Java

一、变量的定义

1、数据类型

```
字符型：char
整型：int/long
双精度型：float/double
字符串型：String
布尔类型：boolean----true/false
```

2、应用举例

```
class VariableTest{
    public static void main(String[] args){
        //直接定义类型和赋值
        int myAge=12;
        //先定义类型，再赋值
        int myNumber;
        myNumber=1001;
    }
}
```

二、打印

```
public class HelloWorld {
    public static void main(String[] args) {
        int a=12;
        int b=13;
        System.out.println(a+b);
    }
}

>>>>25
```

```
public class HelloWorld {
    public static void main(String[] args) {
        String number="号码";
        int i=1001;
        System.out.println(number+i);
    }
}

>>>>号码1001
```

三、运算符

1、算术运算符

```
1、%：取余数，相当于mod

2、++:在自身的基础上加1

3、--：在自身的基础上-1

注意：int类型的数据两两相除，只能得到整数，例如：

int a=20;

int b=15;

System.out.println(a/b);

输出结果为：1

解决方案：将数据强制转换成浮点数，比如将代码改成：System.out.println(a/（b*1.0）);此时b被转换成浮点数。相除之后结果也将成为浮点数。
```

2、逻辑运算符

```
1、&&：且

2、||:或

3、!:非
```

3、比较运算符

```
1、==：相等

2、！=：不等

3、+=：相当于a=a+i-------a+=i

4、-=：相当于a=a-i-------a-=i

5、%=：相当于a=a%i------a%=i
```

四、流程控制

1、顺序结构

2、选择结构（分支结构）

```
//if循环

//if-else if
public class HelloWorld {
    public static void main(String[] args) {
        int i=1;
        if (i>1) {
            System.out.println("答对了");
        }
        else if(i<=1) {
            System.out.println("答错了");
        }
    }
}

//if-else
public class HelloWorld {
    public static void main(String[] args) {
        int i=1;
        if (i>1) {
            System.out.println("答对了");
        }
        else{
            System.out.println("答错了");
        }
    }
}
```

```
//使用scanner做交互（需要引入依赖）
import java.util.Scanner;
public class HelloWorld {
    public static void main(String[] args) {
        //定义Scanner
        Scanner scan=new Scanner(System.in);
        //提示语句
        System.out.println("请输入数字");
        //定义输入对象的数据类型（next字符串，nextInt整数）
```

```
        int num=scan.nextInt();
        //定义具体的操作
        System.out.println("输入数字为："+num);
    }
}
```

3、循环结构

（1）结构

```
1、for循环

public class HelloWorld {
    public static void main(String[] args) {
        int s=0;
        for(int i=1;i<=100;i=i+1) {
            s=s+i;
        }
        System.out.println("和为："+s);
    }
}

2、while循环

int i=0;

while(i<=5){

System.out.println(i);

i=i+1;

}
```

（2）关键字

break\continue：都是使用在循环语句中

五、数组

1、申明一维数组

```
public class HelloWorld {
    public static void main(String[] args) {
        //一维数组申明和初始化
        //静态初始化
        int[] number =new int[]{1001,1002,1003};
        //动态初始化，这里并没有赋值，String表示之后输入的数组的数据类型，[5]表示数组内将来有5个值
        String[] names=new String[5];
    }
}
```

2、调用指定位置的元素

```
//调用number数组中2号位的元素值（数组中的第三个数字）
System.out.println(number[2]);
>>>>1003
```

3、获取长度

```
System.out.println(number.length);
>>>>3
```

4、遍历数组

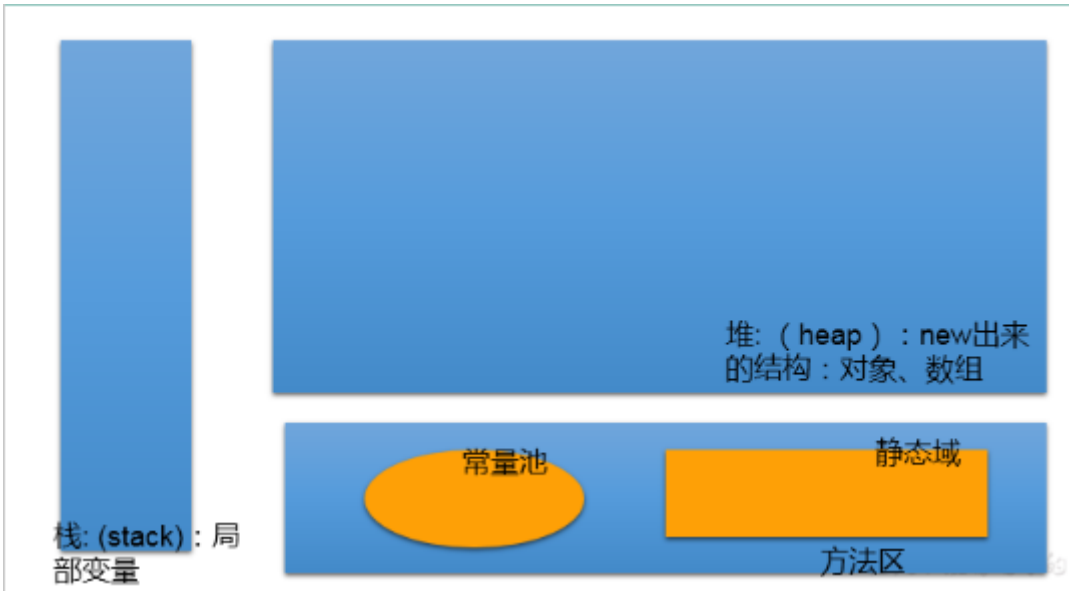
```
for(int i=0;i<names.length;i=i+1) {
System.out.println(names[i]);
}
```

5、动态数组赋值

```
names[0]="xiong";
names[1]="fan";
names[2]="xiong2";
names[3]="fan2";
names[4]="xiong3";
```

思考题：一维数组的内存解析是怎么样的

6、内存解析



栈：

堆

六、面向对象

成员：属性、方法、构造器、代码块、内部类

特征：封装、继承、多态

前言：区分类中的属性和局部变量

类的定义	属性（称员变量）	局部变量
相同点	1格式相同：数据类型 变量名=变量值 2都有对应的作用域	
不同点	1.直接定义在class的{}中 2.属性可以申明权限、使用权限修饰符 3.属性，根据定义的类型，有默认值，例如int默认0，float默认0.0,String默认null,boolean默认false	1.定义在main方法的{}中 2.局部变量需要在定义的过程中就赋值

1、创建class类

```
class person{
     //属性
    String name;
    boolean isMale;
    //方法：权限权限修饰符+返回值状态+方法名
    public void eat() {
        System.out.println("吃饭");
    }
    public void sleep() {
        System.out.println("读书");
    }
    public void talk(String language) {
        System.out.println("人使用的语言是："+language);
    }
}

//注意：void表示没有返回值
```

2、调用class类

```
public class HelloWorld {
    public static void main(String[] args) {
        //实例化person类的对象
        person p1=new person();
        //调用对象的属性
        p1.name="Tom";
        p1.isMale=true;
        System.out.println(p1.name);
        //调用方法
        p1.eat();
        p1.sleep();
        p1.talk("中文");
    }
}
```

3、方法定义注意事项

```
方法定义：权限类型+返回值类型+方法名
（1）权限类型：private,public,缺省,protected
（2）返回值：无返回值--void，有返回值--直接定义返回值类型（String,int,boolean等）并且需要return关键字
    //无返回值
    public void eat(){
    }
    //有String类型的返回值，方法中可以调用属性
    public String getName(){
        return name;
    }
```

4、构造方法

（1）作用：一方面创建对象的时候用， 另一方面构造方法在调用对象的时候会自动执行构造方法，当需要自动执行某个功能省去调用的化，可以写一个构造方法。（可以理解为初始化值 ）

（2）格式：public +与类同名

```
public class person {
    String name;
    int age;
    public static void main(String[] args) {
        //实例化person类的对象
        person p1=new person("Tom");
    }
    public person(String n){
        name=n;
    }
}

//这里的person()就是一个构造器，用于创建p1这个对象
//new person("Tom")表示，每次调方法的时候都会默认是name="Tom"而不需要再用p1.name="Tom"这样去复制了
```

```
//（练习）使用构造方法定义一个动物类
package com.unit8.pojo;
public class Dog {
   //定义类的属性
   String nick;
   String color;
   int age;

   //定义构造方法
   public Dog(String nick,String color,int age){//public后面直接接类名,不需要void和static
       this.nick=nick;
       this.color=color;
       this.age=8;

   }

   public static void main(String[] args) {
```

```
18
19          //对象实例化，并且赋值,有构造方法的好处是，在对象实例化的时候一定会调用构造方法
20          Dog dog=new Dog("小白","白色",8);
21
22          //获取对象的属性值
23          System.out.println(dog.nick);
24          System.out.println(dog.color);
25          System.out.println(dog.age);
26      }
27  }
28
```

5、匿名对象

```
1  //有名的对象
2  person p=new person();
3  //匿名的对象，不要定义变量名称，直接在new的对象后面调用方法,但是匿名对象是一次性的
4  new person().eat();
```

6、方法重载

同一个类中，允许存在相同方法名，只要参数个数或类型不同即可

重载的作用是调一个方法名称可以输入不同的参数，方便好记，灵活性高

System.out.println();就是典型的方法重载

```
1  //以下方法之间属于方法重载
2  public void getSum(int i,int j){
3      System.out.println(i+j);
4  }
5  public void getSum(int i,int j,int k){
6      System.out.println(i+j+k);
7  }
8  public void getSum(String i,int j){
9      System.out.println(i+j);
10 }
11 //特殊写法，这种写法表示可以输入0~无限多个String类型的参数
12 //格式：数据类型+“...”+变量名称
13 publiv void getSum(String ... strs){
14     System.out.println("这里可以输入0~无限多个的参数");
15 }
```

7、封装性

隐藏对象内部的复杂性，只对外暴露公开的几个单一的功能选项，用private关键字限制属性，用set/get方法限制和提取属性值

```
1  class Animal_test{
2      String name;
3      int age;
4      private int legs;//私有化
5
6      //封装，把leg属性封装起来并且限制,提前是把legs先私有化
7      public void setLegs(int l) {
```

```
        if(l>=0 && 1%2 ==0) {
            legs=l;
        }else {
            legs=0;
        }
    }
    //调取封装的legs属性，当要查看属性的值的时候使用
    public int getLegs(){
        return legs;
    }
    public void eat() {
        System.out.println("动物进食");
    }
    public void show() {
        System.out.println("name:"+name+"age:"+age+"legs:"+legs);
    }
}
```

七、This

作用：在get和set方法中，传入参数的时候往往为了见名知意，形参往往用属性名，为了在set方法中区别开属性和形参的关键字的意义，在set的属性前加上this关键字

```
public class PersonTest {
    public static void main(String[] args) {
        Person p1 = new Person();
        p1.setAge(1);
        System.out.println(p1.getAge());

    }
}

class Person{
        private String name;
        private int age;

        public String getName() {
            return name;
        }
        //this.name表示属性,为了和形参name区分
        public void setName(String name) {
            this.name = name;
        }

        public int getAge() {
            return age;
        }
        //this.age表示属性，为了和形参age区分
        public void setAge(int age) {
            this.age = age;
        }
}
```

八、MVC设计模式

常用的设计模式之一，将整个程序分为三个层次

模型层：model

视图层：view

控制层：controller

九、继承性

子承父类之后，子类就继承了父类所有的属性、方法，但是一个父类对应多个子类继承，不能出现一个子类继承多个父类的情况

```
//关键字extends
public class Student extends Person{
    String name;
    int age;
    String major;
    public static void main(String[] args){
        Student stu=new Student();
        stu.eat();//这里假设eat()方法是父类Person的，需要继承，否则需要重新写
    }
}
//私有属性（private）也被继承了，需要通过get/set调取
```

十、方法的重写

用途，子类继承父类之后，对父类同名同参数的方法（方法名、权限和形参要一样），进行覆盖操作。要区分方法的重载。

十一、super关键字

当子类继承父类之后，子类重写了父类的方法，当还需要调父类的属性的时候，为了区分属性还是父类的需要在父类属性中加入super关键字表示区分,也可以调用父类的

```
class bank extends person{
    String name;
    int number;
    @Override
    public void sleep() {
        // TODO Auto-generated method stub
        System.out.println("name"+this.name+"age"+super.age);
        System.out.println("name"+super.name+"age"+this.age);
    }
}
//这里的super.age是指父类的age,super.name是父类的属性

//super()表示这个构造方法是父类Culinder
public class Circle extends Cylinder{
    public Cylinder(){
        super();
    }
}
```

十二、多态

是什么：父类的引用指向子类的对象，多态的这种方式不能调子类特有的方法，只能调父类在子类中重写的方法，多态的使用前提，需要有父子类的继承关系才行。对象的多态只适用于方法，不适用于属性。

```
public class PersonTest{
    public static void main(String[] args){
        Person p2=new Man();
        Person p3=new Woman();
        p2.eat();
        p2.walk();
    }
    //这里的person类是父类，man和woman类分别是person类的子类
    //结果，调用的eat和walk方法是子类的
    //但是多态的这种方式不能调子类特有的方法，只能调父类在子类中重写的方法
}
```

多态的向下转型

```
//将p2原本为父类person类的对象强制转换成Man类型方法如下
Man m2=(Man)p2;
//instanceof关键字的使用,判断对象是否是woman类的实例，如果是的话强转成woman类。
//instanceof类的作用是避免强转的过程中出现报错
if(p2 instanceof Woman){
    Woman w1=(Woman)p2;
    w1.goShopping();
    System.out.println("*********Woman**************");
}

```

十三、获取变量的数据类型

```
public class Test{
    public static void main(String[] args) {
        short a = 1;
        a += 1;
        //打印数据类型
        System.out.println(getType(a));
    }
    public static String getType(Object obj) {
        return obj.getClass().getName();
    }
}
```

十四、数据类型之间的转换

```
public class WrapperTest {
    //基本数据类型转换成包装类
    //1.数值转换成字符串
    @Test
    public void test1() {
        //先把基本数据类型转换成包装类
        int num1=10;
```

```
        Integer in1=num1; //Integer是int的包装类
        //在包装类中做数据类型的转换
        String num2=in1.toString();
        System.out.println(num2);
    }

    //2.字符串转换成数值
    @Test
    public void test4() {
        String str1="123";
        int num3=Integer.parseInt(str1);
        System.out.println(num3);
    }

    //3.字符串转换成布尔类型
    String str2="true";
    boolean b1=Boolean.parseBoolean(str2);
    System.out.println(b1);
}
```

十五、三元运算符

```
public class InterviewTest{
    @Test
    public void test1(){
        Object o1=true ? new Integer(1) : new Double(2.0);
        System.out.println(o1);
    }
}

//表示，o1是否为true，如果是的话执行new Integer(1),否则执行Double(2.0)
```

十六、Static关键字

作用：

(1)static修饰属性，所有对象都将拥有这个属性的值，例如c1.nation="CHN",那么c2.nation也将被默认为“CHN”

(2)static修饰方法，可以直接通过类的形式调用

注意点：

静态方法中不能使用this关键字和super关键字

静态的方法中，只能调用静态的方法或属性，非静态方法中，既可以调用非静态的方法或属性，也可以调用静态的方法或属性

```
public class StaticTest{
    public static void main(String[] args){
        person c1=new person();
        c1.nation="CHN";
        System.out.println(c2.nation);
        //此时c2.nation的值也为"CHN",
    }

    public void person(){
```

```
        static String nation;
        int age;
        String name;
    }
}
```

十七、final关键字

final关键字修饰的类不能被其他类继承；如果是修饰变量的话，此时的变量将会变成一个常量

```
final class FinalA{
}
//FinalA不能被其他类继承
```

十八、抽象类与抽象方法

如果一个类中没有包含足够的信息来描绘一个具体的对象，这样的类就是抽象类。

抽象类除了不能实例化对象之外，类的其它功能依然存在，成员变量、成员方法和构造方法的访问方式和普通类一样。由于抽象类不能实例化对象，所以抽象类必须被继承，才能被使用。目的是圈定一种规范和方法，限制使用在Java中抽象类表示的是一种继承关系，一个类只能继承一个抽象类

```
public abstract class Employee
{
   private String name;
   private String address;
   private int number;
   public Employee(String name, String address, int number)
   {
      System.out.println("Constructing an Employee");
      this.name = name;
      this.address = address;
      this.number = number;
   }
   public double computePay()
   {
     System.out.println("Inside Employee computePay");
     return 0.0;
   }
   public void mailCheck()
   {
      System.out.println("Mailing a check to " + this.name
       + " " + this.address);
   }
   public String toString()
   {
      return name + " " + address + " " + number;
   }
   public String getName()
   {
      return name;
   }
   public String getAddress()
   {
      return address;
   }
```

```
    public void setAddress(String newAddress)
    {
       address = newAddress;
    }
    public int getNumber()
    {
      return number;
    }
}
```

十九、接口关键字

使用interface关键字来定义，在java语言中接口和类是并列的结构，接口中不能定义构造器，所以接口不能实例化。在java开发当中通过让类去实现（implements）的方式调用接口。如果实现类覆盖了接口中的所有抽象方法，则此实现类就可以实例化，如果实现类没有覆盖接口中的所有抽象方法，则次实现类仍为一个抽象类，接口实际上是提供了一个规范。接口的使用过程如下：定义接口-->>创建类继承并重写接口-->>调用接口。接口中所有的方法必须是抽象方法

```java
package com.xiong.contact;

public class interfaceTest {
    public static void main(String[] args) {
        //调用接口Flyable
        Plane plane=new Plane();
        plane.fly();
        plane.stop();
    }
}

interface Flyable{
    //全局常量
    public static final int MAX_SPEED=7990;
    public static final int MIN_SPEED=1;
    //抽象方法
    public abstract void fly();
    public abstract void stop();
}

//实现Flyable的接口
class Plane implements Flyable{
    @Override
    public void fly() {
        System.out.println("通过引擎起飞");
    }

    @Override
    public void stop() {
        System.out.println("驾驶员减速停止");
    }
}

```

二十、异常处理

异常出现的原因，用户输入错误，网络错误，文件不存在导致程序出现异常情况。

异常处理的目的，避免外部操作错误导致系统崩溃。

异常处理方式有两种，一种是try-catch-finally作用是可以让编译时不报错，但是不保证运行时不报错

```java
//第一种异常处理方式，try-catch-finally
public class ExceptionTest1 {
    public static void main(String[] args) {
        String str="123";
        str ="abc";
        try {
            int num =Integer.parseInt("");
        } catch (Exception e) {
            System.out.println("程序出现异常了，不要着急");
        }
        finally {
            //一定会执行的代码
            System.out.println("不管你解决没有，我都支持你。");
        }
    }
}
```

```java
//throws异常抛出处理方式
package com.xiong.contact;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;

public class exceptionTest2 {
    public void method1() throws IOException {
        File file=new File("hello.txt");
        FileInputStream fis=new FileInputStream(file);
        int data=fis.read();
        while (data !=-1) {
            System.out.println((char)data);
            data=fis.read();
        }
        fis.close();
    }
}
```

二十一、Maven

是什么：是一种服务于java平台的自动化构建工具

为什么：解决依赖包的调用，避免手动准备jar包等依赖，如果一个项目非常的庞大（有多个工程）就不适合用package划分项目模块，否则显得非常臃肿。