9. 类和对象（一）

【本节目标】

1. 掌握类的定义方式以及对象的实例化
2. 掌握类中的成员变量和成员方法的使用
3. 掌握对象的整个初始化过程

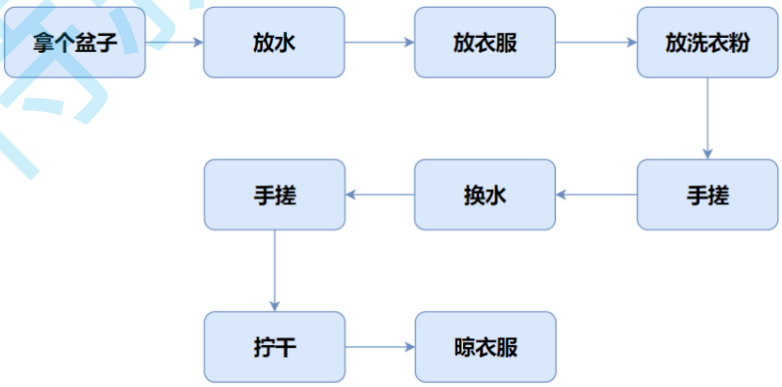
1. 面向对象的初步认知

1.1 什么是面向对象

Java是一门纯面向对象的语言(Object Oriented Program，简称OOP)，在面向对象的世界里，一切皆为对象。**面向对象是解决问题的一种思想，主要依靠对象之间的交互完成一件事情**。用面向对象的思想来设计程序，更符合人们对事物的认知，对于大型程序的设计、扩展以及维护都非常友好。

1.2 面向对象与面向过程

传统洗衣服过程

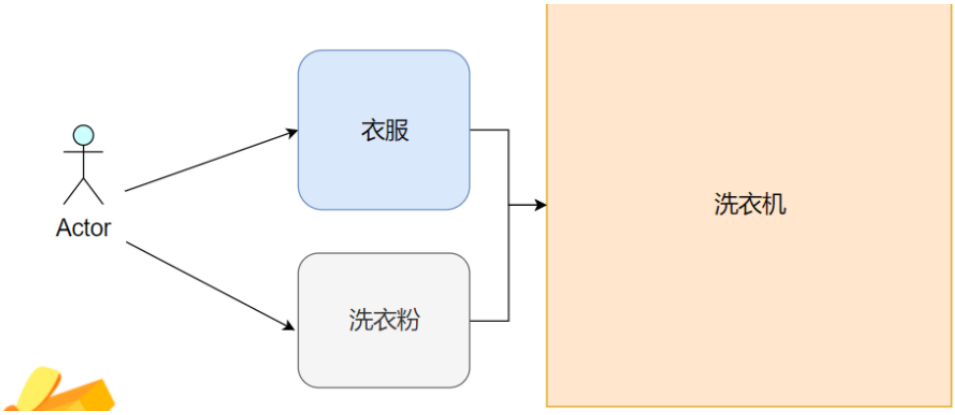


传统的方式：**注重的是洗衣服的过程**，少了一个环节可能都不行。

而且不同衣服洗的方式，时间长度，拧干方式都不同，处理起来就比较麻烦。如果将来要洗鞋子，那就是另一种方式。

按照该种方式来写代码，将来**扩展或者维护起来会比较麻烦**。

现代洗衣服过程



以**面向对象方式来进行处理，就不关注洗衣服的过程**，具体洗衣机是怎么来洗衣服，如何来甩干的，用户不用去关心，只需要将衣服放进洗衣机，倒入洗衣粉，启动开关即可，**通过对象之间的交互来完成的**。

注意：面向过程和面向对象并不是一门语言，而是解决问题的方法，没有那个好坏之分，都有其专门的应用场景。

2. 类定义和使用

面相对象程序设计关注的是对象，而对象是现实生活中的实体，比如：洗衣机。但是洗衣机计算机并不认识，需要开发人员告诉给计算机什么是洗衣机。

上图左侧就是**对洗衣机简单的描述，该过程称为对洗衣机对象(实体)进行抽象(对一个复杂事物的重新认知)**，但是这些简化的抽象结果计算机也不能识别，开发人员可以采用某种面相对象的编程语言来进行描述，比如：Java语言。

2.1 简单认识类

类是用来对一个实体(对象)来进行描述的，主要描述该实体(对象)具有哪些属性(外观尺寸等)，哪些功能(用来干啥)，描述完成后计算机就可以识别了。

> 比如：洗衣机，它是一个品牌，在Java中可以将其看成是一个类别。
>
> 属性：产品品牌，型号，产品重量，外观尺寸，颜色...
>
> 功能：洗衣，烘干、定时....

在Java语言中，如何对上述的洗衣机类来进行定义呢?

2.2 类的定义格式

在Java中定义类时需要用到class关键字，具体语法如下

```
// 创建类
class ClassName{
    field;        // 字段(属性) 或者 成员变量
    method;       // 行为 或者 成员方法
}
```

class为定义类的关键字，ClassName为类的名字，{}中为类的主体。

类中包含的内容称为类的成员。属性主要是用来描述类的，称之为**类的成员属性或者类成员变量**。方法主要说明类具有哪些功能，称为**类的成员方法**。

```
class WashMachine{
    public String brand;   // 品牌
    public String type;    // 型号
    public double weight;  // 重量
    public double length;  // 长
    public double width;   // 宽
    public double height;  // 高
    public String color;   // 颜色

    public void washClothes(){    // 洗衣服
        System.out.println("洗衣功能");
    }

    public void dryClothes(){     // 脱水
        System.out.println("脱水功能");
    }

    public void setTime(){        // 定时
        System.out.println("定时功能");
    }
}
```

采用Java语言将洗衣机类在计算机中定义完成，经过javac编译之后形成.class文件，在JVM的基础上计算机就可以识别了。

注意事项

- **类名注意采用大驼峰定义**
- 成员前写法统一为public，后面会详细解释
- **此处写的方法不带 static 关键字**. 后面会详细解释

2.3 课堂练习

- 定义一个Dog类，大家可以思考一下，哪些是Dog的属性？哪些是Dog的行为？

参考代码如下：

```
class PetDog {
    public String name;//名字
    public String color;//颜色

    // 狗的属性
    public void barks() {
        System.out.println(name + ": 旺旺旺~~~");
    }

    // 狗的行为
    public void wag() {
        System.out.println(name + ": 摇尾巴~~~");
    }
}
```

- 定义一个学生类Student，大家可以思考一下，哪些是Student的属性？哪些是Student的行为？

参考代码如下：

```
public class Student{
    public String name;
    public String gender;
    public short  age;
    public double score;

    public void DoClass(){}
    public void DoHomework(){}
    public void Exam(){}
```

```
10  }
```

注意事项：

1. 一般一个文件当中只定义一个类
2. public修饰的类必须要和文件名相同
3. 不要轻易去修改public修饰的类的名称，如果要修改，通过开发工具修改

3. 类的实例化

3.1 什么是实例化

定义了一个类，就相当于在计算机中定义了一种新的类型，与int，double类似，只不过int和double是java语言自带的内置类型，而类是用户自定义了一个新的类型，比如上述的：PetDog类和Student类。它们都是类(一种新定义的类型)有了这些自定义的类型之后，就可以使用这些类来定义实例(或者称为对象)。

用类类型创建对象的过程，称为类的实例化，在java中采用new关键字，配合类名来实例化对象。

```
1  public class Main{
2      public static void main(String[] args) {
3          PetDog dogh = new PetDog();     //通过new实例化对象
4
5          PetDog dogs = new PetDog();
6      }
7  }
```

3.2 如何访问对象当中的成员

```
1  public class Main{
2          public static void main(String[] args) {
3          PetDog dogh = new PetDog();     //通过new实例化对象
4          dogh.name = "阿黄";
5          dogh.color = "黑色";
6          dogh.barks();
7          dogh.wag();
8
9          PetDog dogs = new PetDog();
10         dogs.name = "赛虎";
11         dogs.color = "黄色";
12         dogs.barks();
13         dogs.wag();
```

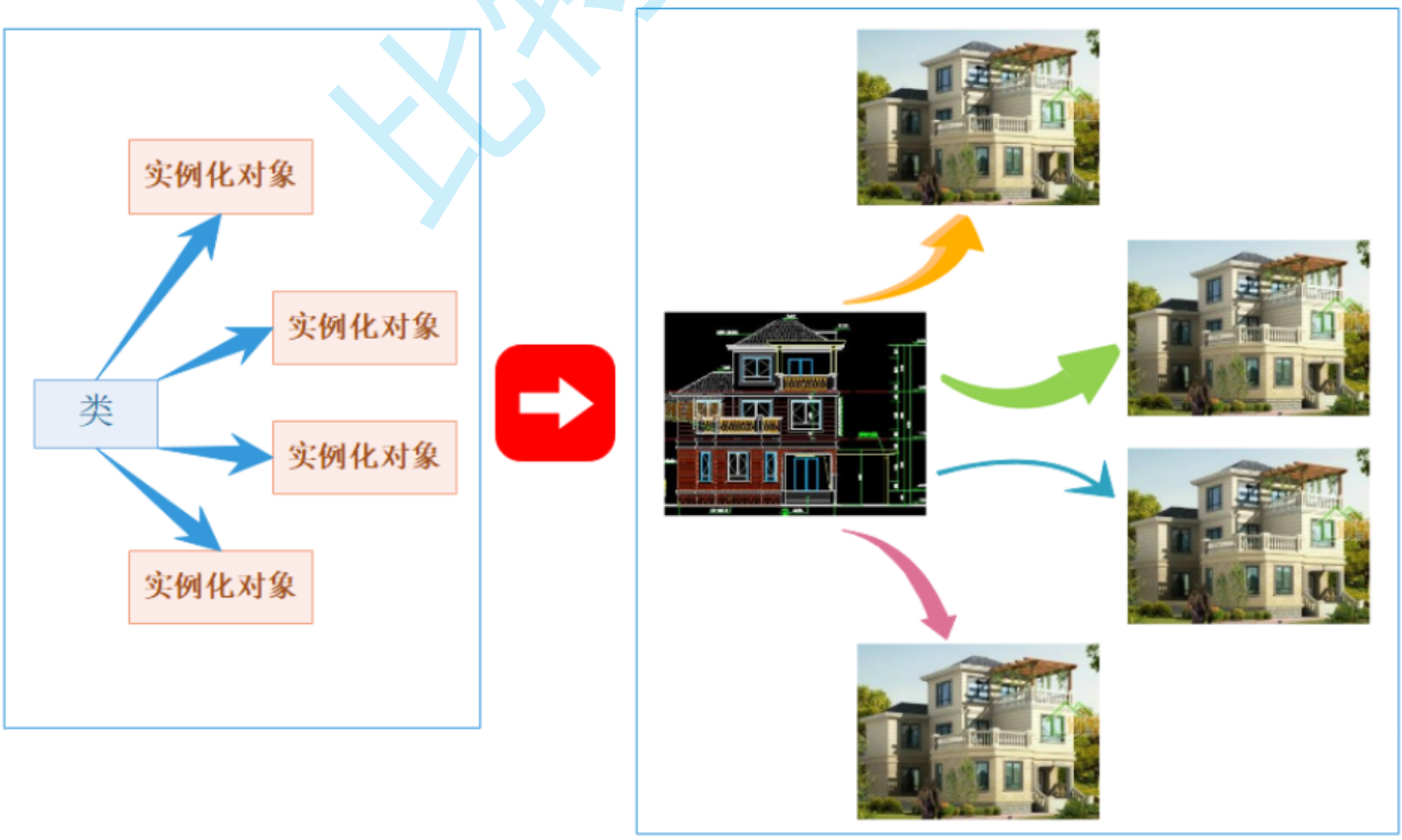
```
        }
    }

    输出结果:
    阿黄: 旺旺旺~~~
    阿黄: 摇尾巴~~~
    赛虎: 旺旺旺~~~
    赛虎: 摇尾巴~~~
```

注意事项

- new 关键字用于创建一个对象的实例.
- 使用 `.` 来访问对象中的属性和方法.
- 同一个类可以创建多个实例.

3.3 类和对象的再次理解

1. **类只是**一个**模型**一样的东西，用来对一个实体进行描述，限定了类有哪些成员.
2. **类是一种自定义的类型**，可以用来定义变量.
3. 一个类可以实例化出多个对象，**实例化出的对象 占用实际的物理空间，存储类成员变量**
4. 做个比方。**类实例化出对象就像现实中使用建筑设计图建造出房子，类就像是设计图**，只设计出需要什么东西，但是并没有实体的建筑存在，同样类也只是一个设计，实例化出的对象才能实际存储数据，占用物理空间



4. this关键字

4.1 为什么要有this引用

如下代码定义了一个Date类，Date类中包含3个属性分别是year，month，day。分别使用setDay和printDate对进行设置和打印日期。

```
public class Date {
    public int year;
    public int month;
    public int day;

    public void setDay(int y, int m, int d){
        year = y;
        month = m;
        day = d;
    }

    public void printDate(){
        System.out.println(year + "/" + month + "/" + day);
    }

    public static void main(String[] args) {
        // 构造三个日期类型的对象 d1 d2 d3
        Date d1 = new Date();
        Date d2 = new Date();
        Date d3 = new Date();

        // 对d1，d2，d3的日期设置
        d1.setDay(2020,9,15);
        d2.setDay(2020,9,16);
        d3.setDay(2020,9,17);

        // 打印日期中的内容
        d1.printDate();
        d2.printDate();
        d3.printDate();
    }
}
```


提出2个疑问：

1. **形参名不小心与成员变量名相同，会发生什么？**

```
public void setDay(int year, int month, int day){
    year = year;
    month = month;
    day = day;
}
```

2. 三个对象都在调用setDate和printDate函数，但是这两个函数中没有任何有关对象的说明，**setDate和printDate函数如何知道打印的是那个对象的数据呢**?

4.2 this是什么

this引用指向当前对象(成员方法运行时调用该成员方法的对象)，在成员方法中所有成员变量的操作，都是通过该引用去访问。只不过所有的操作对用户是透明的，即用户不需要来传递，编译器自动完成。

```
public class Date {
    public int year;
    public int month;
    public int day;

    public void setDay(int year, int month, int day){
        this.year = year;
        this.month = month;
        this.day = day;
    }

    public void printDate(){
        System.out.println(this.year + "/" + this.month + "/" + this.day);
    }

    public static void main(String[] args) {
    Date d = new Date();
    d.setDay(2020,9,15);
    d.printDate();
}
}
```


此时通过指定this，就能正确解决上述的2个问题了。这里之所以可以起作用，**关键点在于this代表的是当前对象的引用，当前对象就是指：哪个对象调用setDay方法，谁就是this引用！**

4.3 this引用的特性

1. this的类型：对应类类型引用，即哪个对象调用就是哪个对象的引用类型
2. this只能在"成员方法"中使用

3. 在"成员方法"中，this只能引用当前对象，不能再引用其他对象
4. this是“成员方法”第一个隐藏的参数，编译器会自动传递，在成员方法执行时，编译器会负责将调用成员方法对象的引用传递给该成员方法，this负责来接收

```java
public class Date {
    public int year;
    public int month;
    public int day;

    public void setDate(int y, int m, int d){
        year = y;
        month = m;
        day = d;
    }

    public void printDate(){
        System.out.println(year + "-" + month + "-" + day);
    }

    public static void main(String[] args) {
        Date d = new Date();
        d.setDate(2021,6,9);
        d.printDate();
    }
}
```

编译器编译之后将"成员方法"的隐藏的this参数还原，在方法中所有"成员变量"都通过this引用来访问

```java
public class Date {
    public int year;
    public int month;
    public int day;

    public void setDate(Date this, int y, int m, int d){
        this.year = y;
        this.month = m;
        this.day = d;
    }

    public void printDate(Date this){
        System.out.println(this.year + "-" + this.month + "-" + this.day);
    }

    public static void main(String[] args) {
        Date d = new Date();
        d.setDate(2021,6,9);
        d.printDate();
    }
}
```

5. this的使用总结

```java
this.成员变量 //访问对象的成员变量
this.成员方法 //访问对象的成员方法
```

事实上，我们也可以通过this来访问构造方法，那么什么是构造方法呢?

5. 对象的构造及初始化

5.1 如何初始化对象中的成员变量

通过前面知识点的学习知道，在Java方法内部定义一个局部变量时，必须要初始化，否则会编译失败。

```java
public static void main(String[] args) {
    int a;
    System.out.println(a);
}

// Error:(26, 28) java: 可能尚未初始化变量a
```

要让上述代码通过编译，非常简单，只需在正式使用a之前，给a设置一个初始值即可。如果是对象：

```java
public static void main(String[] args) {
    Date d = new Date();
    d.printDate();
```

```
        d.setDate(2021,6,9);
        d.printDate();
    }

    // 代码可以正常通过编译
```

需要调用之前写的SetDate方法才可以将具体的日期设置到对象中。**通过上述例子发现两个问题：**

1. **每次对象创建好后调用SetDate方法设置具体日期，比较麻烦，那对象该如何初始化？**
2. **局部变量必须要初始化才能使用，为什么字段声明之后没有给值依然可以使用？**

5.2 默认初始化

```
public class Date {
    public int year;
    public int month;
    public int day;

    public static void main(String[] args) {
        // 此处a没有初始化，编译时报错：
        // Error:(24, 28) java: 可能尚未初始化变量a
        // int a;
        // System.out.println(a);
        Date d = new Date();
        System.out.println(d.year);
        System.out.println(d.month);
        System.out.println(d.day);
    }
}
//输出
0
0
0
```


对于成员变量来说，如果没有进行初始化，会有一个对应的默认值，默认值遵循如下规则：

数据类型	默认值
byte	0
char	'\u0000'
short	0
int	0
long	0L
boolean	FALSE
float	0.0f
double	0
reference	null

5.3 就地初始化

在声明成员变量时，就直接给出了初始值。

```
1   public class Date {
2       public int year = 1900;
3       public int month = 1;
4       public int day = 1;
5
6       public Date(){
7       }
8
9       public Date(int year, int month, int day) {
10      }
11
12      public static void main(String[] args) {
13          Date d1 = new Date(2021,6,9);
14          Date d2 = new Date();
15      }
16  }
```

5.4 构造方法初始化

5.4.1 构造方法概念

构造方法(也称为构造器)是一个特殊的成员方法，**名字必须与类名相同，在创建对象时，由编译器自动调用，并且在整个对象的生命周期内只调用一次。**

```
1   public class Date {
```

```
    public int year;
    public int month;
    public int day;

    public Date(int year, int month, int day){
        this.year = year;
        this.month = month;
        this.day = day;
        System.out.println("Date(int,int,int)方法被调用了");
    }

    public void printDate(){
        System.out.println(year + "-" + month + "-" + day);
    }

    public static void main(String[] args) {
        // 此处创建了一个Date类型的对象，并没有显式调用构造方法
        Date d = new Date(1999,6,9);   // 输出Date(int,int,int)方法被调用了
        d.printDate();    // 1999-6-9
    }
}
```

5.4.2 构造方法注意事项

1. 名字必须与类名相同
2. 没有返回值类型，设置为void也不行
3. 创建对象时由编译器自动调用，并且在对象的生命周期内只调用一次(相当于人的出生，每个人只能出生一次)
4. 构造方法可以重载(用户根据自己的需求提供不同参数的构造方法)

```
public class Date {
    public int year;
    public int month;
    public int day;

    // 无参构造方法
    public Date(){
        this.year = 1900;
        this.month = 1;
        this.day = 1;
    }

    // 带有三个参数的构造方法
    public Date(int year, int month, int day) {
```

```
        this.year = year;
        this.month = month;
        this.day = day;
    }

    public void printDate(){
        System.out.println(year + "-" + month + "-" + day);
    }

    public static void main(String[] args) {
        Date d = new Date();
        d.printDate();
    }
```

上述两个构造方法：名字相同，参数列表不同，因此构成了方法重载。

5. 如果用户没有显式定义，编译器会生成一份默认的构造方法，生成的默认构造方法一定是无参的。

```
public class Date {
    public int year;
    public int month;
    public int day;

    public void printDate(){
        System.out.println(year + "-" + month + "-" + day);
    }

    public static void main(String[] args) {
        Date d = new Date();
        d.printDate();
    }
}
```

上述Date类中，没有定义任何构造方法，编译器会默认生成一个不带参数的构造方法。

6. 一旦用户定义了其他的构造方法，编译器则不再生成。

```
public class Date {
    public int year;
    public int month;
    public int day;

    public Date(int year, int month, int day) {
        this.year = year;
```

```
        this.month = month;
        this.day = day;
    }

    public void printDate(){
        System.out.println(year + "-" + month + "-" + day);
    }

    public static void main(String[] args) {
        Date d = new Date();
        d.printDate();
    }
}

```

上述代码运行之后会显示，没有无参构造方法。

6. 构造方法中，可以通过this调用其他构造方法来简化代码

```
public class Date {
    public int year;
    public int month;
    public int day;

    public Date(){
        this(1900, 1, 1);
    }

    // 带有三个参数的构造方法
    public Date(int year, int month, int day) {
        this.year = year;
        this.month = month;
        this.day = day;
    }
}
```

- **this(...)必须是构造方法中第一条语句**
- **不能形成环的调用**

```
public Date(){
   this(1900,1,1);
}

```

```
5   public Date(int year, int month, int day) {
6       this();
7   }
```

7. 绝大多数情况下使用public来修饰，特殊场景下会被private修饰(后序讲单例模式时会遇到)

6. 对象的打印

如果我们直接打印对象的引用，此时输出的结果为：类路径名@对象的hashcode值。

```
1   public class Person {
2       String name;
3       String gender;
4       int age;
5
6       public Person(String name, String gender, int age) {
7           this.name = name;
8           this.gender = gender;
9           this.age = age;
10      }
11
12      public static void main(String[] args) {
13          Person person = new Person("Jim","男", 18);
14          System.out.println(person);
15      }
16  }
17
```

如果想要默认打印对象中的属性该如何处理呢？答案：重写toString方法即可。

```
1   public class Person {
2       String name;
3       String gender;
4       int age;
5
6       public Person(String name, String gender, int age) {
7           this.name = name;
8           this.gender = gender;
9           this.age = age;
10      }
11
```

```
    @Override
    public String toString() {
        return "[" + name + "," + gender + "," + age + "]";
    }

    public static void main(String[] args) {
        Person person = new Person("Jim","男", 18);
        System.out.println(person);
    }
}

// 输出结果：[Jim,男,18]
```

关于**重写**的理解，我们会在继承多态章节讲到。

7. 小试牛刀

1. 下列说法哪些是错误的？为什么？

   A. 一个类只能实例化一个对象

   B. 一个引用可以同时指向多个对象

   C. Person p = null; 表示p这个引用指向一个空对象

   D. 当一个类没有定义构造方法的时候，该类中不会默认生成构造方法

2. 找出以下代码错误的地方并改正？

```
class Person {
    public String name;
    public int age;

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

    public static void main(String[] args) {
        Person person1 = new Person();
        person1.name = "zhangsan";
        person1.age = 10;
        System.out.println(person1.name);
        System.out.println(person1.age);
        System.out.println("============================");

        Person person2 = new Person("zhangfei");
```

```
19          person2.age = 10;
20          System.out.println(person2.name);
21          System.out.println(person2.age);
22          System.out.println("============================");
23
24          Person person3 = new Person("lisi",9);
25          System.out.println(person3.name);
26          System.out.println(person3.age);
27      }
28  }
```

答案解析：

```
1  1. A、 B、 C、 D
```

完