JAVA(J2SE/J2EE)学习笔记

JAVA(J2SE)基础

一、JAVA 语法基础

标识符 -date:090811

1.什么叫标识符？

就是一个名字

2.在JAVA中，遇见的标识符有类名、方法名和变量名。

3.标识符的命名规则：

i. 标识符有字母、数字、下划线（_）以及美元（$）符号组成。
ii. 标识符不能以数字开头
iii. 不能和 JAVA 中的关键字或者保留字重名

4.标识符的命名规范：

i. 最好不要一下划线（_）或者美元（$）符号开头
ii. 如果标识符是类名：首字母应该大写；
如果有多个单词组成，那么每个字符的首字符都大写。

例如：

```
class Student{
}
class HelloWorld{
}
```

iii. 如果标识符是方法名或者变量名：首字母应该小写；
如果有多个单词组成，那么从第二个单词开始，首字母大写。

例如：

```
String name;        //用户名
int stuAge;         //学生的年龄
```

iv. 标识符最好有一定的意义。

```
int 整数 = 1;
```

v. 如果是常量名，那么全部字母都大写。

final double PI = 3.1415926;

关键字 -date:090811

1.什么叫关键字？

具有特殊意义的一些单词。

2.举例说出几个关键字

byte short int long float double char boolean
public private protected
void
return
true false
static final abstract
if else
switch case
for
while
do
try catch
continue
break

数据类型的转换 -date:090811

1.基本数据类型可以相互转换(除boolean)

2.可以将低精度的数据类型直接赋值给高精度的数据类型

```
int i    = 1;
short s = 1;
i = s;            //可以自动进行转换
```

3.当把高精度的数据类型赋值给低精度的数据类型时，必须强制类型转换。

```
int i = 1;
short s = 1;
s = （short）i;                //强制类型转换
```

4.整数默认类型是int

例如

```
short s = 1;
s = s + 1;        //报错，不能从 int 类型转换成 short 类型
```

//因为表达式中的 1 为 int 类型，在计算时，s 自动转换成 int 类型，之后再进行计算。
//所以，返回值应该是 int 类型

```
float f = 1.1;
int i = 1;
i = f + 1;
```

5.小数默认类型的double数据类型

包的定义规范 -date:090812

1.为什么定义包？

解决名字空间的问题。

2.在同一个包中，类名是唯一。

3.在同一个项目中，包也是不能重复的。

4.包的命名规范：

com/org + "." + 公司的名字 + "." + "项目的名字" + "." + "模块的名字" + "." + "功能点的名字"

5.包的声明：

package 包的名字; //要放在源文件的最上面（除注释外，因为注释可以放在 package 上面）

6.引入外部类的方法

import 包的名字.类的名字;
import 包的名字.*;

数组 -date:090812

1.数组的声明

声明一维数组：

i. 数据类型 数组名[]; //基本数据类型和引用数据类型
ii. 数据类型[] 数组名;

声明二维数组：
i. 数据类型 数组名[][]; //基本数据类型和引用数据类型
ii. 数据类型[][] 数组名;

2.创建数组：new

创建一位数组：
i.数据类型 数组名[] = new 数据类型[数组长度];
ii.数据类型[] 数组名 = new 数据类型[数组长度];

例如：

```
/*
    声明一个 int 类型的数据，数组的长度为 5
*/

int intArray[] = new int[5];

int intArray2[];

intArray2[] = new int[6];
```

创建二维数组：
i.数据类型 数组名[][] = new 数据类型[行数][列数]; //行数必须要有，列数可以没有
ii.数据类型[][] 数组名 = new 数据类型[行数][列数];

3.数组的初始化：

初始化一个一维数组：
i.
数据类型[] 数组名 = {数据 1,数据 2,...};

ii.
数组类型[] 数组名 = new 数据类型[]{数据 1,数据 2,...};

iii. 初始化一个二维数组：

4.数组的使用：

取出数组中的数据：
数据类型 变量 = 数据名[下标]

变量数组：length ：返回数组的长度

5.一维数组总结：

一维数组在创建时，系统会对其进行默认初始化。
整数型数组默认初始化为 0；
浮点型数组默认初始化为 0.0；
char 类型数组默认初始化为空字符；
boolean 数组默认初始化为 false；
引用数据类型默认初始化为 null；

二、面向对象

类和方法 -date:090813--090814

1.什么对象？

描述：

对象具有标识、属性和行为；
对象是唯一的。
是类的一个实例。

2.什么是类？

描述：

类具有标识、属性和行为。
类是一类具有相同属性和行为的对象的抽象。

3.类和对象的关系？

i.类是对象的抽象
ii.对象是类的一个实例

创建类的一个实例的方式：

类名 对象名 = new 类名();

4.类里面包含的成员？

变量：（属性）
方法：（行为）

5.变量

分：

成员变量：在类体内声明的变量，称之为成员变量。

成员变量分：

i.常量

描述：用 final 关键字修饰的成员变量，称之为常量。

特点：
第一：在声明常量时，必须对其进行初始化；
第二：初始化后，常量的值不能被修改。

ii.实例变量（非静态变量）

描述：没有被关键字 static 修饰的成员变量，称为实例变量(非静态变量）。

特点：
实例变量，只能通过对象来访问。

iii.类变量（静态变量）

描述：用关键字 static 修饰的成员变量，称为类变量（静态变量）；

特点：
类变量不仅可以通过对象来访问；
还可以直接通过类名来访问。

类变量被此类的所有实例共享。

局部变量：在方法体内声明的变量，称之为局部变量

6.变量的作用域

成员变量:

作用域是在整个类体内。

局部变量:

在方法体内声明的变量，其作用域从创建此变量开始，直到此方法体结束。
其他情况下声明的变量，其　　作用域是从局部变量创建开始到 本层　第一个“}”结束

例如：

```
public void show(){

int dd = 2;            //局部变量
System.out.println(dd);
System.out.println(i);

if(d > 1){
    int f = 0;
    if(f > 1){
        int ff = 1;
    }
}
}
```

成员方法：

方法的定义：

[权限修饰符] [static] [final] 返回值类型　方法名([参数列表]){
　//方法体
}

注意：

权限修饰符：public protected　private　　默认（友好）

返回值类型：基本数据类型或者是引用数据类型。

如果方法没有返回值，那么关键字 void 表示。

参数类表：

JAVA 中方法的传参方式：

JAVA 中的传参方式只有一种：传值

分两类：

基本数据类型传值

特点：

形参的改变，不会影响到实参

引用数据类型的传值

特点：

1.形参的改变，不会影响到实参

2.如果形参所指向的对象的值被改变了，那么会影响实参所指向的对象。

方法的重载：

方法重载的条件：

1.方法名必须相同

2.参数列表不同

分三种情况：

i.参数类型不同

ii.参数个数不同

iii.参数顺序不同

方法的分类：

实例方法（非静态方法）：没有被 static 修饰的方法，称之为实例方法。

特点：

实例方法只能通过对象来访问。

类方法（静态方法）：被 static 修饰的方法，称之为类方法。

特点：

可以通过对象来访问，也可以直接通过类名来方法。

类方法类此类的所有实例共享。

构造方法特点：

1.方法名和类名完全一样

2.此方法没有返回值，并且 void 必须省略不写。

构造方法的定义:

```
[权限修饰符] 类名([参数列表]){
    //方法体
}
```

构造方法总结：

1.如果类中没有自定义构造方法，那么系统会生成一个默认的无参构造方法。

否则的话，系统不在创建一个默认无参构造方法。

2.构造方法的作用：

构造方法是用来创建一个对象，并对此对象进行初始化。

this:

this:只当前类的对象,此时，可以通过”this.变量“的方式来调用当前类的成员变量

this([参数列表]):只当前类的构造方法，此代码必须放在构造方法的第一行。

在面向对象编程中：

1.如果在一个类中，有多个方法中使用了同名的局部变量,那么就把此变量声明成成员变量。

2.在一个类中，如果在多个方法中，使用了几行相同的代码，此时，就可以把这几行代码抽象为一个方法。

3.在多个类中，如果使用的几个相同的成员变量或成员方法此时，就可以把这几个成员变量和成员方法抽象为一个类。

面象对象抽象　-date:090813--090814

1.JAVA 是纯面向对象的语言

2.什么是面向对象？

面向对象是一种思想或者是一种解决问题的模式。

也就是说，不管我们遇见什么问题，我们都是从面向对象的角度出发。

用面向对象的思想对问题进行分析，建模，直到解决问题。

3.怎样用“面向对象”的思想解决问题？

用“面向对象”的思想解决问题需要三步：

第一步：OOA：面向对象分析

第二步：OOD：面向对象设计

第三步：OOP：面向对象编码

```
/*
    对学生类进行分析，抽象：
    创建一个学生类
*/
```

注意：什么是“抽象”

取其精华，去其糟粕。

抽象出对我们有用的属性和行为。

从面向对象的思想出发：

第一步：**OOA**

标识：学生（对象）

属性：

学号
姓名
性别
专业
。。。

行为：

上课
吃饭
走路
。。。

第二步：**OOD**

标识：Student

属性：

stuNo:String
stuName:String
stuSex:int (0 表示女；1 表示男)
dept:String

行为：

goSchool():void
eat():void
run():void

第三步：**OOP**

```
class Student{
    String stuNo;
    String stuName;
    int stuSex;
    String dept;

    void goSchool(){
        //方法体
    }

    void eat(){
    }
```

```
    void run(){
    }
}
```

问题：写一个程序，模仿自动提款机存钱、取钱的过程

第一步：OOA

标识：自动提款机——开户

属性：

账号

帐户余额

行为：

存钱

取钱

第二步：OOD

标识：Account

属性：

accountId:String

sumMoney:double

行为：

saveMoney(int money):void

drawMoney(int money):int

第三步：OOP

```
class Account{
    String accountId;
    double sumMoney;

    /**
        余额增加
    */
    void saveMoney(int money){
        sumMoney += money;
```

```
        }

        int drawMoney(int money){

            sumMoney -= money;          //==> sumMoeny = sumMoney - money;
            return money;
        }
    }
```

类/对象的初始化顺序　-date:090817

单个类的初始化顺序

1.在加载字节码文件时，初始化此类的所有类变量

2.在 Main 方法中，实例化一个对象时，先初始化此类的所有实例变量，然后执行构造方法。

子类对象在实例化时的初始化顺序

i.在加载字节码文件时，
先初始化父类中的类变量；
再初始化子类中的类变量。

ii.在 Mian 方法中，实例一个子类对象时：

先初始化父类中的实例变量，
再执行父类中的构造方法。

然后初始化子类中的实例变量，
再执行子类中的构造方法。

基本数据类型的封装类　-date:090817

byte ---------- java.lang.Byte
short ---------- java.lang.Short
int ---------- java.lang.Integer
long ---------- java.lang.Long
float ---------- java.lang.Float
double ---------- java.lang.Double
char ---------- java.lang.Character
boolean ---------- java.lang.Boolean

封装类的作用：封装类中封装了对其基本数据类型进行操作的一些方法。

作业：

/*

将整数 123 转换成字符串”123“

*/

解题思路：

因为此问题中牵涉到两个类：

String

Integer

所以，第一种情况：可以去 String 这个类中，找相应的技术支持。又因为 JAVA 中的类只有变量和方法，所以，将目标锁定在方法上。

目标方法的特点：

String 方法名（int num）{

xxx

}

目标方法范围缩小：

1.找一个参数只有一个，int/Integer 数据类型

2.此方法必有有一个返回值，返回值的类型为 String

找到目标方法：String valueOf(int num)

//==========

第二种情况：可以去 Integer 这个类中，找相应的技术支持。又因为 JAVA 中的类只有变量和方法，所以，将目标锁定在方法上。

目标方法的特点：

String 方法名（int num）{

xxx

}

或者是：能将其自身转换成字符串进行输出的方法。

例如：

```
class Student{
    String name = "yexu";
    public String getName(){
        return name;
    }
}
```

目标方法范围缩小：

1.找一个参数只有一个，int/Integer 数据类型

2.此方法必

目标方法：

Stirng toString()

String toString(int num)

作业：

*/

/*

将字符串”123“转换成整数 123

*/

/*

解题思路：

此问题中所牵涉的对象：

String

int/Integer

第一种情况：

String 找相应的技术支持，可以使字符串转换成整数。因为 JAVA 中只有类，类中只有变量和方法。

所以，我们只能找相应的方法。

方法特点：

1.int 方法名（String str）{}

2.或者是能将其自身转换成 int/Integer 类型的方法

目标方法：

没有找到

第二种情况：

Integer 找相应的技术支持，可以时字符串转换成整数。因为 JAVA 中只有类，类中只有变量和方法。

所以，我们只能找相应的方法。

方法特点：

1.int 方法名（String str）{}

目标方法：

1.int parseInt(String num)

2.int valueOf(String num)

权限修饰符 -date:090817

1.JAVA中的权限修饰符：

public :公有权限

protected : 受保护的权限

默认友好权限

private : 私有权限

比较/区别：

1.在当前类体内，此类的对象可以访问此对象中的所有成员变量及成员方法。

2.在同一个包中，此类的对象只能访问此对象中非私有的成员变量及成员方法。

3.在不同的包中，此类的对象只能访问此对象只能够的公有权限的成员变量及成员方法。

4.在同一个包中，子类可以继承父类中非私有的成员变量及方法。

5.在不同的包中，子类只能继承公有的及protected修饰权限的成员变量及方法。

继承 extends -date:090818

1.什么是继承？

i.继承是从已有类派生出一个子类的一种机制。

ii.子类中的成员变量和方法，有一部分是自己声明，另一部分是从父类中继承的，我们就称这种方式为继承。

iii.如果一个类没有显式的继承任何类，那么此类默认继承Object类。所以，Object

类是所有类的父类。

例如：

```
class ClassName{
}

class ClassName extends Object{
}
```

2.继承的语法：

```
class subClass extends superClass{
}

****************************************
//注意：ＪＡＶＡ只支持单重继承。
```

3.继承的优点：

i.减少代码的冗余

ii.便于维护

iii.便于程序的扩展

4.extends

5.继承性：

i.在同一个包中，子类可以继承父类中非私有的成员变量及成员方法。

ii.在不同的包中，子类只能继承父类中public及protected修饰权限的成员变量及成员方法。

6.方法的重写：

只有在继承中才有方法的重写。

规则：

i.子类中的方法的名字、参数列表以及返回值和父类中的方法完全一样。

ii.方法的权限修饰符不能降低。

方法重写的优点：

多态的一种体现。

7.super：

i.super:指当前类的直接父类的对象。

可以通过"super.成员变量","super.成员方法"的方式来调用从父类中继承的成员变量和方法。

例如：

```
class SuperClass{
  public void show(){
  }

  void show(String name){
  }
}

class SubClass extends SuperClass{
    public void show(){
}

  public void show(String name){
        }
    }
```

ii.super();指当前类的直接父类的构造方法。必须在构造方法的第一行。

例如：

```
    class SuperClass{

        public SuperClass(String name){
        }

        public void show(){
        }

        void show(String name){
        }
    }

    class SubClass extends SuperClass{

        public SubClass(String name){
            super(name);
        }
        public void show(){
        }

        public void show(String name){
        }
    }
```

8.子类对象在实例化时的初始化顺序：

i.在加载字节码文件时，
先初始化父类中的类变量；
再初始化子类中的类变量。

ii.在Mian方法中，实例一个子类对象时：

先初始化父类中的实例变量，
再执行父类中的构造方法。

然后初始化子类中的实例变量，
再执行子类中的构造方法。

9.上转型

上转型：指把子类的对象赋给父类的引用。

父类 父类引用 = new 子类();

i.这种方式是否可行？：可行

ii.此时父类的引用可以调用哪些成员变量和方法。

要考虑到：方法的重写，以及成员变量的隐藏。

例如：
对动物和猫进行分析建模

OOA：

标识：动物	猫
属性：	
体重	体重
毛色	毛色
年龄	年龄
性别... ...	性别
行为:	
吃	吃
跑	跑
叫... ...	叫

OOD：

Animal	Cat

属性：

weight	weight
color	color
age	age
sex.. ...	sex

行为：

eat()
run()
cry()

OOP:

```
class Animal{
    double weight;
    String color;
    int age;
    String sex;

    void eat(){
        System.out.println("Animal eat...");
    }

    void run(){
        System.out.println("Animal run...");
    }

    void cry(){
        System.out.println("Animal cry...");
    }
}

class Cat{
    double weight;
    String color;
    int age;
    String sex;

    void eat(){
        System.out.println("Cat eat...");
    }

    void run(){
```

```
            System.out.println("Cat run...");
        }

        void cry(){
            System.out.println("Cat cry...");
        }
    }

    class sonCat extends Animal{        //继承

    }

    class Test{
        Animal animal = new Animal();
    }
```

方法重载 vs 方法重写

1.方法重写：

应满足一下条件：

a.方法的名称、参数列表及返回值应和从父类继承的方法完全一样。

b.重写的方法权限不能降低。

c.如果从父类中继承的方法名后抛出了异常，那么重写的方法抛出的异常可以和父类中一样，或者是其子类；还可以不抛出任何异常。

例如：

```
    public class Animal {
        public void eat() throws Exception{
        }
    }

    public class Cat extends Animal {
        public void eat() throws RuntimeException{
        }
    }
```

2.方法重载：

a.方法名必须相同

b.参数列表不同

toString()

来自于 Object 类：
默认输出当前对象所在的类@对象在内存中的地址

例如：
com.umt.extend.Cat@d9f9c3

当输出一个对象时，此对象的引用会自动调用此方法。

例如：
```
Cat cat = new Cat();
System.out.println(cat);        //com.umt.extend.Cat@d9f9c3
```

向上转型　-date:090819

1.上转型：
将子类的对象赋给父类的应用，这种形式称之为上转型。
父类的引用所指向的对象，称之为上转型对象。

2.
i.上转型对象可以调用子类中从父类继承的成员方法及成员变量。如果子类重写了从父类继承的方法，那么上转型对象调用子类重写的方法。
ii.但是上转型对象丢失了一部分功能（子类自己定义的成员方法及成员变量）。
iii.上转型对象调用的方法中所操作的成员变量，是此方法所在类的成员变量。
iv.上转型对象.成员变量，此时的成员变量一定的父类的成员变量。
v.可以将上转型对象强制转换成子类对象，此时，子类对象拥有其所有的成员变量及成员方法。

注意：不能把父类的对象强制赋值给子类的引用。
区别：上转型对象　&　父类对象：不是一个概念

Final　-date:090820

1.final
final 修饰类：此类是最终类，不能被继承。

final 修饰方法，此方法是最终方法，不能被重写。
注意：final 修饰的方法是可以别重载的。

final 修饰变量，此变量是最终量（常量），定义时必须初始化，之后不能被修改。

2.final & static & abstract 区别

3.finally & finalize & final 区别

final:

final 修饰类：此类是最终类，不能被继承。
final 修饰方法，此方法是最终方法，不能被重写。
注意：final 修饰的方法是可以别重载的。
final 修饰变量，此变量是最终量（常量），定义时必须初始化，之后不能被修改。

finally:在 try{}catch(){}finally{}中出现，
不管 try 语句块中是否出现异常，finally 语句块中的程序一定被执行。

finalize():用于回收当前对象。

抽象类 abstract -date:090820

1.什么抽象类：
用关键字 abstract 修饰的类，就是抽象类。

2.定义抽象类有什么用？

抽象类中可以用抽象方法：
抽象方法：用 abstract 的方法就叫抽象方法。

抽象方法的特点：
只有方法的声明，而没有方法体。

```
/*
    写一个方法，此方法用来输出“HelloWorld”
    必须继承抽象类 TestAbstract
*/
```

3.抽象类的特点：
1.抽象类是用关键字 abstract 修饰的类
2.抽象类中可以用抽象方法，也可以没有抽象方法。
3.抽象类中可以有非抽象的方法。
4.抽象类不能实例化自己的对象，所以，抽象类必须被继承。
5.如果一个类中有抽象方法，那么此类一定是抽象类。

接口 interface -date:090820

1.接口的声明：

[权限修饰符] interface 接口名{

}

2.接口的特点：

1.接口是一个特殊的抽象类，但并不是类。

2.接口中的方法默认都是 public、abstract 类型的方法。

3.可以修饰接口的权限修饰符可以是 public 或者默认友好权限。

4.接口可以有自己的对象吗？不可以实例化自己的对象。

接口 接口引用 = new 接口实现类(); //称之为：接口回调（将接口实现类的对象赋值给接口的引用。是多态的一种体现。）

5.一个类可以实现多个接口

例如：

class 类名 implements 接口 1,接口 2,接口 3...{

}

此时，实现类必须实现接口中全部的抽象方法。

6.接口与接口之间，可以多重继承。

7.接口中的变量全部都是 final、static 类型常量。

三、异常 exception -date:090824

按程序出错的时间区分：

a.编译时出错。——一般是语法错误。

b.在运行时出错。——异常

1.什么叫异常？

运行时错误。

2.异常的分类：

Error：在运行时，发生的错误，但此错误不可处理。

Exception：在运行时，发生错误。但此错误可以处理。

int i = 2 / 0;

注意：所有异常类的父类：java.lang.Throwable

3.处理异常的方式：

throws　:被动处理异常。在方法名后向上抛出异常。

try{}catch（）{}finally{}：

主动处理异常。此时，程序可以继承执行。

try：块中存放可能出现异常的 JAVA 代码。

catch 块中的程序，只有在 try 块中的程序出现异常时才会被执行；否则不执行。

finally 块中的程序，不管 try 块中的程序是否出现异常，一定会被执行。

4.在 Exception 中，其子类 RuntimeException 及 RuntimeException 的子类异常，不需要我们进行处理。

其他异常，系统强制我们必须进行处理。

5.自定义异常步骤：

第一步：创建一个类，让此类继承已有的一个异常类。

第二步：重写方法：

a.创建此类构造方法，让此构造方法调用父类的构造方法。

例如：public class ByZeroException extends Exception {

```
        public ByZeroException(String msg){
            super(msg);
        }
    }
```

其他方法：

```
    public String getMessage(){
    }

    public void printStackTrace(){
        toString();
        //... ...
    }

    public String toString(){
        System.out.println(this.getClass().getName() + ":" + getMessage());
        //
    }
```

6.throw　&　throws

throw 在方法体中调用，用来抛出一个异常；

throws 在方法名后面调用，用来向上抛出异常。

四、容器/数据结构　-date:090825

Conllection

1.Collection & List & ArrayList & LinkedList & Vector 区别

1.Collection 是List的超接口；ArrayList LinkedList Vector 是List的实现类。

2.ArrayList、LinkedList以及Vector在使用形式上完全一样。

不同点：

1.ArrayList采用顺序存储结构来保存数据。这种结构方便我们查询数据。
但不利于对数据进行更新操作（例如：插入、删除）

2.LinkedList采用链表存储结构，方面对数据进行更新操作，
但不利于对数据遍历操作。

3.Vector和ArrayList基本上一样。不过Vector是线程安全的。ArrayList不是。

Set

2.Set & List & HashSet & TreeSet区别

i.Set将像数学中的集合。第一：Set中的对象是无序的。第二：Set中的对象是不能重复的。
List中的对象是有序的，并且可以重复。

ii.HashSet & TreeSet都实现了Set接口。
但HashSet中的对象的无序的。而TreeSet中的对象按升序进行排列。
并且HashSet中可以存储任意类型的对象；TreeSet只能存储可以和当期对象进行比较的对象。

3.Collections

类：封装了对容器进行操作的一些方法。并且这些方法是静态的，可以直接通过类型访问。

Map -date:090826

1.Map & HashMap & TreeMap & HashTable 区别

Map 是接口；HashMap 和 TreeMap 是 Map 的实现类；
Map 中存放的对象是以键值对的形式进行存储。
HashMap 和 TreeMap 都可以存储 Null 键和 Null 值；HashTable 不允许；
HashMap 和 HashTable 中存储的对象是无序的；TreeMap 是按键进行升序排列。

五、IO 流 -date:090827

1.流的概念：

在 JAVA 中，对数据进行输入和输出的管道，就是流。

2.流的分类：

第一种：按流的方向（以程序为参照物）

输入流:InputStream FileInputStream Reader FileReader
输出流：OutputStream FileOutputStream Writer FileWriter

第二种：按输入/输出数据块的大小

字节流：
InputStream FileInputStream OutputStream FileOutputStream
字符流：Reader FileReader Writer FileWriter

第三种：按流的功能

节点流:
InputStream FileInputStream OutputStream FileOutputStream
Reader FileReader Writer FileWriter

处理流:
BufferedInputStream BufferedOutputStream BufferedReader BufferedWriter

3.流的使用：

分三步：

第一步：打开流（创建流）
第二步：操作流
第三步：关闭流

六、多线程　-date:090901

创建线程的方式：两种

第一种：实现Runnable接口

第i步：创建一个类，实现Runnable接口
例如：

```
class TestRunnable implements Runnable {
    public void run() {
    }
}
```

第ii步：实现run（）方法

例如：

```
public void run() {
        for(int i = 1; i <= 2000; i++){
            System.out.println(i);
        }
}
```

启动线程类：
第i步：创建此类的对象
例如：

```
TestRunnable tr = new TestRunnable();
```

第ii步：通过Thread类来启动

```
Thread t = new Thread(tr);
t.start();
```

第二种方式：继承Thread类
第i步：创建一个线程类，此类继承Thread
例如：

```
public class TestThread extends Thread{}
```

第ii步：重写run（）方法
例如：

```
public void run() {
    for(int i = 0; i < 100; i++){
        System.out.println(i);
    }
}
```

```
第iii步：启动线程
    TestThread tt = new TestThread();
    tt.start();
```

Oracle

一、Oracle 单表基本操作 -date:090909

在 oracle 文件中，以--开头表示单行注释。

--创建表

--创建一张表，练习 create 语句的语法
--create table <表名> (<列名> <数据类型> [约束条件]，<列名> <数据类型> [约束条件]。。。)

--问题：创建一张表 student(sno,sname,sex,birth,score)

```
create table STUDENT(
        sno number primary key,
        sname nvarchar2(10) not null,
        sex nvarchar2(1),
        birth date,
        score number(3,1)
)
```

--修改表(结构)

```
--alter table <表名>
--[add  列名  数据类型   约束条件]
--[modify  列名  数据类型]
--[drop  列约束]
alter table student
add other nvarchar2(20) not null;


alter table student
modify other number;
```

--删除表

--drop table 表名
drop table student

--单表查询语句

--查询 student 表中的全部记录
Select * from student;

--查询 student 表的 **sno**，**sname** 的全部记录**(**映射**)**
select sno,sname from student;

--查询 student 表中全部记录，按学分进行升序排列
select *
from student
order by score asc

select *
from student
order by score

--查询 student 表中全部记录，按学分进行升序排列
--如果学分相同，再按学号进行降序排列
select *
from student
order by score,sno desc

--查询 student 表中全部记录，按学分进行降序排列
select *
from student
order by score desc

--查询出 student 表中的全部性别,相同值只显示一个

```
select distinct sex
from student
```

--查询 student 表中女学生的记录

```
select *
from student
where sex = 12
```

--查询 student 表中“cccc”学生的详细记录

```
select *
from student
where sname = 'e'
```

--查询 student 表中学号在 2 和 5 之间的学生记录

--between startNum and endNum :包含开始和结束的两个数

```
select *
from student
where sno between 2 and 5
```

--in

```
select *
from student
where sno in (2,4,5,9);
```

--> <

```
select *
from student
where sno >= 2 and sno <=5 and sex = 1
```

--在 student 表中查询所有男学生的记录，并按学分进行降序排列

```
select *
from student
where sex = '1'
order by score desc
```

--查询 student 表中姓名以 e 开头的学生的记录

```
select *
from student
where sname like 'e%'
```

--查询 student 表中姓名含有 e 的学生的记录

```
select *
from student
where sname like '%e%'
```

-- 查询出 student 表中姓名的第三个字母是 e 的学生记录

```
select *
from student
where sname like '__e%'
```

--插入记录

--insert into <表名>[(列名，列名。。。)] values(列值，列值。。。)

```
insert into student
values(8,'xxxxx','01',null,10)
insert into student(sno,sname,sex)
values(6,'e','1');
```

二、表函数、数据类型转换、哑表、语法顺序 -date:090910

--按性别进行分组，统计男同学的人数及女同学的人数

```
select sex,count(*)
from student
group by sex;
```

--统计男同学及女同学的人数，及 sno

--group by
--select 语句后面查询的字段，如果没有用组函数，那么此字段必须出现在 gruop by 后面。
--select 语句后面没有查询的字段，也可以出现在 group by 后面。

```
select sex,count(*)
from student
group by sex,sno

--count()
select count(*)  总人数
from student

select count(*) as  总人数
from student

select sname "姓  名",sno as  编号
from student

select s.sno,s.sname
from student s

select s.sno,s.sname
from student    s
```

--别名

--给字段定义别名

--有两种方式：一种是用 as，另一种是直接在字段名后写别名

--给表定义别名

--只有一种，就是在表名后添加别名，不能用‘as’

--select 语句的语法顺序

--单表查询的 select 语句
select 字段名/组函数 [,字段名/组函数...]
from   表名
where 查询条件
group by 字段/组函数
having 组函数判断条件
order by 字段

总结一下顺序：s f w g h o

--统计男同学及女同学的人数，按人数进行升序排列

--注意：order by 语句必须出现在 select 语句的最后一行
--having:对应用了组函数的字段进行判断时，必须用 having，不能用 where

```
select sex,count(*)
from student
where sex = '1'
group by sex
having count(*) >= 4
order by count(*)
```

--max
--min
--avg

--查询 student 表中的最高成绩，最低成绩及平均成绩

```
select max(score) 最高成绩,min(score) 最低成绩,avg(score) 平均成绩
from student;
```

```
--select
--insert into
--update
--delete
```

--update 更新数据库中的一条记录

--update 表名
--set 字段 1 = 值 1[,字段 2 = 值 2，字段 3 = 值 3]
--where 条件

```
update student
set sname = 'yexu'
where sno = 6

update student
set sname='leining', sex = '1' , score = 32
where sno = 4
```

--delete 删除表中指定的记录

--delete [from] 表名
--[where 条件]

--删除 student 表中 sno 为 9 的学员的记录

```
delete student
where sno = 9
```

--oracle 特点

--top:注意在 Oracle 数据库中没有 top 这个关键字
--rownum
--rownum 的特点：
    -rownum 的值永远从 1 开始
    -rownum 的值是动态产生的，它会根据查询结果来动态产生 where sex = 0。

--查询 student 表中的前三条记录

```
select *
from student
where rownum <= 3
```

--查询 student 表中的第 3 条到第 6 条记录

```
select * from (
        select s.*,rownum as myrow
        from student s
)
where   myrow <= 6 and myrow >=3
```

--oracle 数据中的常见数据类型
--number
--char
--date

--数据类型转换

--to_date　to_char　to_number

--dual:哑表

select 1 + 1 from dual;

--1+200 * 34

select 1 + 200 * 34 from dual;
select to_char(123) from dual;
select to_number('123') from dual;
select to_char(to_date('1999-1-1','yyyy-mm-dd'),'yyyy-mm-dd') from dual;

--sysdate

select sysdate from dual;

三、多表操作、左(右全)查询、视图、序列、数据库备份

-date:090911

--查询出员工的详细信息，及员工的姓名，编号，所在部门

select empname,empid,deptname
from emp e,dept d
where e.deptId = d.deptId;

select empname,empid,deptname
from emp e,dept d
where d.deptId = empId;

select * from emp
--emp(empid,empname,deptid)
--dept(deptid,deptname);

select　e.*,d.*
from emp e,dept d
where e.deptId = d.deptId;

```
select e.empid,e.empname,e.deptid,d.deptname
from emp e,dept d
where e.deptid = d.deptid;
```

--外连接

--左外连接　　右外连接　　全连接

```
select    e.*,d.*
from emp e left join    dept d
on e.deptId = d.deptId;

select    e.*,d.*
from emp e right join    dept d
on e.deptId = d.deptId;

select    e.*,d.*
from emp e full join    dept d
on e.deptId = d.deptId;
```

--查询员工的详细信息，包括用户名和密码，及所属部门

```
--emp      users    dept
select e.*,d.*,u.*
from emp e
left join dept d on e.deptid = d.deptid
left join users u on e.empid = u.empid
--[left join ...]

select e.*,d.*,u.*
from emp e,dept d,users u
where e.deptid = d.deptid and e.empid = u.empid

--insert delete upate select
```

--视图

```
create view emp_view
as
select e.empid,e.empname,d.deptname,u.name,u.pwd
from emp e
left join dept d on e.deptid = d.deptid
left join users u on e.empid = u.empid
where e.empid <> 7
```

--查询员工的详细信息，包括用户名和密码，及所属部门

```
select * from emp_view
where empid != 6
```

--序列

--第一次使用序列时，必须注册此序列

```
select seq_users.nextval from dual;

select seq_users.currval from dual;

insert into users(id,name,pwd,empid)
values(seq_users.nextval,'aaaa','123',5);

select * from users;
```

四、数据库的备份迁移

1.备份数据库

//==========DOS 下操作==========导出

```
C:\Documents and Settings\Administrator>exp
Export: Release 10.2.0.1.0 - Production on 星期五 9 月 11 11:18:47 2009
Copyright (c) 1982, 2005, Oracle.    All rights reserved.


用户名: test
口令:

连接到: Oracle Database 10g Enterprise Edition Release 10.2.0.1.0 - Production
With the Partitioning, OLAP and Data Mining options
输入数组提取缓冲区大小: 4096 >
  导出文件: EXPDAT.DMP > c:\mytable.dmp
(1)E(完整的数据库), (2)U(用户) 或 (3)T(表): (2)U >
导出权限 (yes/no): yes >
导出表数据 (yes/no): yes >
压缩区 (yes/no): yes >
已导出 ZHS16GBK 字符集和 AL16UTF16 NCHAR 字符集
即将导出指定的用户...
要导出的用户: (按 RETURN 退出) > test
要导出的用户: (按 RETURN 退出) >
. 正在导出 pre-schema 过程对象和操作
. 正在导出用户 TEST 的外部函数库名
. 导出 PUBLIC 类型同义词
. 正在导出专用类型同义词
. 正在导出用户 TEST 的对象类型定义
即将导出 TEST 的对象...
. 正在导出数据库链接
. 正在导出序号
. 正在导出簇定义
. 即将导出 TEST 的表通过常规路径...
. . 正在导出表                          DEPT 导出了          6 行
. . 正在导出表                           EMP 导出了          7 行
. . 正在导出表                       STUDENT 导出了          9 行
. . 正在导出表                         USERS 导出了          5 行
. 正在导出同义词
. 正在导出视图
. 正在导出存储过程
```

. 正在导出运算符
. 正在导出引用完整性约束条件
. 正在导出触发器
. 正在导出索引类型
. 正在导出位图, 功能性索引和可扩展索引
. 正在导出后期表活动
. 正在导出实体化视图
. 正在导出快照日志
. 正在导出作业队列
. 正在导出刷新组和子组
. 正在导出维
. 正在导出 post-schema 过程对象和操作
. 正在导出统计信息
成功终止导出, 没有出现警告。

2.数据库的导入

//==========DOS 下操作==========导入数据库备份

C:\Documents and Settings\Administrator>imp

Import: Release 10.2.0.1.0 - Production on 星期五 9 月 11 11:24:37 2009

Copyright (c) 1982, 2005, Oracle. All rights reserved.

用户名: edu@orcl_192.168.1.252
口令:

连接到: Oracle Database 10g Enterprise Edition Release 10.2.0.1.0 - Production
With the Partitioning, OLAP and Data Mining options

导入文件: EXPDAT.DMP> c:\mytable.dmp

输入插入缓冲区大小 (最小为 8192) 30720>

经由常规路径由 EXPORT:V10.02.01 创建的导出文件

警告: 这些对象由 TEST 导出, 而不是当前用户

已经完成 ZHS16GBK 字符集和 AL16UTF16 NCHAR 字符集中的导入
只列出导入文件的内容 (yes/no): no >

火龙果•整理
uml.org.cn

由于对象已存在, 忽略创建错误 (yes/no): no >

导入权限 (yes/no): yes >

导入表数据 (yes/no): yes >

导入整个导出文件 (yes/no): no > yes

. 正在将 TEST 的对象导入到 EDU
. . 正在导入表 "DEPT"导入了 6 行
. . 正在导入表 "EMP"导入了 7 行
. . 正在导入表 "STUDENT"导入了 9 行
. . 正在导入表 "USERS"导入了 5 行
即将启用约束条件...
成功终止导入, 没有出现警告。

END