass полями, которые будут являться публичными статическими неизменяемыми объектами этого же класса:

public static final ColorClass *RED* = new ColorClass("красный");  
public static final ColorClass *BLUE* = new ColorClass("синий");  
public static final ColorClass *GREEN* = new ColorClass("зеленый");

Отлично у нас есть три цвета, но объекты ColorClass можно создавать в ЛЮБОМ месте нашей программы. В классе App мы можем сделать например так:

ColorClass color = new ColorClass("мама мыла раму");

Что бы сделать наш набор значений действительно ограниченным, сделаем небольшую доработку. Сделаем конструктор ColorClass приватным, запретив создание объектов этого класса везде кроме самого класса.

В итоге получим класс для создания некоторого ограниченного круга значений.

public class ColorClass {  
 public static final ColorClass *RED* = new ColorClass("красный");  
 public static final ColorClass *BLUE* = new ColorClass("синий");  
 public static final ColorClass *GREEN* = new ColorClass("зеленый");  
  
 private String color;  
  
 private ColorClass(String color) {  
 this.color = color;  
 }  
  
 public String getColor() {  
 return color;  
 }  
}

Изменим класс Brush для работы с ColorClass.

Для этого необходимо только изменить коструктор:

public Brush(ColorClass color) {  
 this.color = color.getColor();  
}

Как видим IDE сама подсказывает нам список полей класса ColorClass



Brush brush = new Brush(ColorClass.*RED*);  
System.*out*.println(brush);

Вывод в консоль:

Brush{color='красный'}

Как видим мы добились своей цели и теперь, нашу кисть мы можем окунать только в краску, а не во все что попало ))

Однако разработчики Java обратили внимание на существующую проблему. Конечно, здорово, что программисты смогли придумать ее решение с помощью имеющихся в языке средств, но оно выглядит не таким уж простым! Необходимо было очевидное решение, доступное даже новичкам.

Так в Java и появился Enum.

Создадим enum ColorEnum, который будет повторять функционал класса ColorClass

public enum ColorEnum {  
 *RED*("красный"),  
 *BLUE*("синий"),  
 *GREEN*("зеленый");  
  
 private String color;  
  
 ColorEnum(String color) {  
 this.color = color;  
 }  
  
 public String getColor() {  
 return color;  
 }  
}

Посмотрим отличия:

* Для создания используется ключевое слово enum
* Не необходимости писать public static final ColorClass
* При перечислении объектов используем “,”

Только после последнего объекта используем “;”

* Конструктор по умолчанию private

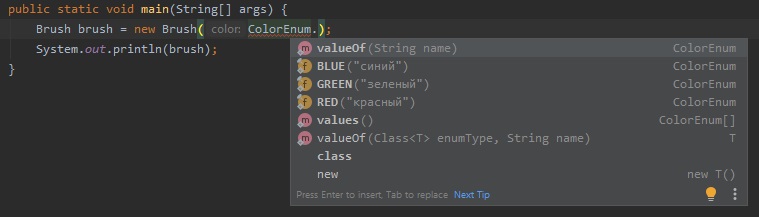
Данные изменения позволяют сообщить цель с которой создается класс(enum) и избежать написания дополнительного/очевидного кода.

Проверим как работает класс Brush с созданным ColorEnum.

Сново необходимо только изменить коструктор:

public Brush(ColorEnum color) {  
 this.color = color.getColor();  
}

IDE также подсказывает нам список полей ColorEnum



И соответственно вывод в консоль не изменился:

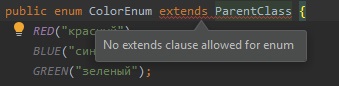
Brush brush = new Brush(ColorEnum.*RED*);  
System.*out*.println(brush);

Вывод в консоль:

Brush{color='красный'}

Напомню, что enum это в первую очередь Java-класс.

Тогда попробуем унаследовать ColorEnum от какого-либо класса:



Enum нельзя наследовать?

Можно!!!

Если вспомнить подсказки IDE при выборе цвета, то можно заметить возможность выбрать методы, которые мы не писали (valueOf(String name), …)

Откуда они?

Правильно…!!!

Любой созданный enum, неявно унаследован от java.lang.Enum

public enum ColorEnum extends Enum

А Java не поддерживает множественное наследование

Рассмотрим унаследованные методы:

* toString() – переопределен в классе Enum, и возвращает имя объекта.

ColorEnum colorEnum = ColorEnum.*BLUE*;  
System.*out*.println(colorEnum);  
System.*out*.println(colorEnum.getColor());

Вывод в консоль:

BLUE

синий

Напомню, что метод System.out.println(object) по умолчанию использует метод object.toString()

* static valueOf(String name) возвращает объект enum по его имени

ColorEnum colorEnum = ColorEnum.*valueOf*("BLUE");  
System.*out*.println(colorEnum);

Вывод в консоль:

BLUE

* статический метод valueOf(String name) - возвращает объект enum по его имени

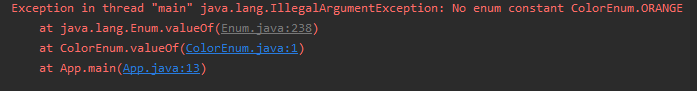
ColorEnum colorEnum = ColorEnum.*valueOf*("BLUE");  
System.*out*.println(colorEnum);

Вывод в консоль:

BLUE

Если такого имени нет вернет ошибку:

ColorEnum colorEnum = ColorEnum.valueOf("ORANGE")

Вывод в консоль:

* статический метод valueOf(Class enumType, String name) - аналогичен вышеописанному методу, при этом позволяет получить объект непосредственно из класса Enum

ColorEnum colorEnum = Enum.*valueOf*(ColorEnum.class, "BLUE");  
System.*out*.println(colorEnum);

Вывод в консоль:

BLUE

* статический метод values() – возвращает массив всех объектов перечисленных в enum

ColorEnum[] colorEnums = ColorEnum.*values*();  
for (int i = 0; i < colorEnums.length; i++) {  
 System.*out*.println(colorEnums[i]);  
}

Вывод в консоль:

RED

BLUE

GREEN

* метод name() – возвращает имя объекта

ColorEnum colorEnum = ColorEnum.*BLUE*;  
System.*out*.println(colorEnum.name());

Вывод в консоль:

BLUE

* метод ordinal() – возвращает номер объекта в порядке их добавления начиная с 0

ColorEnum colorEnum = ColorEnum.*BLUE*;  
System.*out*.println(colorEnum.ordinal());

Вывод в консоль:

1

* метод int compareTo(EnumObject object) – возвращает разницу между порядковыми номерами объектов enum

ColorEnum red = ColorEnum.*RED*;  
ColorEnum green = ColorEnum.*GREEN*;  
System.*out*.println(red.compareTo(green));

Вывод в консоль:

-2

ДЗ:

Написать enum направлений (назад, влево и тд)

Что должен содержать enum:

1. 1 Поле с русским названием направления:
2. Абстрактный метод message(), который будет переопределен к каждом экземпляре(константе). Метод должен печать в консоль, что-нибудь в духе русских сказок. Например: «Направо пойдешь – счастье найдешь»

Создать аналогичный класс (с приватным конструктором и статическими константами в виде констант экземпляров самого себя) дублирующий реализацию enum.

Написать класс с названием Utils, который будет содержать два прегруженых статических метода:

* Public static void print(String text)
* Public static void print(String text, Color color)