第九讲--Java内部类、枚举和注解

任务目标

1、内部类及其类型

2、覆盖Object类中常见的方法

3、Math类的常见方法和常量

相关知识

1、方法的覆盖

2、对象的对比

3、对象数组的排序

1、内部类

1、成员内部类

```
class Person {
    int age;
    String name;
    Address add;
    Person(int a, String n)
    {
        this.age=a;
        this.name=n;
    }
    class Address
    {
        String city;
        String street;
        Address(String c, String s)
        {
            this.city =c;
            this.street =s;
        }
        public String getAddress()
        {
            return this.city + this.street;
        }
    }
    public String addressInfo()
    {
        Address ad  = new Address("ningbo","fenghua road");
        return ad.getAddress();
    }
}
```

```
public class TestPerson {
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Person p = new Person(23,"Zhang");
        System.out.print(p.addressInfo());
    }
}
```

2、局部内部类

在方法体、语句块中定义的内部类。

```
class OutClass {
private String x  ="hello";
public void make(int p)
{
final String y = "local";
class Inner
{
public void see()
{
    System.out.print(x);
    System.out.print(y);
    System.out.print(p);
}
}
new Inner().see();
}
}
public class OutClassTest {
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        OutClass clazz = new OutClass();
        clazz.make(3);
    }
}
```

3、匿名内部类

内部类是没有名字的，一般是使用一次。

```
class Dog
{
    public void eat()
    {
        System.out.print("Dog Eat");
    }
}
public class DogTest {

    public static void main(String[] args) {

        // TODO Auto-generated method stub
        Dog d = new Dog(){public void eat() {System.out.print("Dog eat
bones");}};
        d.eat();
        }
```

```
}

interface Print
{
    public abstract void print(String s);
}

public class PrintTest
{
    public static void main(String[] args)
    {
        Print p = new Print()
        {
          public void print(String m)
          {
              System.out.print(m+"message");
          }
        };
        p.print("Epson");
    }
}
```

4、静态内部类

静态内部类使用static修饰，静态内部类也称嵌套类（nested class）

```
public class MyOut
{
    private static int x =100;
    public static class MyInner
    {
        private String y = "hello";
        public void inner()
        {
            System.out.println(x);
            System.out.println(y);
        }
    }
    public static void main(String[] args)
    {
        MyOut.MyInner snc = new MyOut.MyInner();
        snc.inner();
    }
}

public class Myout2
{
    String s1 = "hello";
    static String s2 = "world!";
        interface My
    {
        void show();
    }
    static class MyInner2 implements My
    {
        public void show()
        {
```

```
            System.out.print(new Myout2().s1);
        }
    }
    public static void main(String[] args)
    {
        Myout2.MyInner2 in = new Myout2.MyInner2();
        in.show();
    }
}
```

2、枚举类型

1、枚举类型的定义

```
final class Direction
{
    public static final int EAST = 1;
    public static final int SOUTH = 2;
    public static final int WEST = 3;
    public static final int NORTH = 4;
}

enum Direction
{
    EAST, SOUTH, WEST,NORTH;
}

public class Direction1
{
public static void main(String[] args)
{
    Direction d = Direction.WEST;
    System.out.print(d);
    for(Direction s : Direction.values())
    {
        System.out.println(s.name() + s.ordinal());
    }
}
}

import java.time.DayOfWeek;
public class EnumSwitch {
    public static void desc(DayOfWeek d)
    {
        switch(d)
        {
            case MONDAY:
                System.out.print("week 1");
                break;
            case FRIDAY:
                System.out.print("week 5");
                break;
        }
    }
    public static void main(String[] args)
    {
        DayOfWeek f1 = DayOfWeek.MONDAY;
```

```
        desc(f1);
    }
}
```