5. 面向对象编程

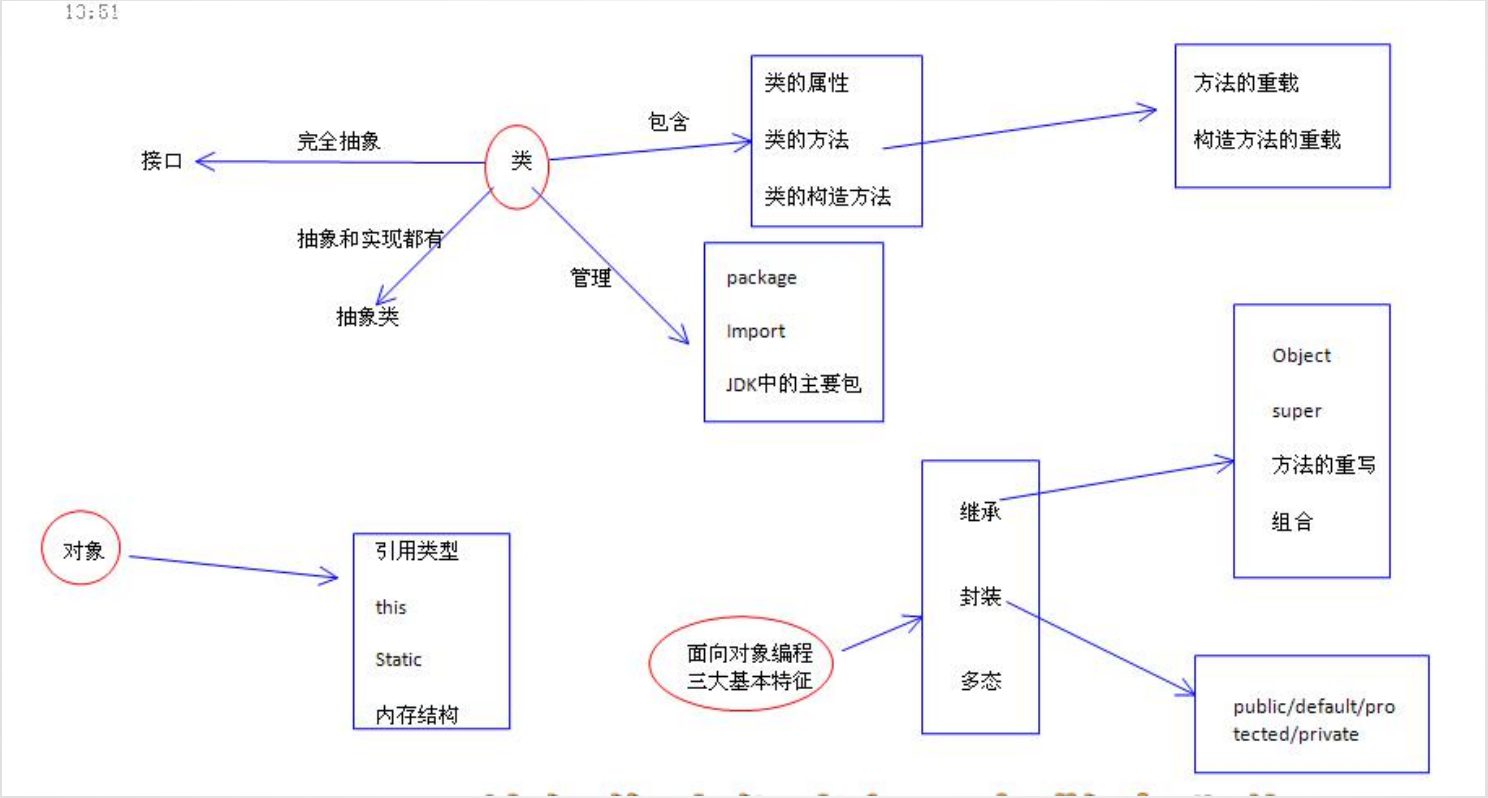
本章技能点列表

技能点名称	难易程度	认知程度	重要程度
面向过程和面向对象	中	理解	**
类和对象	难	理解	***
类的属性	中	应用	***
类的方法	中	应用	***
局部变量和成员变量	中	应用	***
构造方法及其重载	中	应用	***
基本数据类型参数的方法调用	中	应用	***
引用数据类型参数的方法调用	难	应用	***

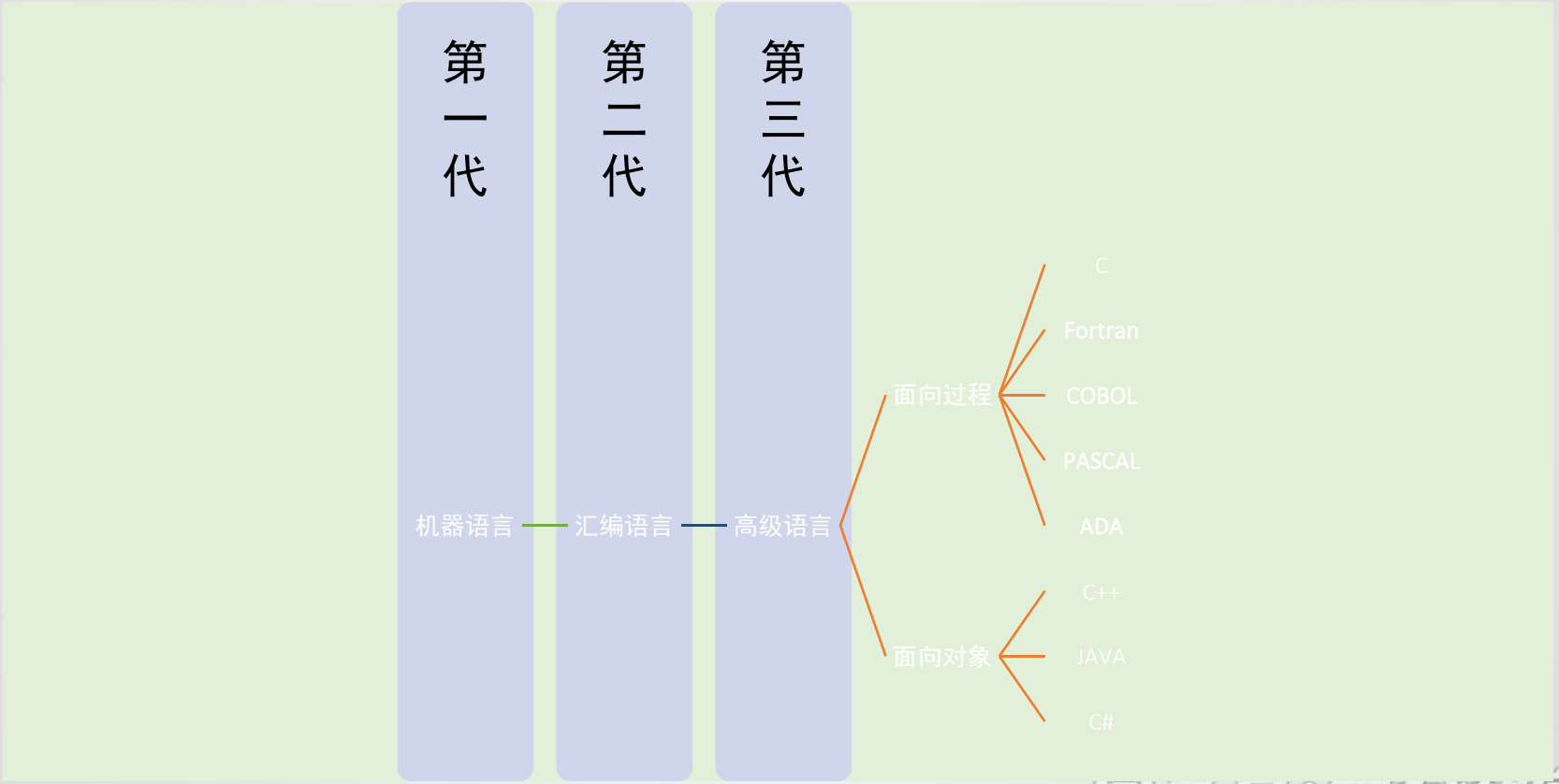
本章技能点列表

技能点名称	难易程度	认知程度	重要程度
this关键字	难	理解	**
static变量	难	理解	***
static方法	难	理解	***
static代码块	易	理解	**
package和import	易	理解	**
静态导入	易	理解	*

本章概述


13:51
接口
完全抽象
类
包含
类的属性
类的方法
类的构造方法
方法的重载
构造方法的重载
抽象和实现都有
抽象类
管理
package
Import
JDK中的主要包
Object
super
方法的重写
组合
对象
引用类型
this
Static
内存结构
面向对象编程
三大基本特征
继承
封装
多态
public/default/protected/private

面向对象编程初步（OOP:Object Oriented Programming）



面向对象编程初步（OOP:Object Oriented Programming）

	面向过程	面向对象
区别	**事物比较简单，可以用线性的思维去解决**	**事物比较复杂，使用简单的线性思维无法解决**
共同点	面向过程和面向对象都是解决实际问题的一种思维方式	
	二者相辅相成，并不是对立的。 解决复杂问题，通过面向对象方式便于我们从宏观上把握事物之间复杂的关系、方便我们分析整个系统；具体到微观操作，仍然使用面向过程方式来处理	

面向对象编程初步（OOP:Object Oriented Programming）

- 如何开汽车(事物比较简单，可以用线性的思维去解决)

- 面向过程：
- 1.采离合
- 2. 挂档
- 3.踩油门,放离合
- 4.开了

- 面向对象：

✓驾驶员

✓汽车

✓驾驶员开汽车！

```
car.start();
```

面向对象编程初步（OOP:Object Oriented Programming）

- **如何造汽车(事物比较复杂，使用简单的线性思维无法解决)**
- 面向过程：
  - 1. 造车轮?
  - 2. 造发动机?
  - 3. 造车皮?
  - 4. 挡风玻璃? ...

很难决定上面这些步骤之间的关系！先造发动机还是先造车轮?

- **面向对象：**
- 车轮
  - 买橡胶
    - 到马来西亚
    - 找到橡胶厂
    - 掏钱买
    - 用船将橡胶运到国内
  - 造磨具
  - 将橡胶放入磨具
  - 出车轮
- 发动机
  - ....
- 车壳
  - ....
- 座椅
  - ...
- 挡风玻璃
  - ....
- **将上面的造出的东东，组装，汽车造出！**

面向对象编程初步（OOP:Object Oriented Programming）

解决问题：如何统一中国?（很复杂，不能用面向过程解决！）

- 蒋介石
- 面向过程的
  杰出代表

1. 抢占战败日本的物资，收编伪军，拉美国的金援。
2. 和共产党谈判，拖延时间，为战争准备争取时间
3. 抢占城市等战略要点
4. 开打！
5. 赢！哦耶！

- 毛泽东
- 面向对象的
  杰出代表

- 共产党（加强党的建设、增强凝聚力战斗力）
- 农民
  - 贫农 中农 富农
- 工人
- 知识分子
- 国民党
  - 进步派 中间派 反动派
- 各民主派别

用统一战线战略来处理这些对象的关系

对象和类的概念

对象：是具体的事物 xiaoming xiaohong
类：是对对象的抽象（抽象　抽出象的部分）Person

先有具体的对象，然后抽象各个对象之间象的部分，归纳出类通过类再认识其他对象。

生活案例

- 类是一个图纸 对象是根据该图纸制造多个实物
- 类是一个模具　对象是使用模具制造的多个铸件（月饼模子 ）
- 类是上海大众汽车，对象就是大家购买的一辆辆具体上海大众汽车

认识类和对象

- 需求：使用面向对象思想表示人的日常生活

- 分析
  - 由多个具体的人（小红、小张、老李）抽象出所有的公共特征
  - 静态特征：姓名、年龄、性别
  - 动态行为：吃饭、休息、自我介绍

- 实现
  - 创建抽象的人-----Person类
  - 创建具体的人-----对象

类和对象的总结

- 类：class。
- 对象：Object, instance(实例)。以后我们说某个类的对象，某个类的实例。是一样的意思。

- 对象和类的关系：
  - 特殊到一般，具体到抽象。
  - 类可以看成一类对象的模板，对象可以看成该类的一个具体实例。
  - 类是用于描述同一类形的对象的一个抽象的概念，类中定义了这一类对象所应具有的静态和动态属性。

  - JDK提供了很多类供编程人员使用，编程人员也可定义自己的类。

类和对象的总结

- 定义类（类的组成）
  - 属性 field
  - 方法 method
  - 构造方法 construtor
  - 其他：代码块 静态代码块 内部类
- 创建对象
  - 类名 对象名 = new 类名();
  - Person p1=new Person();
- 调用类的属性和方法
  - 对象名.成员变量
  - 对象名.成员方法

类的属性

- 属性 field，或者叫成员变量
- 属性用于定义该类或该类对象包含的数据或者说静态属性。
- 属性作用范围是整个类体
- 属性定义格式：
  - [修饰符] 属性类型 属性名 = [默认值]

可以省略。
可以是：public, protected, private
Static, final 。
讲到隐藏和封装时再说。

可以是任何类型，基本类型和引用类型

合法标识符即可。首字母小写，驼峰原则

类的属性

- 在定义成员变量时可以对其初始化
- 如果不对其初始化，Java使用默认的值对其初始化。

基本类型	默认值
boolean	Flase
char	‘\u0000’(null)
byte	(byte)0
short	(short)0
int	0
long	0L
float	0.0f
double	0.0d

局部变量和成员变量

- 区别
  - 声明位置不同 类中 方法中
  - 作用范围不同： 当前类的方法 当前方法
    - 不同的方法中即使有同名的局部变量，没有关系，互不影响，建议相同
  - 内存存放的位置的：栈内存中 堆内存中
  - 成员变量有默认值；局部变量没有默认值

实例变量
的作用域

形参的作用域

局部变量的
作用域

```
class MyClass {
        ……
        实例变量声明
        ……
        public void aMethod(方法形参) {
            ……
            局部变量声明
            ……
        }
        ……
}
```

引用类型

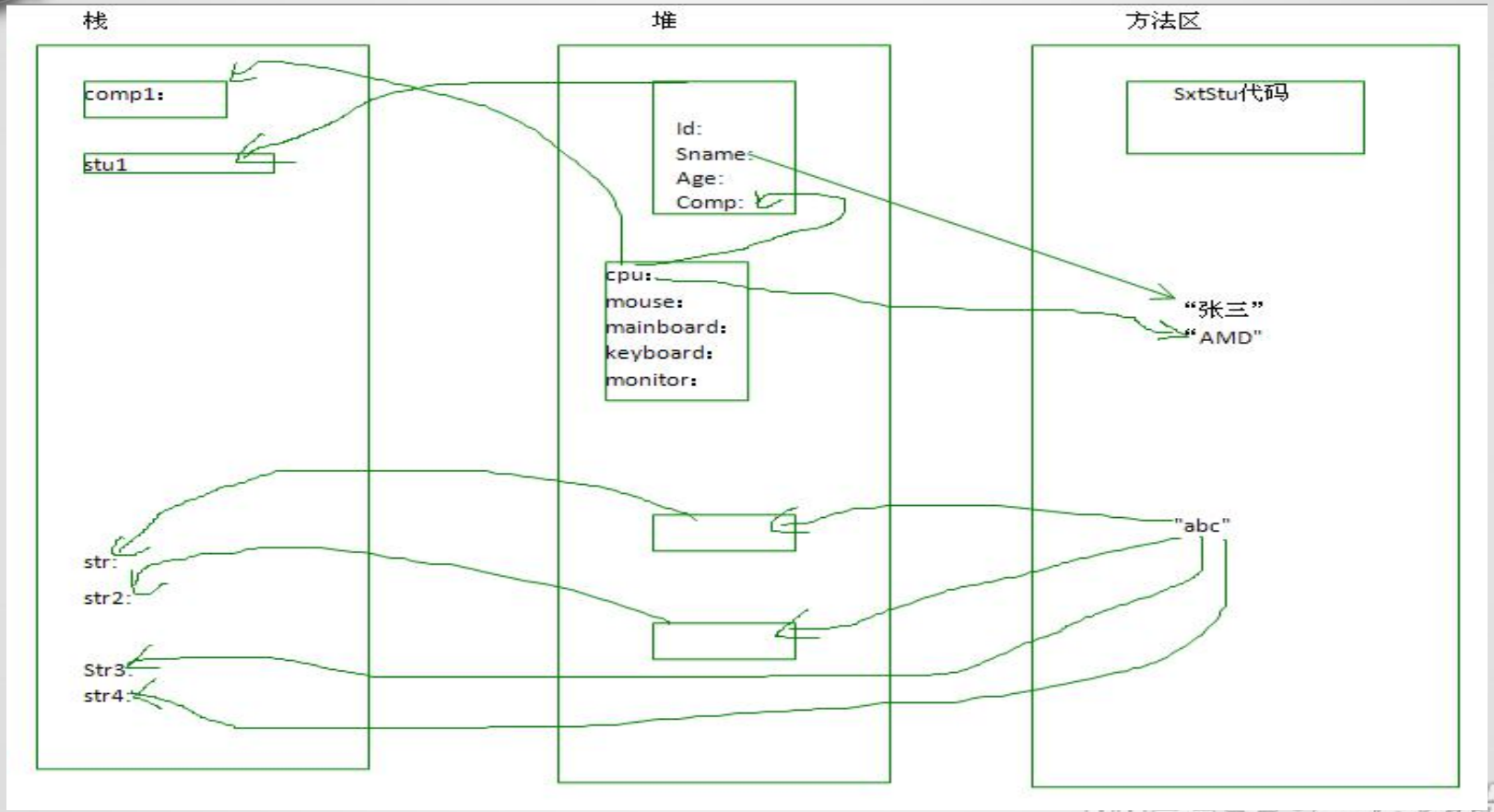
- Java 语言中除基本类型之外的变量类型都称之为引用类型

- Java中的对象和数组是通过引用对其操作的.
  - 引用可以理解为一种受限的指针
  - 指针是可以进行与整数做加减运算的，两个指针之间也可以进行大小比较运算和相减运算。引用不行，只能进行赋值运算。
  - 引用就是一个变量或对象的别名（引用的本质是一个对象）；指针是一个段内存空间的地址(指向存储一个变量值的空间或一个对象的空间)

内存分析

- 栈：
  - 存放：局部变量
  - 先进后出，自下而上存储
  - 方法执行完毕，自动释放空间
- 堆：
  - 存放new出来的对象
  - 需要垃圾回收器来回收
- 方法区：
  - 存放：类的信息(代码)、 static变量、字符串常量等.

内存分析(SxtStu. java)



构造器

- 构造器定义： constructor 构造方法
  - 一个在创建对象时被自动调用的特殊方法。
- 构造器作用：
  - 为对象进行初始化（成员变量）工作。
- 构造器是一种特殊的方法：
  - 构造器的方法名必须和类名一致！
  - 构造器虽然有返回值，但是不能定义返回类型(返回值的类型肯定是本类)，不能在构造器里调用return。
  - 通过new关键字调用！！
  - 如果我们没有定义构造器，则系统会自动定义一个无参的构造方法。如果已定义则编译器不会添加无参数构造方法！
  - 与普通方法一样，构造方法也可以重载

示例

- 定义一个“点”（Point）类用来表示二维空间中的点（有二个坐标）。要求如下：
  - 可以生成具有特定坐标的点对象。
  - 提供可以设置二个坐标的方法。
  - 提供可以计算该“点”距另外点距离的方法。

方法调用

- 形参和实参
  - 定义方法的参数是形式参数
  - 调用方法的参数是实在参数
  - 调用方法时要求参数个数相同，类型兼容

- 参数传递
  - 基本数据类型的参数传递
    - 无法通过方法调用改变变量的值
  - 引用数据类型的参数传递
    - 可以通过方法调用改变变量的值

this关键字

- this的作用:
  - this表示的是当前对象本身，
  - 更准确地说，this代表当前对象的一个引用。
- 普通方法中使用this。
  - 区分类成员属性和方法的形参.
  - 调用当前对象的其他方法（可以省略）
  - 位置：任意
- 构造方法中使用this。
  - 使用this来调用其它构造方法
  - 位置：必须是第一条语句
- this不能用于static方法。（讲完static，大家就知道为什么了！）

this 测试代码

```
public class TestThis {
            int a,b,c;
            TestThis(){
                        System.out.println("正要new一个Hello对象");
            }

            TestThis(int a,int b){
                        //Hello();     // //这样是无法调用构造方法的!
                        this();   //调用无参的构造方法，并且必须位于第一行!

                        a = a;//这里都是指的局部变量而不是成员变量
                        this.a = a;//这样就区分了成员变量和局部变量.   这种情况占了this使用情况的大多数!
                        this.b = b;
            }

            TestThis(int a,int b,int c){
                        this(a,b);     //调用无参的构造方法，并且必须位于第一行!
                        this.c = c;
            }


            void sing(){}

             void chifan(){
                        this.sing();     //sing();
                        System.out.println("你妈妈喊你回家吃饭！");
            }

            public static void main(String[] args){
                        TestThis hi = new TestThis(2,3);
                        hi.chifan();
            }
}
```

static 关键字

- 在类中，用static声明的成员变量为静态成员变量,或者叫做： 类属性，类变量.
  - 它为该类的公用变量，属于类，被该类的所有实例共享，在类被载入时被显式初始化，
  - 对于该类的所有对象来说，static成员变量只有一份。被该类的所有对象共享！！
  - 可以使用”对象.类属性”来调用。不过，一般都是用“类名.类属性”
  - static变量置于方法区中！

- 用static声明的方法为静态方法
  - 不需要对象，就可以调用(类名.方法名)
  - 在调用该方法时，不会将对象的引用传递给它，所以在static方法中不可访问非static的成员。
  - 静态方法不能以任何方式引用this和super关键字

Static示例代码

```
public class TestStatic {
	int a;
	static int width;

	static void gg(){
		System.out.println("gg");
	}
	void tt(){
		System.out.println("tt");
	}

	public static void main(String[] args){
		TestStatic hi = new TestStatic();
		TestStatic.width = 2;
		TestStatic.gg();  //gg();
		hi.gg();    //通过引用也可以访问static变量或static方法。不过，一般还是使用类名.static成员名来访问。
		gg();
	}
}
```

静态初始化块

- 如果希望加载后，对整个类进行某些初始化操作，可以使用static初始化块。
- 类第一次被载入时先执行static代码块；类多次载入时，static代码块只执行一次；Static经常用来进行static变量的初始化。
- 是在类初始化时执行，不是在创建对象时执行。
- 静态初始化块中不能访问非static成员。

```
public class TestStaticBlock {

	static {
		System.out.println("此处，可执行类的初始化工作！");
	}

	public static void main(String[] args) {
		System.out.println("main方法中的第一句");
	}
}
```

pakage

- 为什么需要package?
  - 为了解决类之间的重名问题。
  - 为了便于管理类：合适的类位于合适的包！
- package怎么用?
  - 通常是类的第一句非注释性语句。
  - 包名：域名倒着写即可，再加上模块名，并与内部管理类。
- 注意事项：
  - 写项目时都要加包，不要使用默认包。
  - com.gao和com.gao.car，这两个包没有包含关系，是两个完全独立的包。只是逻辑上看起来后者是前者的一部分。

JDK中的主要包

- java.lang
  - 包含一些Java语言的核心类，如String、Math、Integer、System和Thread，提供常用功能。
- java.awt
  - 包含了构成抽象窗口工具集（abstract window toolkits）的多个类，这些类被用来构建和管理应用程序的图形用户界面(GUI)。
- java.net
  - 包含执行与网络相关的操作的类。
- java.io
  - 包含能提供多种输入/输出功能的类。
- java.util
  - 包含一些实用工具类，如定义系统特性、使用与日期日历相关的函数。

Import

- 为什么需要import?
  - 如果不适用import，我们如果用到其他包的类时，只能这么写：java.util.Date，代码量太大，不利于编写和维护。通过import可以导入其他包下面的类，从而可以在本类中直接通过类名来调用。
- import怎么使用?
  - import java.util.Date;
  - import java.util.*; //导入该包下所有的类。会降低编译速度，但不会降低运行速度。
- 注意要点：
  - java会默认导入java.lang包下所有的类，因此这些类我们可以直接使用。
  - 如果导入两个同名的类，只能用包名+类名来显示调用相关类：
  - java.util.Date date = new java.util.Date();

import static

- 静态导入的作用：用于导入指定类的静态属性
- JDK5.0后增加！

- 如何使用：
  - import static java.lang.Math.*;//导入Math类的所有静态属性
  - import static java.lang.Math.PI;//导入Math类的PI属性
  - 然后，我们可以在程序中直接使用：System.out.println(PI);

总结

- 1.面向过程和面向对象的区别oop
  - 蒋介石和毛泽东 蛋炒饭和盖浇饭
- 2.定义类和创建对象 class object/instance
  - 类的属性 field
  - 类的方法 method
  - 创建对象
- 3.内存结构图
  - 基本数据类型的内存结构图
  - 引用数据类型的内存结构图
- 4.类的构造方法constructor
  - 构造方法的作用
  - 构造方法的特点
  - 构造方法的重载

总结

- 方法调用中的参数传递（重中之重）
  - 基本数据类型的参数传递：不能改变参数的值
  - 引用数据类型的参数传递：不能改变参数的值
- this
  - This代表当前对象自身的引用（必须new）
  - This可以修饰属性，区别成员变量和局部变量
  - This修饰方法
  - This修饰构造方法（必须是第一条语句）
- static
  - static变量：只有一份，属于类，可以类名. Static变量
  - static方法: 类名. Static方法，不能出现this和super
  - static代码块：只执行一次，最早执行的（类第一次调用）
- package import
  - 包：作用
  - 导入：import com.bjsxt.oop.*;
  - 静态导入： import static java.lang.Math.PI;