01 绪论

```
public class HelloWorldApp{//类名
    public static void main(String args[]){//程序入口点，C-Style写法不推荐

        //System.out.println(obj)用于输出自带换行的obj.toString()/null
        System.out.println("Hello World!");
    }
}
```

Java 基础语法-1

```
public class IncrementDecrement {
    public static void main(String[] args) {
        int i = 0; //定义一个int基本类型的变量i并赋值0
        System.out.println(i++);//0
        System.out.println(++i);//2
        System.out.println(i--);//2
        System.out.println(--i);//0
    }
}


public class IncrementDecrement2 {
    public static void main(String[] args) {
        int i = 0;
        i++;
        System.out.println(i);//1
        ++i;
        System.out.println(i);//2
        i--;
        System.out.println(i);//1
        --i;
        System.out.println(i);//0
    }
}


class If_NoElse{
    public static void main(String[] args){
        int scoreOfMark=89;
        if(scoreOfMark>=90){//如果 scoreOfMark>=90 则输出 祝贺你,你的成绩优秀
            System.out.println("祝贺你,你的成绩优秀");
        } else {//如果 scoreOfMark<90 则输出 你的成绩未达到优秀，请努力!
            System.out.println("你的成绩未达到优秀，请努力!");
        }
    }
}


public class WhileOfSum{
    public static void main(String[ ] args) {
        int i,sum;
        sum=0; //累加器清0
        i=1; //i的初始值为1
        //计算 1+2+...+10 的值
        while (i<=10){
            sum+=i;
            i++;
        }
        System.out.println("sum="+sum); //输出sum
    }
}

public class ForOfSum{
    public static void main(String[ ] args)  {
        int i,sum;
        sum=0; // 累加器清0
        //计算 1+2+...+10 的值
        for(i=1;i<=10;i++){
            sum+=i;
        }
        System.out.println("sum="+sum);
```

```java
    }
}

class SwitchDemo{
    public static void main(String[] args) {
        int testScore=88;
        char grade;
        switch (testScore/10) {//两个整型数相除的结果还是整型
            case 10://此处没有使用break，如果没有break则会一直往下执行
            case 9:
                grade='A';break;//值为10和9时的操作是相同的
            case 8:
                grade='B';break;
            case 7:
                grade='C';break;
            case 6:
                grade='D';break;
            default://默认，当上面的情况都不满足时
                grade='E';break;
        }
        System.out.println("grade is:"+grade);
    }
}

public class BreakDemo {
     public static void main(String[] args) {
        int index=0;
        while (index<=100) {
            index+=10;
            if (index==40)
                break;
           //当index的值大于100时，循环将终止。但有一种特殊的情况，如果index的值等于40，循环也将立即终止。
            System.out.println("The index is "+index);
        }
    }
}

public class ContinueDemo{
    public static void main(String[] args){
        int index=0;
        while(index<=99){
            index+=10;
            //当index的值等于40时，使循环回到while语句处，  而不像正常处理那样去执行后面的输出语句
            if(index==40)
                continue;
            System.out.println("The index is "+index);
        }
    }
}

public class JHelloWorld2{
    //args 存储输入的多个参数
    public static void main(String[] args){
        System.out.println(
            "第一个参数：  "+args[0]+
            "  第二个参数：  "+args[1]+
            "  第三个参数：  "+args[2]
        );
    }
}

public class JHelloWorld3{
    public static void main(String[] args){
        //array是一个数组
        String[] array={"abc","d","ef"};
        System.out.println(
            "第一个参数：  "+array[0]+
            "  第二个参数：  "+array[1]+
            "  第三个参数：  "+array[2]
        );
    }
}


import javax.swing.JOptionPane; //导入JOptionPane类
```

```
public class JHelloWorld4 {
    public static void main(String[] args) {
        //显示一个对话框窗口并提供输入字符串
        String ss = JOptionPane.showInputDialog("请输入一个数", "");
        System.out.println("输入参数为：  " + ss);
    }
}

import java.io.IOException;
public class JSysteminReadTest {
    //throws 声明会抛出 IOException
    public static void main(String[] args) throws IOException{
        //定义一个数组
        byte[] b = new byte[100];
        //System.in.read(b) 可以将控制台输入的内容读入缓冲区，返回 count 为字符数
        int count = System.in.read(b);
        //遍历b
        for (int i = 0; i <= count - 1; i++)
            //(char)b[i]将byte强制转换为char
            System.out.print((char) b[i]);
    }
}

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
public class JBufferedReaderTest{
    public static void main(String[] args) throws IOException{
        String ss;
        int a;
        //使用BufferReader通过标准输入流读取控制台输入
        BufferedReader buf = new BufferedReader(new InputStreamReader(System.in));
        System.out.println("请输入一个数：");
        //获取输入的一行
        ss = buf.readLine();
        //将String通过Integer类的方法转换为Int, 该操作可能会出现异常
        a = Integer.parseInt(ss);
        System.out.println("输入的数为：" + a);
    }
}

import java.util.Scanner; //导入Scanner类
public class JScannerTest{
    public static void main(String[] args){
        Scanner scan = new Scanner(System.in);
        int a;
        System.out.println("请输入数据:");
        //scan.nextInt() 会将输入的数字返回
        a = scan.nextInt();
        System.out.println("输入的数据是：\n" + a);
    }
}
```

03-1类和对象

@null

03-2类和对象

```
class Line{
    private int a,b;
    public Line (int x,int y){
        this.x=x;
        this.y=y;
    }
}

public class StaticVar{
    //声明静态成员变量
    public static int number;
}
public class Otherclass{
    public void method(){
        //可访问StaticVar类中的public静态成员变量
        int x = StaticVar.number;
    }
}

public class GeneralFunction{
    //定义一个静态成员方法，可供类内外调用
    public static int addUp(int x, int y){
        return x+y;
    }
}
public calss UseGeneral{
    public void method(){
        //调用GeneralFunction的静态方法addUp()
        int c = GeneralFunction.addUp(9,10);
    }
}

class AccessVar{
    int x;
    //静态方法无法访问非静态成员变量
    public static void setX(){
        x=9;
    }
}


public class Book{
    public int id; // 书的编号
    public static int bookNumber = 0; // 书的总数

    public Book(){
        bookNumber ++;
    } // Book构造方法结束

    public void info(){
        System.out.println("当前书的编号是：" + id);
    } // 方法info结束

    public static void infoStatic(){
        System.out.println("书的总数是：" + bookNumber);
    } // 静态方法引用静态变量

    public static void main(String args[]){
        Book a = new Book();
        Book b = new Book(); // 声明并实例化a, b
        a.id = 1101;
        b.id = 1234; //为a, b设定id
        System.out.print("变量a对应的");
        a.info();
        System.out.print("变量b对应的");
        b.info();
        Book.infoStatic(); //调用Book的静态方法
        System.out.println("比较(a.bookNumber==Book.bookNumber)"
            + "的结果是：" + (a.bookNumber==Book.bookNumber));
        System.out.println("比较(b.bookNumber==Book.bookNumber)"
            + "的结果是：" + (b.bookNumber==Book.bookNumber));
    } // 方法main结束
} // 类Book结束

class MyClass{
    int i = 0;
    public int myMethod(int x){
        return i+x;
    }
    public static void main(String[] args) {
        //静态方法无法访问非静态成员变量和方法
        System.out.println(myMethod(10));
    }
}


public class FinalEx {
     public static void main(String args[]){
        final StudentTest stu1=new StudentTest(22,"Tom");
        StudentTest stu2=new StudentTest(25,"Jerry");
        //stu1=stu2;就会出错：final修饰的引用无法更改
        System.out.println("stu1的name:"+stu1.sname+" stu1的age:"+stu1.sage);
        // 调用setter
        stu1.setSname("Jerry");
        stu1.setSage(25);
        //这时stu1指向的对象发生了改变
        System.out.println("stu1的name:"+stu1.sname+" stu1的age:"+stu1.sage);
    }
}
class StudentTest{
    String sname;
    int sage;
    // 两个参数的构造方法
    public StudentTest(int sage,String sname){
        this.sage=sage;
        this.sname=sname;
    }
    // 定义sname和sage的setter
    public void setSname(String sname){
        this.sname=sname;
    }
    public void setSage(int sage){
        this.sage=sage;
    }
}

class StudentFinal{
    private final long StudentID;
    private static long number=2007030801;
    public StudentFinal (){
        //final 变量仅可赋值一次
        StudentID = number++;
    }
    public long getID(){
        return StudentID;
    }
    public static void main(String[] args){
        //创建StudentFinal数组
        StudentFinal[] s = new StudentFinal[5];
        for (int i=0; i<s.length; i++){
            //实例化对象
            s[i]=new StudentFinal();
            System.out.println("The StudentID is "+s[i].getID());
        }
    }
}

class StudentFinal{
    private final long StudentID=20120101;
    private static long number=2007030801;
    public StudentFinal (){
        //StudentID无法改变
        StudentID = number++;
    }
    public long getID(){
        return StudentID;
    }
    public static void main(String[] args){
        StudentFinal[] s = new StudentFinal[5];
        for (int i=0; i<s.length; i++){
            s[i]=new StudentFinal();
            System.out.println("The StudentID is "+s[i].getID());
        }
    }
}

class StudentFinal{
    private final long StudentID;
    private static long number=2007030801;
    public StudentFinal (){
        StudentID = number++;
    }
    public long getID(){
        return StudentID;
    }
    public long ChangeID(){
        //StudentID无法改变
        StudentID =StudentID+1;
        return StudentID;
    }
    public static void main(String[] args){
        StudentFinal[] s = new StudentFinal[5];
        for (int i=0; i<s.length; i++){
            s[i]=new StudentFinal();
            System.out.println("The StudentID is "+s[i].getID());
        }
    }
}

package firstpackage;
//导入java.util.*下的包，含有Calendar类
import java.util.*;
public class Date{
    private int year,month,day;
    //三个参数的构造，初始化
    public Date(int y,int m,int d){
        year=y;
        month=m;
        day=d;
    }
    //无参构造
    public Date(){}
    public int thisyear(){
        //获得Calendar对象的YEAR字段，即今年年份
        return Calendar.getInstance().get(Calendar.YEAR);
    }
}

import firstPackage.Date;
class Person{
    String name;
    int age;
    public Person(String na,int ag){
        name=na;
        age=ag;
    }
    public Person(){}
    //获取出生年份
    public int birth(int y){
        return y-age+1;
    }
}
public class PersonDemo{
    public static void main(String[] args){
        Person ps=new Person("Tom",21);
        Date now=new Date();
        int y=now.thisyear();
        System.out.println(ps.name+" was born in "+ps.birth(y));
    }
}
```

03-2类和对象-1

```
public class Triangle
{
    double length=10.0;
    double height=5.0;
    double area()
    {
        return length*height/2.0;
    }
    public static void main (String args[])
    {
        double s;
        s=(new Triangle()).area();
        System.out.println("该三角形的面积是: " + s);
    }
```

```java
}

public class Triangle {
    double length=10.0;
    double height=5.0;
    double area() {
        //计算面积
        return length*height/2.0;
    }
    public static void main (String[] args) {
        double s;
        //调用area方法
        s=(new Triangle()).area();
        System.out.println("该三角形的面积是： " + s);
    }
}

public class Student {
    String name;
    char sex;
    int stuID;
    //多余的空构造
    public Student(){}

    public Student(String stuName,char sex,int stuID){
        name=stuName;
        this.sex=sex;
        this.stuID=stuID;
    }

    //setter
    public void setName(String stuName){
        name=stuName;
    }
    public void setSex(char sex){
        this.sex=sex;
    }
    public void setStuID(int stuID){
        this.stuID=stuID;
    }
}

class Student {
    String name;
    char sex;
    int stuID;
    public Student(){}
    public Student(String stuName,char sex,int stuID){
        name=stuName;
        this.sex=sex;
        this.stuID=stuID;
    }
    public void setName(String stuName){
        name=stuName;
    }
    public void setSex(char sex){
        this.sex=sex;
    }
    public void setStuID(int stuID){
        this.stuID=stuID;
    }
    public static void main(String[] args){
        Student[] s=new Student[30];
        for(int i=0;i<s.length;i++){
            s[i]=new Student();
        }
    }
}
public class TestStudent{
    public static void main (String[] args) {
        Student s1=new Student();
        //调用setter赋值
        s1.setName("张三");
        s1.setSex('男');
        s1.setStuID(20080201);
        //用含有三个参数的构造器实例化
        Student s2=new Student("李四",'男',20080301);
    }
}

public class PassTest{
    float ptValue;
    //参数类型是基本类型
    public void changeInt(int value){
        value = 55 ;
    }
    //参数类型是引用型，并且方法中改变参数的值
    public void changeStr(String value){
        value = new String("different");
    }
    //参数类型是引用型，并且方法中改变了参数所指向对象的成员变量值
    public void changeObjValue(PassTest ref){
        ref.ptValue = 99.0f;
    }
    public static void  main(String[] args){
        String str;
        int val;
        //创建PassTest的对象
        PassTest pt= new PassTest();
        //测试基本类型参数的传递
        val = 11;
        //val不变
        pt.changeInt(val);
        System.out.println("Int value is: " +val);
        //测试引用类型参数的传递
        str = new String("Hello");
        //str不变
        pt.changeStr(str);
        System.out.println("Str value is: " +str);
        //测试引用类型参数的传递
        pt.ptValue = 101.0f;
        //ptValue会变
        pt.changeObjValue(pt);
        System.out.println("Pt value is: " +pt.ptValue);
    }
}
```