第三章 封装

课前回顾

1. 带参方法如何定义

```
访问修饰符  返回值类型  方法名(参数列表){//形式参数列表

}

对象名.方法名(实参列表);
```

2. 方法的参数是如何传递的

在Java中，方法参数的传递都是值传递。只是基本数据类型作为参数传递时，传递的是值的拷贝。引用数据类型作为参数传递时，传递的是堆内存的空间地址值

3. 什么是方法重载

在同一个类中，多个方法名称相同，但参数列表不一致，这些方法构成了方法重载。

4 . return 关键字有什么作用

return 关键字的主要作用就是给出方法执行后的结果，使方法直接结束。

章节内容

- 封装的好处及使用 **重点**
- 包的使用 **重点**
- 访问修饰符 **重点**
- static 修饰符 **重点 难点**

章节目标

- 掌握封装的使用
- 掌握包的使用及作用
- 掌握访问修饰符的使用
- 掌握 static 修饰符的使用

第一节 封装（Encapsulation）

1. 什么是封装

封装就是将类的部分属性和方法隐藏起来，不允许外部程序直接访问，只能通过该类提供的公开的方法来访问类中定义的属性和方法。封装是面向对象的三大特征之一。

2.如何使用封装

示例

```
public class Person{

```

```
    public String name; //姓名

    public int age; //年龄

    public String secret; //秘密

    public Person(String name, int age, String secret){
        this.name = name;
        this.age = age;
        this.secret = secret;
    }
}
```

修改属性的可见性：将类中定义的所有属性全部修改为private修饰

```
public class Person{

    private String name; //姓名

    //private表示私有的意思 相当于个人隐私
    private int age; //年龄

    private String secret; //秘密

    public Person(String name, int age, String secret){
        this.name = name;
        this.age = age;
        this.secret = secret;
    }
}
```

创建公开的getter/setter方法：用于读取/修改属性值

```
public class Person{

    private String name; //姓名

    //private表示私有的意思 相当于个人隐私
    private int age; //年龄

    private String secret; //秘密

    public Person(String name, int age, String secret){
        this.name = name;
        this.age = age;
        this.secret = secret;
    }

   //ALT + INSERT

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
```

```
    }

    public int getAge() {
        return age;
    }

    public void setAge(int age) {
        this.age = age;
    }

    public String getSecret() {
        return secret;
    }

    public void setSecret(String secret) {
        this.secret = secret;
    }
}

```

在getter/setter方法中加入控制语句：用于对属性值的合法性校验

```
public void setAge(int age) {
    if(age < 0){
        System.out.println("你输入的年龄不合法，年龄必须为 > 0 的整数");
    } else {
        this.age = age;
    }
}
```

3. 为什么要使用封装

a. 封装提高了代码的重用性。因为封装会提供对外访问的公开的方法，而方法可以重用，因此封装提高了代码的重用性。

b. 封装提高了代码的可维护性。修改代码时，只需要修改部分代码，但不会影响其他代码

比如：年龄在设计时只考虑到了负数的情况，没有考虑实际生活中的情况，人的年龄一般都不会超过200岁，因此还需要加上一层验证

```
public void setAge(int age) {
    if(age < 0 || age > 200){
        System.out.println("你输入的年龄不合法，年龄必须为 0 ~ 200之间的整数");
    } else {
        this.age = age;
    }
}
```

c. 封装隐藏了类的具体实现细节，保护了代码实现逻辑。

第二节 包（Package）

1. 什么是包

包是Java中的一个专业词汇，包的本质就是一个文件夹。

2. 为什么要使用包

因此包可以对我们编写的类进行分类、可以防止命名冲突和访问权限控制

3. 如何创建包

语法

```
package 包名;
```

包名的命名规范:

包名一般都是由小写字母和数字组成，每个包之间使用'.'隔开，换言之，每出现一个'.'，就是一个包

包名一般都含有前缀。比如个人/组织通常都是`org.姓名`，公司通常都是`com.公司名称简写`或者`cn.公司名称简写`

如果一个类中有包的定义，那么这个类的第一行有效代码一定是包的定义

4. 如何引入包

为了使用不在同一包中的类，需要在Java程序中使用import关键字引入这个类

语法

```
import 包名.类名;
```

为什么会引入包?

因为`JVM`只能识别当前包下所有的类，如果要使用当前包之外的其他包中的类，那必须告诉`JVM`，使用的是哪一个包中的哪一个类。

示例

```
package org.wu.lesson;

//类的全限定名:  包名 + "." + 类名
import java.util.Scanner;//告诉JVM，到java.util包下去找一个名为Scanner的类

public class Test {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        Student student = new Student();
    }
}
```

一个类同时引用了两个来自不同包的同名类，必须通过完整类名（类的全限定名）来区分。

示例

```
package org.wu.lesson;

//类的全限定名:  包名 + "." + 类名
import com.alibaba.dubbo.User;

import java.util.Scanner;//告诉JVM，到java.util包下去找一个名为Scanner的类

```

```java
public class Test {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        Student student = new Student();

        User user = new User();
        //因为该类中引入了com.alibaba.dubbo.User，如果不写包名，那么默认使用的就是
        //com.alibaba.dubbo.User。如果需要使用其他包中的User，则必须使用类的全限定
        //名来进行对象的构建与赋值操作
        com.ly.chapter11.User user1 = new com.ly.chapter11.User();
    }
}
```

5. 常用包说明

java.lang 包：属于Java 语言开发包，该包中的类可以直接拿来使用，而不需要引入包。因此 `JVM` 会自动引入。比如我们经常使用的System、String、Math

java.util 包：属于Java 提供的一些使用类以及工具类。比如我们经常使用的Scanner

第三节 访问修饰符

1. 概念

访问修饰符就是控制访问权限的修饰符号

2. 类的访问修饰符

类的访问修饰符只有两种：public修饰符和默认修饰符（不写修饰符就是默认）

public 修饰符修饰类表示类可以公开访问。默认修饰符修饰类表示该类只能在同一个包中可以访问。

示例

```java
package cn.lyxq.chapter11;
//使用默认修饰符修饰类时，该类只能在同一个包的其他类中使用
class Teacher {
}
```

```java
package cn.lyxq.chapter11;

public class School {

    private Teacher[] teachers;//可以访问Teacher类

}
```

```java
package cn.lyxq.chapter11.test;

import cn.lyxq.chapter11.Teacher;

public class TeacherTest {

    public static void main(String[] args) {
        Teacher teacher = new Teacher();
    }
}
```

3. 类成员访问修饰符

类成员包括了成员属性和成员方法。类成员访问修饰符换言之就是成员属性和成员方法的访问修饰符。

访问修饰符	作用范围			
	同一个类中	同一个包中	子类	任何地方
private	可以访问	不可以访问	不可以访问	不可以访问
默认修饰符	可以访问	可以访问	不可以访问	不可以访问
protected	可以访问	可以访问	可以访问	不可以访问
public	可以访问	可以访问	可以访问	可以访问

示例

```java
package cn.lyxq.chapter11;

public class School {

    private Teacher[] teachers;//可以访问Teacher类
    //使用默认修饰符修饰name属性
    String name;

    protected int age;

    public String address;

    public void show(){
        System.out.println(teachers.length + "\t" + name + "\t" + age + "\t" + address);
    }
}
```

```java
package cn.lyxq.chapter11;

public class SchoolTest {

    public static void main(String[] args) {
        School school = new School();
        //外部不能访问private修饰的属性
        System.out.println(school.teachers);
        System.out.println(school.name);
        System.out.println(school.age);
        System.out.println(school.address);
    }
}
```

```java
package cn.lyxq.chapter11.test;

import cn.lyxq.chapter11.School;

public class SchoolTest1 {

    public static void main(String[] args) {
        School school = new School();
        //外部不能访问private修饰的属性
        System.out.println(school.teachers);
        System.out.println(school.name);
        System.out.println(school.age);
        System.out.println(school.address);
    }
}
```

第四节 static 修饰符

1. static 修饰符应用范围

static 修饰符只能用来修饰类中定义的成员变量、成员方法、代码块以及内部类（内部类有专门章节进行讲解）。

2. static 修饰成员变量

static 修饰的成员变量称之为类变量。属于该类所有成员共享。

示例

```java
package cn.lyxq.chapter11;

public class ChinesePeople {

    private String name;

    private int age;
    //使用static修饰的成员变量称为类变量，不会随着成员变化而变化，属于所有成员共享
    public static String country = "中国";

    public ChinesePeople(String name, int age){
        this.name = name;
```

```
        this.age = age;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public int getAge() {
        return age;
    }

    public void setAge(int age) {
        this.age = age;
    }
}
```

```
package cn.lyxq.chapter11;

public class ChinesePeopleTest {

    public static void main(String[] args) {
        ChinesePeople cp1 = new ChinesePeople("张三", 20);

        ChinesePeople.country = "日本";

        ChinesePeople cp2 = new ChinesePeople("李四", 30);
        System.out.println(ChinesePeople.country);

        ChinesePeople cp3 = new ChinesePeople("王五", 32);
        System.out.println(cp3.country);
    }
}
```

如果类变量是公开的，那么可以使用 `类名.变量` 名直接访问该类变量。

3. static 修饰成员方法

static 修饰的成员方法称之为类方法。属于该类所有成员共享。

```
package cn.lyxq.chapter11;

public class ChinesePeople {

    private String name;

    private int age;
    //使用static修饰的成员变量称为类变量，不会随着成员变化而变化，属于所有成员共享
    private static String country = "中国";

    public ChinesePeople(String name, int age){
        this.name = name;
        this.age = age;
    }
```

```
15
16      public String getName() {
17          return name;
18      }
19
20      public void setName(String name) {
21          this.name = name;
22      }
23
24      public int getAge() {
25          return age;
26      }
27
28      public void setAge(int age) {
29          this.age = age;
30      }
31      //类方法
32      public static String getCountry() {
33          return country;
34      }
35      //类方法
36      public static void setCountry(String country) {
37          ChinesePeople.country = country;
38      }
39  }
```

```
1   package cn.lyxq.chapter11;
2
3   public class ChinesePeopleTest {
4
5       public static void main(String[] args) {
6           ChinesePeople cp1 = new ChinesePeople("张三", 20);
7           cp1.setCountry("日本");
8
9           ChinesePeople cp2 = new ChinesePeople("李四", 30);
10          System.out.println(ChinesePeople.getCountry());
11
12          ChinesePeople cp3 = new ChinesePeople("王五", 32);
13          System.out.println(cp3.getCountry());
14      }
15  }
```

如果类方法是公开的，那么可以使用`类名.方法名`名直接访问该类方法。

4. static 修饰代码块

static 修饰的代码块称为静态代码块，在 `JVM` 第一次记载该类时执行。因此，静态代码块只能够被执行一次，通常用于一些系统设置场景。

```
1   package cn.lyxq.chapter11;
2
3   public class ChinesePeople {
4
5       private String name;
6
7       private int age;
```

```
    //使用static修饰的成员变量称为类变量，不会随着成员变化而变化，属于所有成员共享
    private static String country;
    //static修饰的代码块称为静态代码块，在JVM第一次加载该类的时候执行，只能执行一次
    static {
        country = "中国";
        System.out.println("country属性已经被赋值");
    }

    public ChinesePeople(String name, int age){
        this.name = name;
        this.age = age;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public int getAge() {
        return age;
    }

    public void setAge(int age) {
        this.age = age;
    }
    //类方法
    public static String getCountry() {
        return country;
    }
    //类方法
    public static void setCountry(String country) {
        ChinesePeople.country = country;
    }
}
```

5. static内存