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

<https://javarush.ru/groups/posts/1419-perechislenija-v-java-java-enum>

**Enum**

**Основная информация:**

Enum в переводе с английского – перечисление

Версия появления - java 5

**Определения:**

* - Java-класс, специально «заточенный» для создания некоторого ограниченного круга значений.
* - специальный тип данных, который предоставляет заранее заданный набор значений на выбор.
* - Java-класс, который предоставляет ограниченный набор значений-объектов.

**Enum-ы в реальной жизни:**



**Применение enum в java:**

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 класс App с методом 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'}

Боюсь такой цвет не входит в нашу палитру (((. Необходимо ограничить только цветами, значения, которые кисть может принимать на вход. В этом нам как раз и поможет Enum

**Реализация с помощью класса:**

Представим, что в нашем распоряжении нет такого инструмента как enum и попробуем справиться обычным java классом. 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 это не сложно. Просто дополним класс ColorClass полями, которые будут 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 наш набор значений был действительно ограниченным, необходима небольшая доработка. 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 конструктор:

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

Вернемся в класс App и задействуем новую реализацию. IDE сама подсказывает нам список полей класса ColorClass



Выберем красный цвет.

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.

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.

**Реализация с помощью enum:**

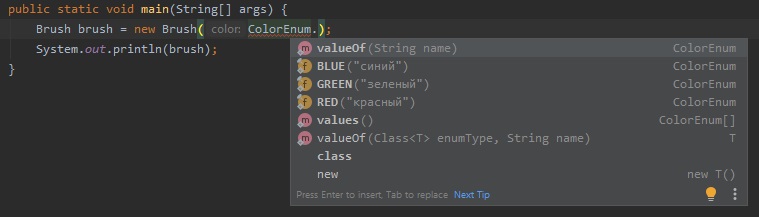
Создадим enum ColorEnum, который будет повторять функционал класса ColorClass. Использует ключевое слово enum для его объявления. Объекты перечисляются через “,” перечисление заканчивается “;”. Конструктор по умолчанию private. В остальном это тот же, созданный ранее класс.

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

Проверим как работает класс Brush с созданным ColorEnum. Изменим тип аргумента конструктора на ColorEnum

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

IDE также подсказывает нам список полей ColorEnum

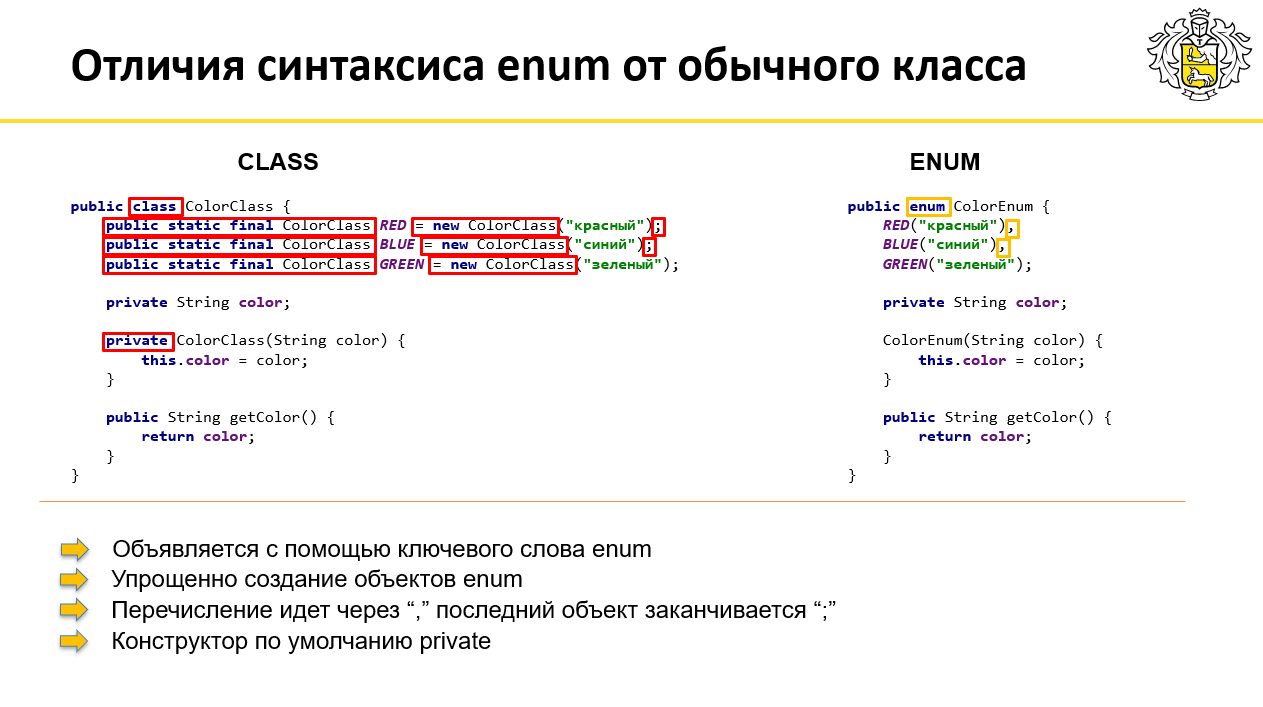


И соответственно вывод в консоль не изменился:

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

Вывод в консоль:

Brush{color='красный'}



**Методы enum:**

Помимо лаконичного синтаксиса, о котором говорилось ранее, есть и другие преимущества.

Любой созданный enum, неявно унаследован от java.lang.Enum. Данный класс имеет ряд методов расширяющие функционал enum-oв.

Рассмотрим унаследованные методы:

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

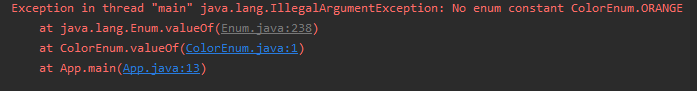
ColorEnum colorEnum = ColorEnum.*valueOf*("BLUE");  
System.*out*.println(colorEnum);

Вывод в консоль:

BLUE

Если такого имени нет вернет ошибку:

ColorEnum colorEnum = ColorEnum.valueOf("ORANGE")

Вывод в консоль:

* статический метод 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

**Подробнее о enum:**

Рассмотрим еще пару примеров. Создадим самый простой enum.

public enum ColorSimple {  
 *RED*,  
 *YELLOW*,  
 *GREEN*}

Обычно такие реализации используются, когда для нам достаточно только наименований. Для их использования специально есть метод name(). Так же данная реализация полезна, если enum используется для выбора одного из вариантов конструкции switch case:

switch (color) {  
 case *RED*:  
 System.*out*.println("Остановка");  
 break;  
 case *GREEN*:  
 System.*out*.println("Поехали");  
 break;  
 case *YELLOW*:  
 System.*out*.println("Внимание");  
}

Enum может содержать абстрактные методы. Но тогда они должны быть реализованы в объектах.

Пример:

public enum Direction {  
 *LEFT*{  
 @Override  
 void message() {  
 System.*out*.println("Налево пойдешь - коня потеряешь");  
 }  
 },  
 *RIGHT*{  
 @Override  
 void message() {  
 System.*out*.println("Направо пойдешь - жизнь потеряешь");  
 }  
 },  
 *STRAIGHT*{  
 @Override  
 void message() {  
 System.*out*.println("Прямо пойдешь – счастье найдешь (или жив будешь, да себя позабудешь).");  
 }  
 };  
  
 abstract void message();  
}

Как видим, при желании мы можем так же переопределить @Override уже реализованные методы о одном или нескольких объектах.

Дополнительные статьи:

* <https://javarush.ru/groups/posts/2260-enum-prakticheskie-primerih-dobavlenie-konstruktorov-i-metodov>
* <https://javarush.ru/groups/posts/1419-perechislenija-v-java-java-enum>

ДЗ:

1. Создать все сущности описанные в лекции.
2. Создать enum Color с цветами о которых будет написано ниже.

Создать класс Utils, который будет содержать два перегруженных статических метода:

* public static void print(String text) – должен просто выводить текст в консоль
* public static void print(String text, Color color) – выводить в консоль текст окрашенный в переданный цвет

Данный код позволит реализовать изменение цвета в консоли:

public class Main {

public static final String ANSI\_RESET = "\u001B[0m";

public static final String ANSI\_BLACK = "\u001B[30m";

public static final String ANSI\_RED = "\u001B[31m";

public static final String ANSI\_GREEN = "\u001B[32m";

public static final String ANSI\_YELLOW = "\u001B[33m";

public static final String ANSI\_BLUE = "\u001B[34m";

public static final String ANSI\_PURPLE = "\u001B[35m";

public static final String ANSI\_CYAN = "\u001B[36m";

public static final String ANSI\_WHITE = "\u001B[37m";

public static void main(String[] args) {

System.out.println(ANSI\_RED + "This text is red!" + ANSI\_RESET);

}

}

1. Дополнить enum Color статическим методом, который будет возвращать случайный объект enum Color.

Как получить случайное число:

<https://javarush.ru/groups/posts/1256-generacija-sluchaynogo-chisla-v-zadannom-diapazone>