Fpt University

FPT POLYTECHNIC

LẬP TRÌNH JAVA

Bài 5: Enumeration, Autoboxing, Static Import và Annotations.

www.poly.edu.vn

Điểm danh

Nhắc lại bài trước

- Khái niệm multitasking và multithreading
- Khái niệm ‘thread’ – luồng
- Thread hiện thời
- Các trạng thái của thread
- Khởi tạo thread
- Quản lý thread

Nội dung bài học

- Enumerations
- Autoboxing
- Static Import
- Annotations

Enumeration

Enumeration là kiểu dữ liệu chứa một tập các **hằng số**.

Các giá trị của enumeration được mặc định là ***static*** và ***final***.

Enumeration

Để tạo enumeration, sử dụng từ khóa **enum**

```
enum Transport{
     CAR, TRUCK, AIRPLANE, TRAIN, BOAT
}
```

CAR, TRUCK, … được gọi là các hằng số liệt kê.

Enumeration

- Khai báo:

```
Transport tp;
```

- Trong phép gán

```
tp = Transport.CAR;
```

- Trong phép so sánh

```
if (tp==Transport.TRAIN)
```

- Trong câu lệnh switch

```
switch(t){
    case CAR: //
    case TRUCK:    //
```

Enumeration

- Phương thức values()
  - Lấy ra một mảng các hằng số

```
Transport tps[] = Transport.values();
```

- Phương thức valueOf(const)
  - Lấy ra hằng số có giá trị là const

```
tp = Transport.valueOf("TRAIN");
```

Enumeration

```
enum Transport {
    CAR(65), TRUCK(55), AIRPLANE(600),
    TRAIN(70), BOAT(22);

    private int speed;

    Transport(int s) {
        speed = s;
    }

    int getSpeed() {
        return speed;
    }
}
```

Thuộc tính

Phương thức khởi tạo

Phương thức get

Enumeration

- Phương thức **ordinal()**

  Cho biết vị trí của hằng số trong enum.

  Vị trí đầu tiên được bắt đầu là 0.

- Phương thức **compareTo(contst)**

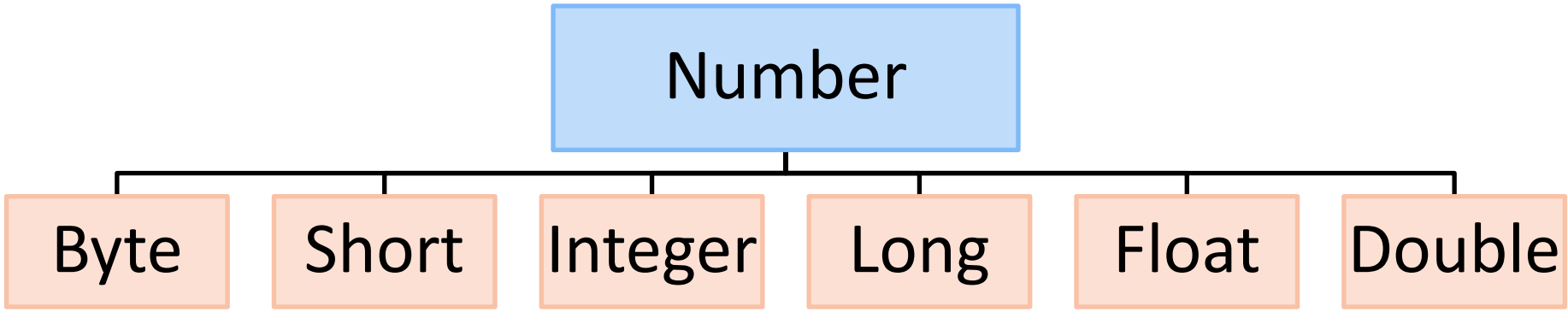
  Cho biết hằng số hiện tại là đứng trước hay đứng sau hay đúng là vị trí của một hằng số cho trước.

Autoboxing

- Autoboxing: Tự động chuyển đổi các kiểu dữ liệu nguyên thủy thành kiểu đối tượng.
- Các kiểu dữ liệu nguyên thủy không phải là đối tượng (object) nên có một số hạn chế trong khi thao tác và làm giảm hiệu suất làm việc.
- Chỉ có object thì mới được truyền theo kiểu tham chiếu vào method.

Autoboxing

Tất cả các class bao bọc (class wrapper) đều được kế thừa từ class abstract Number.



Autoboxing

Mỗi một kiểu dữ liệu nguyên thủy đều có một class wrapper tương ứng.

Primitive Types	Type Wrappers
byte	Byte
short	Short
int	Int
long	Long
float	Float
double	Double

Autoboxing

Các phương thức trả về kiểu dữ liệu nguyên thủy của các class wrapper là:

- byte byteValue( )
- double doubleValue()
- float floatValue( )
- int intValue( )
- long longValue( )
- short shortValue( )

Autoboxing

Các phương thức trả về một đối tượng là instance của class wrapper:

- Integer Integer.valueOf(int)
- Byte Byte.valueOf(byte)
- Short Short.valueOf(short)
- Float Float.valueOf(float)
- Double Double.valueOf(double)
- Long Long.valueOf(long)

Autoboxing

- Tạo đối tượng Integer với giá trị là 100:

```
Integer iobj = new Integer(100);
```

- Tạo một biến int với giá trị lấy ra từ đối tượng Integer ở trên:

```
int i = iobj.intValue();
```

Autoboxing

- Tự động chuyển kiểu trong phép gán:

```
Integer iobj = 100;
int i    = iobj;
```

- Tự động chuyển kiểu trong biểu thức

```
iobj++;
iobj += 10;
```

Autoboxing

Ưu điểm của việc sử dụng class Number:

Làm **tham số** cho các phương thức mà phương thức đó bắt buộc tham số truyền vào **phải là kiểu đối tượng**.

Có nhiều phương thức chuyển đổi giá trị kiểu nguyên thủy ⇔ kiểu đối tượng và **chuyển đổi** từ dạng **string** sang các dạng **số** (decimal, octal, hexadecimal, binary).

Autoboxing

```
List<Integer> li = new ArrayList<>();
for (int i = 1; i < 50; i += 2)
        li.add(i);
```

→ **Autoboxing**

li.add(
Integer.**valueOf**(i));

```
public static int sumEven(List<I
        int sum = 0;
        for (Integer n: li)
         if (n % 2 == 0)
                sum = sum + n;
         return sum;
}
```

→ **Unboxing**

sum = sum +
n.**intValue()**;

Static Import

- Java sử dụng từ khóa import khi muốn sử dụng các class ở **package khác**.
- Java 5 cũng đã bổ sung từ khóa '**static**' cùng với từ khóa import để thuận tiện cho việc sử dụng các phương thức static có trong các class mà không cần phải viết tên class.

Static Import

- Gói **java.lang** chứa class Math, trong class này có các phương thức tính toán số học như: căn bậc 2, tính số mũ, tính e mũ ...
- Các phương thức trên đều là ***static***
- Thông thường, muốn tính căn bậc 2 một số ta viết:

```
x = Math.sqrt(1000)
```

Static Import

- Sử dụng

```
import static java.lang.Math.sqrt;
```

thì, khi dùng chỉ cần viết:

```
x = sqrt(1000);
```

- Hoặc

```
import static java.lang.Math.*;
import static java.lang.System.out;
```

thì:

```
y = pow(10,2);
out.println(“Hello there !”);
```

Annotation

- Annotation là một dạng metadata – siêu dữ liệu được dùng để mô tả một đối tượng nào đó (như class, method).
- Mô tả dữ liệu là một tập giá trị chứa những thông tin gắn gọn, cơ bản mô tả về đối tượng đó.

Annotation

Annotation mô tả một class

```
@interface ClassTest {
    String value();
    String owner();
}
@ClassTest(value = "Class scope", owner = “Anna")
public class AnnocationForClass {
    public static void main(String[] args) {
        System.out.println("Test annotation !");
    }
}
```

Annotation

Annotation mô tả một phương thức

```
@interface MethodTest {
    String value();
    String owner();
}
...
@MethodTest(value = "Method scope", owner =“Anna”)
   public void display() {
    System.out.println(“Annotation method testing");
   }
```

Annotation

Annotation mô tả một class và field

```
@interface Hibernate_Table {
    String value();
}
@interface Hibernate_Field {
    String value();
}
public class AnnotationForClassField {
    @Hibernate_Table("Sinhvien")
    public class Student {
        @Hibernate_Field("Masinhvien")
        private int id;
        @Hibernate_Field("Hoten")
        private String name;
        @Hibernate_Field("Diem")
        private String score;
```

Annotation

Annotation	Ý nghĩa
@Deprecated	Method được đánh dấu không còn được sử dụng nữa, nó tồn tại bởi vì tính tương thích.
@Override	Thông báo cho trình biên dịch biết phương thức bên dưới là phương thức được viết lại khi kế thừa từ lớp cha.
@SuppressWarnings	Thông báo cho trình biên dịch tắt các cảnh báo.

Annotation

```
public class example {

    @Deprecated
    public void showSomething() {
        System.out.println("Method has been
                                depricated'");
    }
    public static void main(String[] args) {
        example obj = new example ();
        obj.showSomething();
    }
}
```

Annotation

```
class Child extends Parent{
     @Override
     public void hi(){
        System.out.println("Hello !");
    }
}
```

Tổng kết bài học

- Enumerations
- Autoboxing
- Static Import
- Annotations