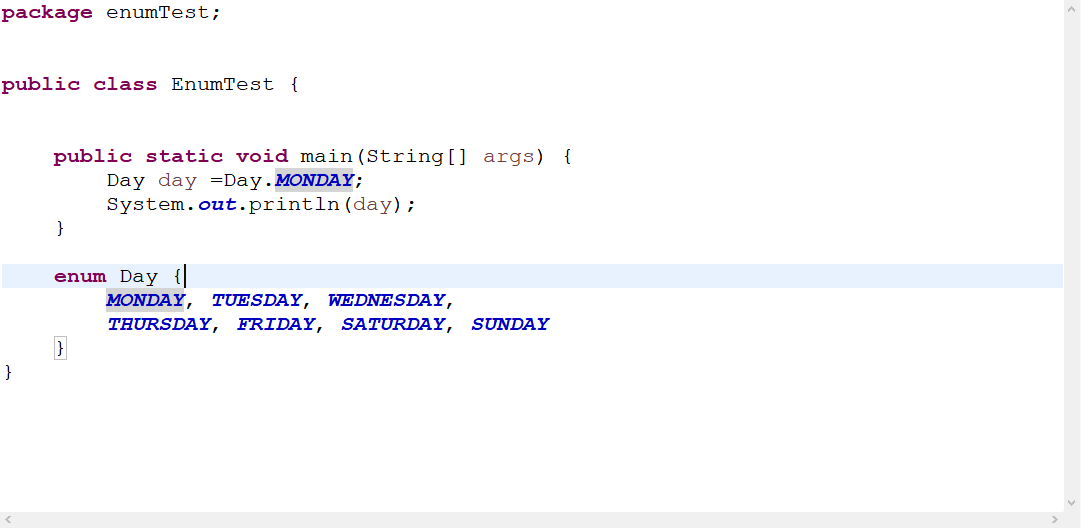
# 枚举的最基本使用

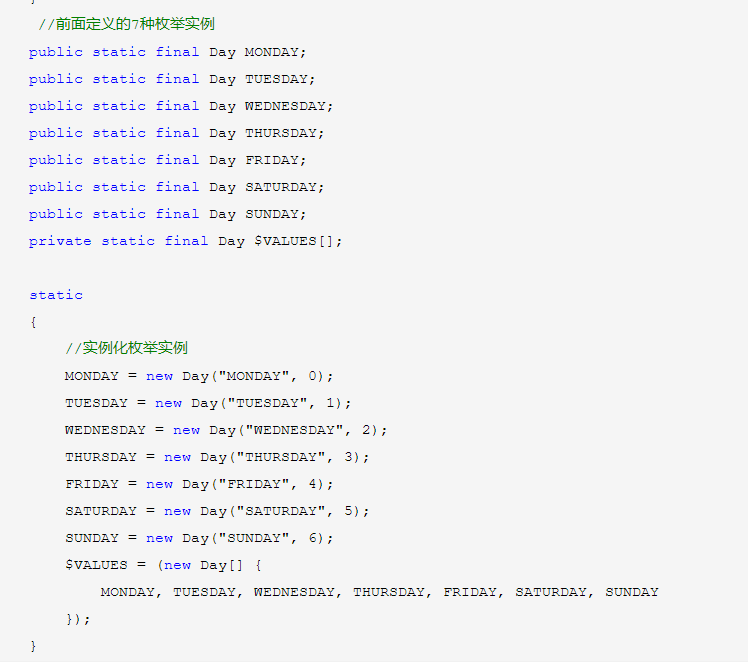


enum关键字类似于class关键字，一个是类类型，一个是枚举类型。

Day.MONDAY，就可以直接获取到这个值，返回的是Day类型的数据，值为MONDAY。

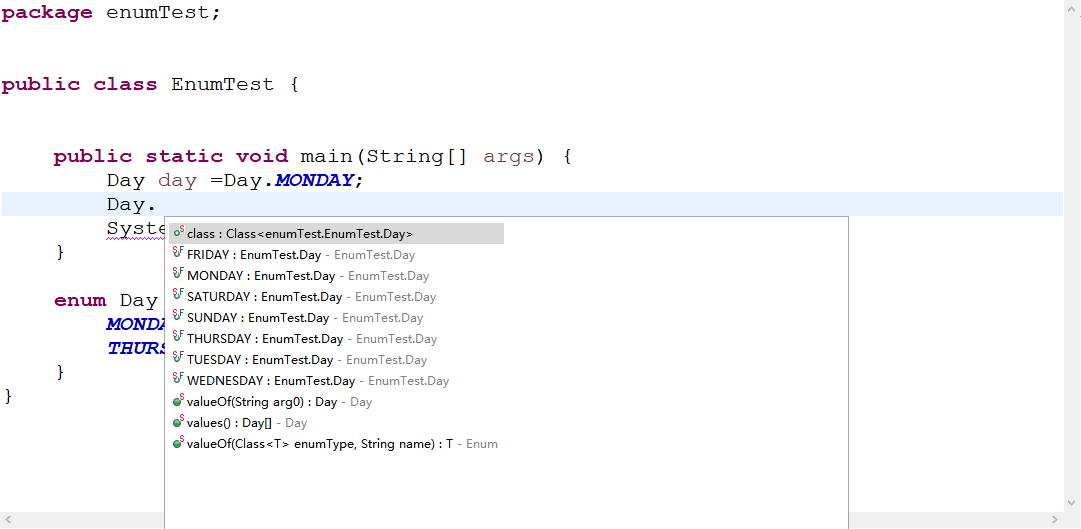
为什么返回值为Day类型？

在使用enum创建枚举类型编译后，编译器会为我们生成一个相关的类，这个类继承java.lang.Enum类。所以，枚举类型编译后其实也是一个类，这个类是final修饰的，不能被其他类继承，并且这个类继承java.lang.Enum类，这个Enum类是抽象类。



从这里就能看出来，枚举中的每一个值都是Day类型的，所以返回值都是Day类型的。

枚举类型编译后，除了创建枚举值的对象，还做了哪些工作？



1. 编译器为这个类添加静态的values()方法，
2. 编译器还为这个类添加静态valueOf()方法，这个方法间接的调用了Enum类的valueOf方法。

# Enum



这是一个抽象的类。

# EnumSet和EnumMap

|  |
| --- |
| package enumTest;  import java.util.Arrays;  import java.util.EnumMap;  import java.util.EnumSet;  import java.util.Iterator;  import java.util.Map.Entry;  public class EnumTest {    public static void main(String[] args) {  Day day =Day.MONDAY;  System.out.println(day);  System.out.println("=======");  Day[] values = Day.values();  Day valueOf = Day.valueOf("MONDAY");  System.out.println("values:"+Arrays.toString(values));  System.out.println("valueOf:"+valueOf);  System.out.println("=======EnumSet=======");  EnumSet<Day> weekSet = EnumSet.allOf(Day.class);  for (Day day2 : weekSet) {  System.out.println(day2);  }    System.out.println("=======weekmap=======");  EnumMap<Day, String> weekMap = new EnumMap<Day, String>(Day.class);  weekMap.put(Day.MONDAY, "星期一");  weekMap.put(Day.TUESDAY, "星期二");  weekMap.put(Day.WEDNESDAY, "星期三");  weekMap.put(Day.THURSDAY, "星期四");  weekMap.put(Day.FRIDAY, "星期五");  weekMap.put(Day.SATURDAY, "星期六");  weekMap.put(Day.SUNDAY, "星期天");  for(Iterator<Entry<Day, String>> iter = weekMap.entrySet().iterator(); iter.hasNext();){  Entry<Day, String> entry = iter.next();  System.out.println(entry.getKey().name()+":"+entry.getValue());  }    System.out.println("判断是否存在");    System.out.println(weekSet.contains(Day.MONDAY));//true  System.out.println(weekMap.containsKey("MONDAY"));//false  System.out.println(weekMap.containsKey(Day.MONDAY));//true  }    enum Day {  MONDAY, TUESDAY, WEDNESDAY,  THURSDAY, FRIDAY, SATURDAY, SUNDAY  }  } |

# 自定义模仿枚举

|  |
| --- |
| package com.thunisoft.jy.data.common;  import java.util.ArrayList;  import java.util.HashMap;  import java.util.List;  public class StringEnum {  private HashMap<String, String> values = new HashMap<String, String>();  private List<String> list=new ArrayList<String>();  public StringEnum(String val) {  String[] vals = val.split(",");  for (String value : vals) {  values.put(value, value);  list.add(value);  }    }  public boolean contains(String value) {  if(value==null) {  return false;  }  return values.containsKey(value);  }  /\*\*  \* 获取所有的值  \* @return  \*/  public List<String> getValues(){  return list;  }  } |

# 枚举的好处以及与常量类的区别

使用常量类时，往往得通过equals判断两者是否相等，使用枚举的话由于常量池地址唯一，可以直接用==直接对比，性能提高。

枚举类编译后为final calss，不允许继承，可以防止被子类修改的可能性。常量类可以被继承，修改增加字段容易导致父类不兼容。