Enumerated Types

کلیدواژه enum این امکان را فراهم می آورد که نوع جدیدی را با مجموعه ای محدود از مقادیر نام گذاری شده ایجاد کنید و با این مقادیر همانند اجزای منظم و باقاعده برنام رفتار کنید و این امکان بسیار کاربردی و مفید است.

ویژگی های اصلی enum

- می توان در طول لیست ثوابت موجود در enum از طریق فراخوانی values() گام برداشت.
- هنگامی که یک enum را ایجاد می کنید،یک کلاس مرتبط توسط کامپایلر برای شما ایجاد می شود.این کلاس به صورت خودکار از java.lang.Enumارث می برد و قابلیت های خاصی را ایجاد می کند که می توانید در مثال زیر ببینید:

```
import static util.Print.*;
enum Shrubbery { GROUND, CRAWLING, HANGING }
public class EnumClass{
    public static void main(String[] args){
    for(Shrubbery s : Shrubbery.values()){
        print(s + " ordinal: " + s.ordinal());
        printnb(s.compareTo(Shrubbery.CRAWLING) + " ");
        printnb(s.equals(Shrubbery.CRAWLING) + " ");
        print(s == Shrubbery.CRAWLING);
        print(s.getDeclaringClass());
        print(s.name());
        print("----------------------");
    }
  for(String s : "HANGING CRAWLING GROUND".split(" ")){
        Shrubbery shrub = Enum.valueOf(Shrubbery.class, s);
        print(shrub);
    }}
}
```

ویژگی های اصلی **enum**

- متد ordinal() یک مقدار int را تولید می کند که ترتیب اعلام هر نمونه enum را که از صفر شروع می شود،نشان می دهد.
- می توان با خیال راحت نمونه های enum را از طریق == ، متد equals() و متد hashCode( ) که به صورت خودکار ایجاد شده اند،مقایسه نمود.
- کلاس Enum قیاس پذیر (Comparable) است بنابراین متدی با نام compareTo() موجود است.همچنین این کلاس ترتیب پذیر (Serializable) است.
- ما تعیین کرده ایم که همه enumها java.lang.Enum را extend کنند.به دلیل اینکه جاوا از ارث بری چندگانه پشتیبانی نمی کند،نمی توان یک enum را از طریق ارث بری ایجاد کرد.

اضافه کردن متدها به یک enum

- جدا از این واقعیت که نمی توانیم از آن ارث ببریم،یک enum می توانند بسیار شبیه به یک کلاس منظم و با قاعده رفتار کند.

```
public enum UserStatus {
  PENDING("P"), ACTIVE("A"), INACTIVE("I"), DELETED("D");

private String statusCode;
  private UserStatus(String s) {
     statusCode = s;
  }
  public String getStatusCode() {
    return statusCode;
  }
}
```

استفاده از EnumMap

- یک EnumMap نوع خاصی از Map است که نیاز دارد که keyهای آن از یک enum مجزا باشد.

```
interface Command { void action(); }
public class EnumMaps
{
    public static void main(String[] args){
        EnumMap<AlarmPoints,Command> em =
            new EnumMap<AlarmPoints,Command>(AlarmPoints.class);
        em.put(KITCHEN, new Command()
        {
            public void action()
            {
                print("Kitchen fire!");
            }});
        em.put(BATHROOM, new Command(){
            public void action(){
                print("Bathroom alert!");
            }});
        for(Map.Entry<AlarmPoints,Command> e : em.entrySet()){
            printnb(e.getKey() + ": ");
            e.getValue().action();
        }
        try{
            em.get(UTILITY).action();
        }catch(Exception e){
            print(e);
        }
  }
```

متدهای Constant-specific

- enum های موجود در جاوا دارای ویژگی های بسیار جالبی هستند که به شما این امکان را می دهد که به هر نمونهenum نوع رفتاری متفاوتی را از طریق ایجاد متدهایی برای هر کدام آن ها،دهید .

```
import java.util.*;
import java.text.*;
public enum ConstantSpecificMethod{
    DATE_TIME {
        String getInfo(){
            return
            DateFormat.getDateInstance().format(new Date());
        }
    },
    CLASSPATH {
        String getInfo(){
            return System.getenv("CLASSPATH");
        }
    },
    VERSION {
        String getInfo(){
            return System.getProperty("java.version");
        }
    };
    abstract String getInfo();
    public static void main(String[] args){
        for(ConstantSpecificMethod csm : values())
            System.out.println(csm.getInfo());
    }}
```