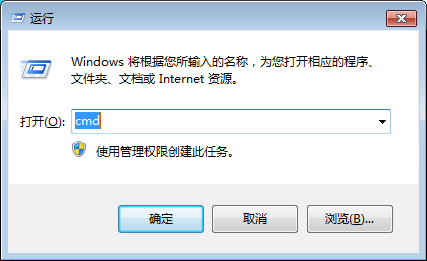
# 第一天 上午

## 学习内容：

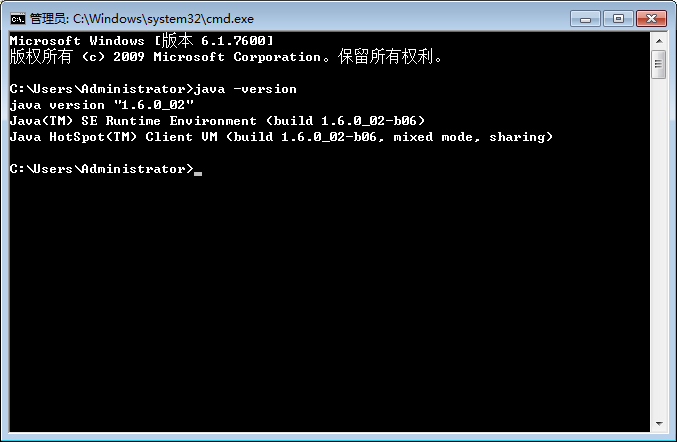
Java基础、sql（Mysql）。周考核Java基础的考核。

## 1.Java环境变量：

检测Java环境变量是否配置成功：

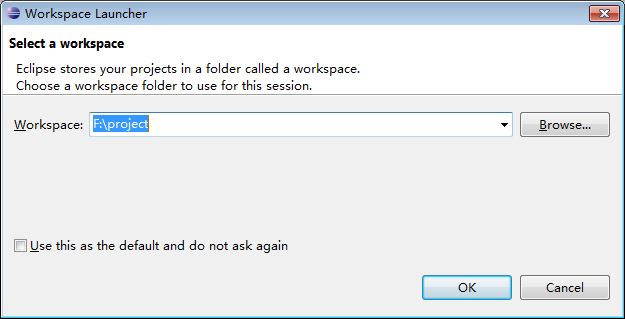


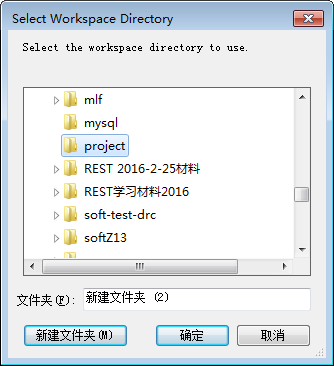
输入【Java】或者【Javac】或者【Java -version】



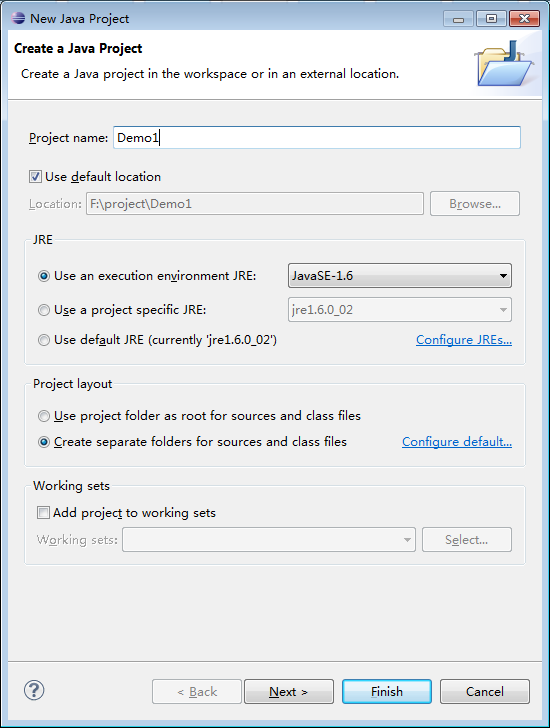
## 2.使用eclipse开发

1、设置工程存放目录【F:\project】，勾选下面默认按钮

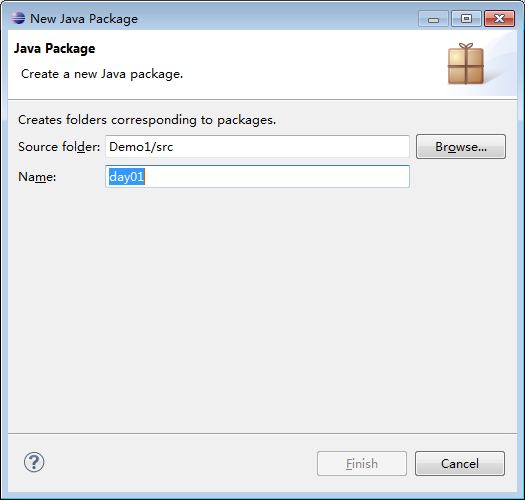




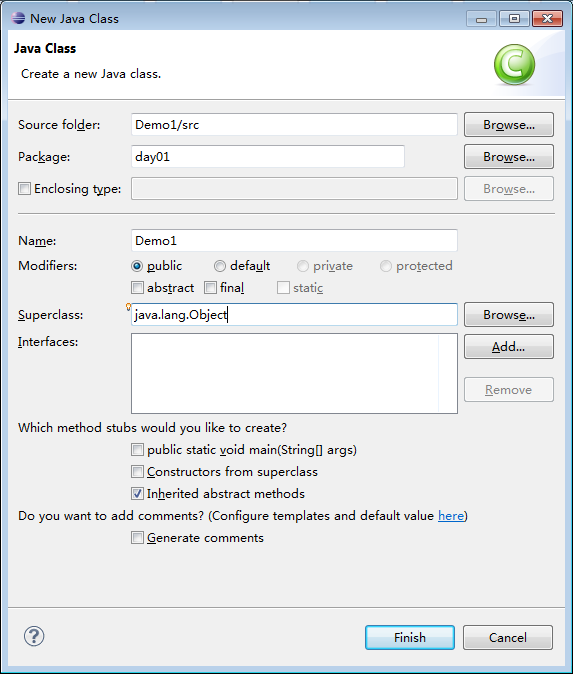
2、新建一个工程【file——new——Java project】然后点击finish



3、新建包【day01】



4、新建class【Demo1】



## 3.编写代码：

### 快捷键：

1. alt+/ ：代码提示。

Mian方法快捷提示：main——》alt+/：自动补全

输出语句的快捷方式：sysout ——alt+/：自动补全

2、ctrl+alt+↓：快速向下复制

简单的第一个程序：

package day01;

public class Demo1 {

public static void main(String[] args) {

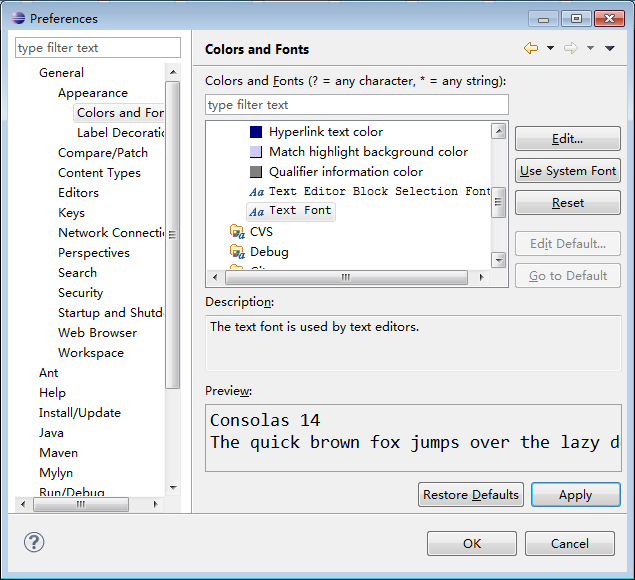
System.out.println("今天是学习java的第一天");

}

}

## 4.设置字体：

Window——preferences——General——Appearance——Colors and Fonts——basic【最下面】——edit



## 5.Java的运行机制：

两个关键的步奏：

1. javac——编译.Java文件—》.class字节码文件
2. java——调用java虚拟机来加载.class文件，并执行

## 6.注释

//单行注释

/\*

\* 多行注释

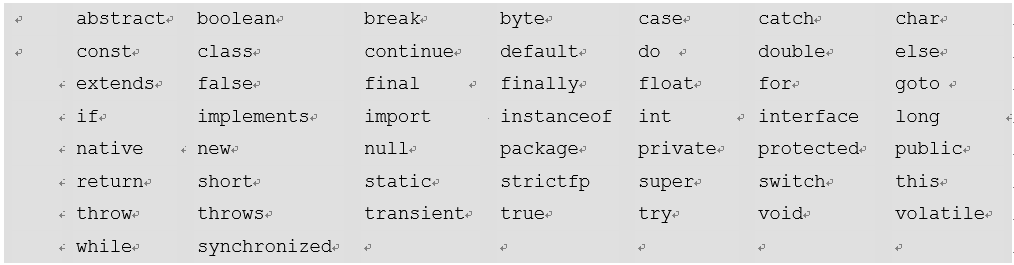
\*/

/\*\*

\* 文档注释

\*/

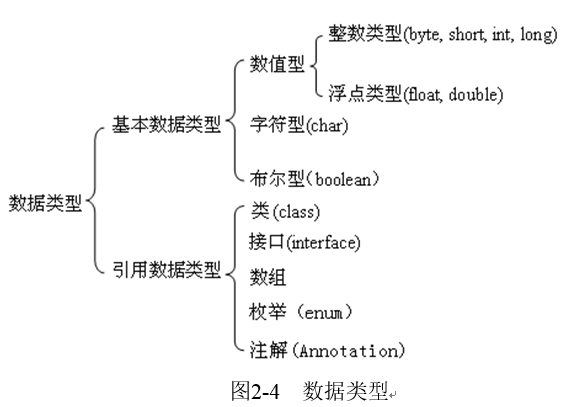
## 7.标识符和关键字：



常量：在代码里面写死的，不会被更改的

变量：可变的值：a = 1;

变量的类型：



## 8.运算符：

+、-、\*、/、++、--、%、&&、||、！、&、|、~、>>、<<

## 9.选择结构：

If（）{

}else{

}

Switch（a）{

Case 1：break；

Default：break；

}

## 10.循环结构：

While（）{

}

//至少执行一次

Do{

}while（）

//常用在已知循环次数的情况

For（int i=12;i<20;++i）{

//++i循环体结束后才执行

}

99乘法表

**for** (**int** i = 1; i < 10; i++) {

**for** (**int** j = 1; j <= i; j++) {

System.*out*.print(j+" \* "+i+" = "+i\*j+" ");

}

System.*out*.println();

}

1 \* 1 = 1

1 \* 2 = 2 2 \* 2 = 4

1 \* 3 = 3 2 \* 3 = 6 3 \* 3 = 9

1 \* 4 = 4 2 \* 4 = 8 3 \* 4 = 12 4 \* 4 = 16

1 \* 5 = 5 2 \* 5 = 10 3 \* 5 = 15 4 \* 5 = 20 5 \* 5 = 25

1 \* 6 = 6 2 \* 6 = 12 3 \* 6 = 18 4 \* 6 = 24 5 \* 6 = 30 6 \* 6 = 36

1 \* 7 = 7 2 \* 7 = 14 3 \* 7 = 21 4 \* 7 = 28 5 \* 7 = 35 6 \* 7 = 42 7 \* 7 = 49

1 \* 8 = 8 2 \* 8 = 16 3 \* 8 = 24 4 \* 8 = 32 5 \* 8 = 40 6 \* 8 = 48 7 \* 8 = 56 8 \* 8 = 64

1 \* 9 = 9 2 \* 9 = 18 3 \* 9 = 27 4 \* 9 = 36 5 \* 9 = 45 6 \* 9 = 54 7 \* 9 = 63 8 \* 9 = 72 9 \* 9

# 第一天 下午

## 1.一维数组

### 步奏

1. 申明（定义）数组变量：int[] arr;
2. 初始化（动态创建）：arr = new int[5]
3. 创建数据（如果没有初始数据，默认值就该基本数据类型的默认值，如果是引用类型，默认值是null）



### 数组的初始化方式：

1. Int arr = new int[5];
2. Int arr = {23,343,454,55};

//数组的循环遍历

**int**[] arr;

arr = **new** **int**[10];

**for** (**int** i = 0; i < arr.length; i++) {

//初始化

arr[i] = i+10;

//输出

System.*out*.println(arr[i]);

}

## 2.多维数组：

//【申明】5个一维数组,并没有初始化第二维

**int**[][] arrr = **new** **int**[5][];

**for** (**int** i = 0; i < arrr.length; i++) {

//第二重循环必须先创建

arrr[i] = **new** **int**[6];

**for** (**int** j = 0; j < arrr[i].length; j++) {

arrr[i][j] = 456-i;

System.*out*.println(arrr[i][j]);

}

}

## 3.类和对象：

//类（分类【虚拟的】）

//对象（类中一个个体【实际存在的】）

//Java里面使用的都是对象

类里面有属性和方法：

类的三个特性：

【封装】【多态】【继承】

**public** **class** Student {

//静态变量:值是属于类,所以的对象公用

**public** **static** **int** *TAG* = 123456;

//属性

String name;

String age;

String number;

//默认构造方法，new 关键字创建对象的时候执行

//没有返回值，方法名和类名相同

**public** Student(){

}

//多态的一种体现

**public** Student(String name){

//this:表示当前对象

**this**.name = name;

}

//方法的多态通过参数的【数目、类型、顺序】什么来控制

//不能通过返回值

**public** Student(String name,String age){

}

**public** Student(**int** name,String age){

}

**public** Student(String age,**int** name){

}

//方法

//谁都能用

**public** **void** xuexi1(){

}

//私有的：我独有的

**private** **void** xuexi2(){

}

//保护的：一家人的（同一包下的）

**protected** **void** xuexi3(){

}

//静态的方法：方便使用》使用原则：不含操作内存的代码块

**public** **static** **void** test(){

}

//在程序启动的时候就加载到内存里面

//知道程序退出后才被清除

//所以不能广泛的使用

**static**{

**int** a;

System.*out*.println();

}

}

单例模式：

//单例模式

//只能有一个实例对象

**private** **static** Student *instance*;

**public** **static** Student getInstance() {

**if**(*instance* == **null**){

//有且只有一次被执行

*instance* = **new** Student();

}

**return** *instance*;

}

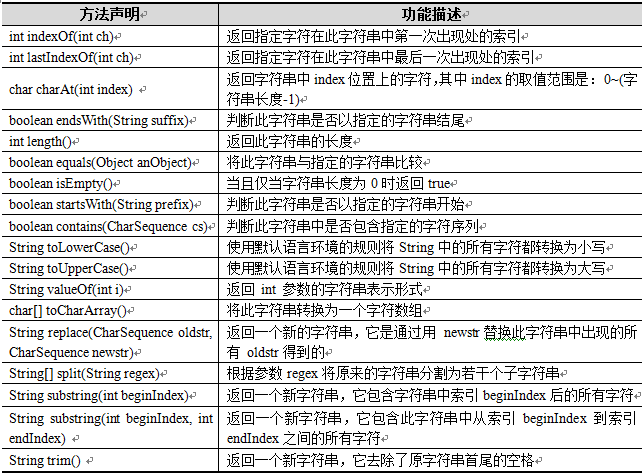
使用：

Student student = Student.*getInstance*();

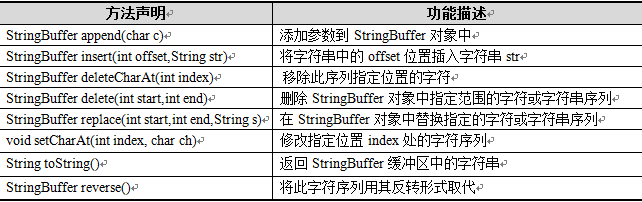
Student student2 = Student.*getInstance*();

Student和student2都是指向同一个student对象。

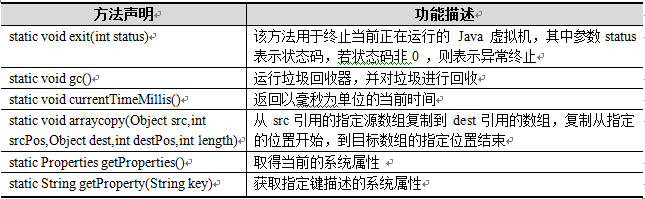
## 4.String字符串：



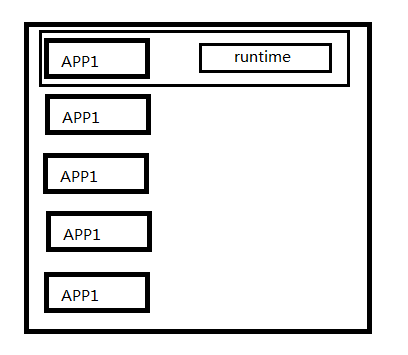
## 5.StringBuffer：

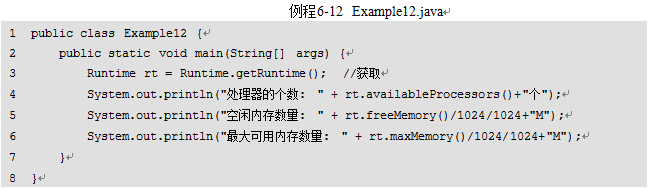


## 6.System：



## 7.Runtime类：



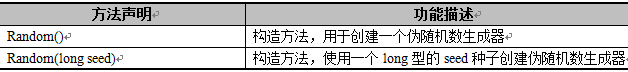


## 8.随机数：

//范围（0,1）0.0006

**int** a = (**int**)(Math.*random*()\*100+1);

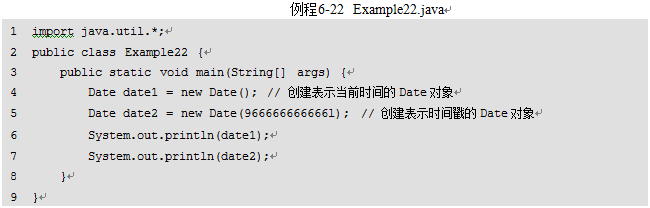
System.*out*.println(a);



## 9.包装类：



## 10.Date：



## 11.DateFormat类：

## 第一天作业：String、StringBuffer、System、Runtime、Date里面的方法都像是用一遍

# 第二天 上午

## list集合



### 单列集合

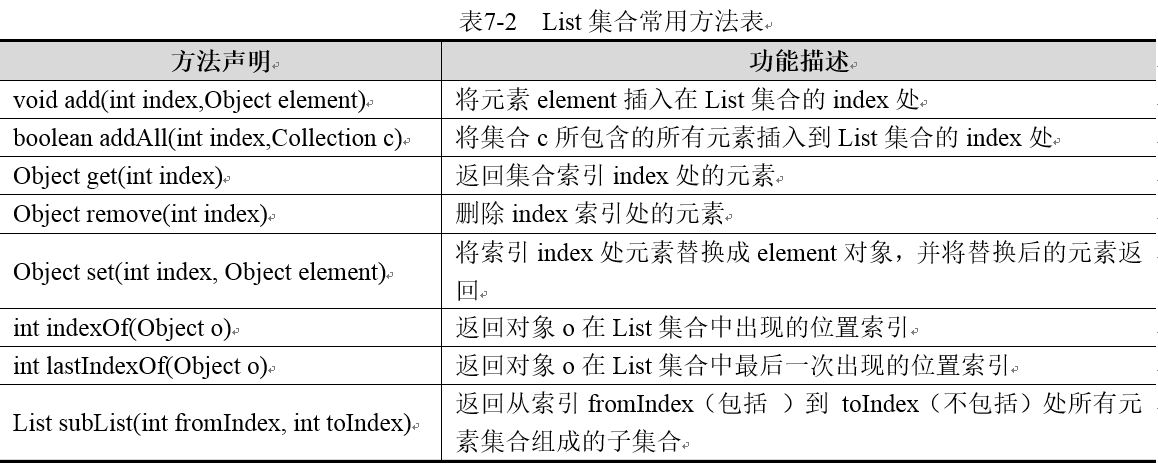
顶层接口【collection】

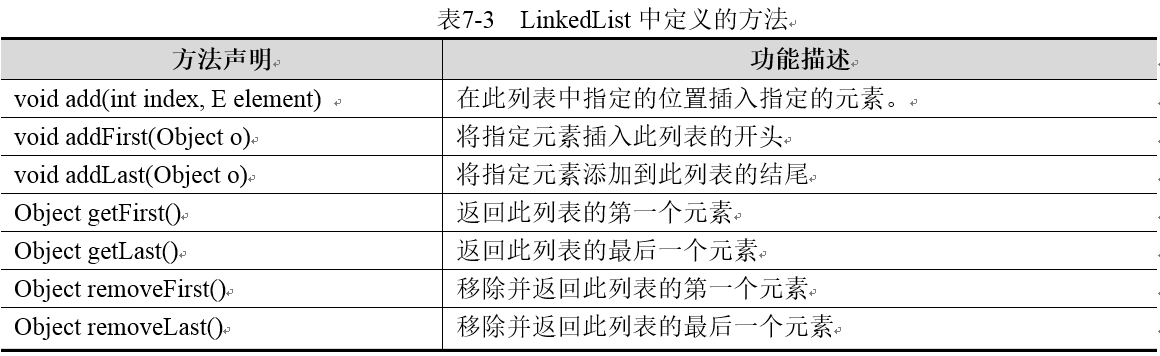
直接子类（接口）：【list：可重复、有序】【set：不可重复、无序（不按照添加顺序）】

List的子类：【ArrayList】【LinkedList】

Set的子类：【HashSet】【TreeSet】







### 作业：ArrayList练习，存你们班学生【Student】

Public Student{

Private String number；

Private String name；

}

### 双列集合类

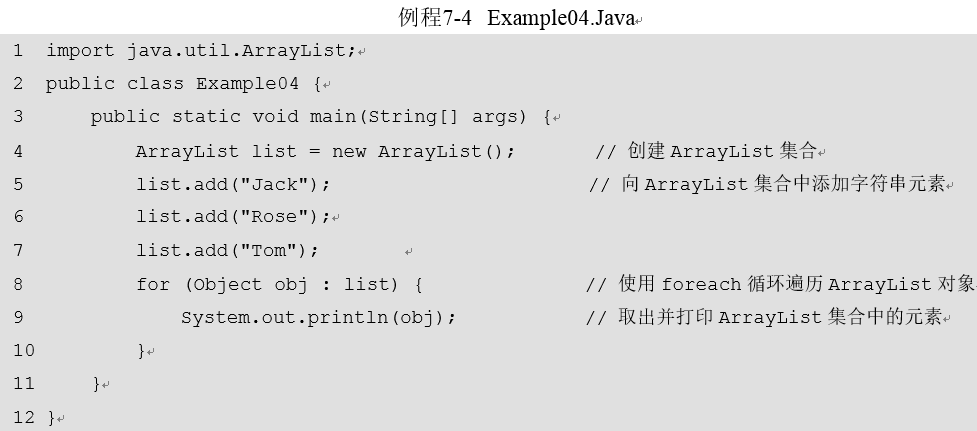
顶层接口【map】

用于存储具有键(Key)、值(Value)。

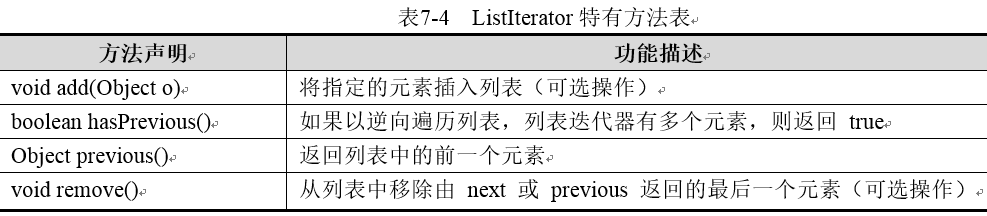
直接子类：【HashMap】【TreeMap】

### Iterate集合的【正向】迭代（循环）操作





### ListIterate【可逆向迭代】



# 第二天 下午

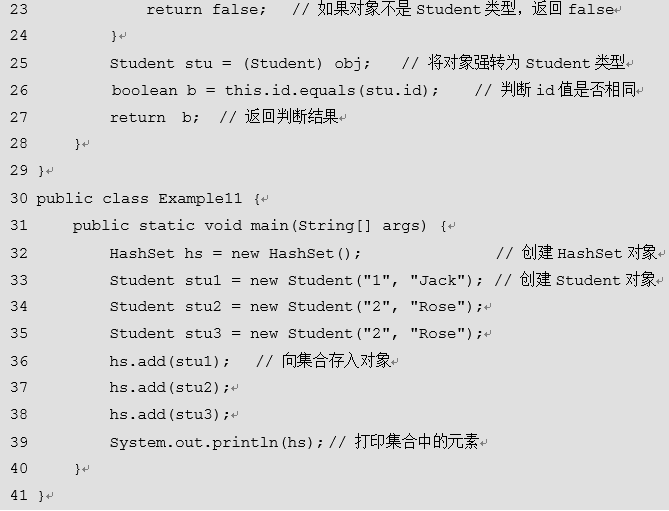
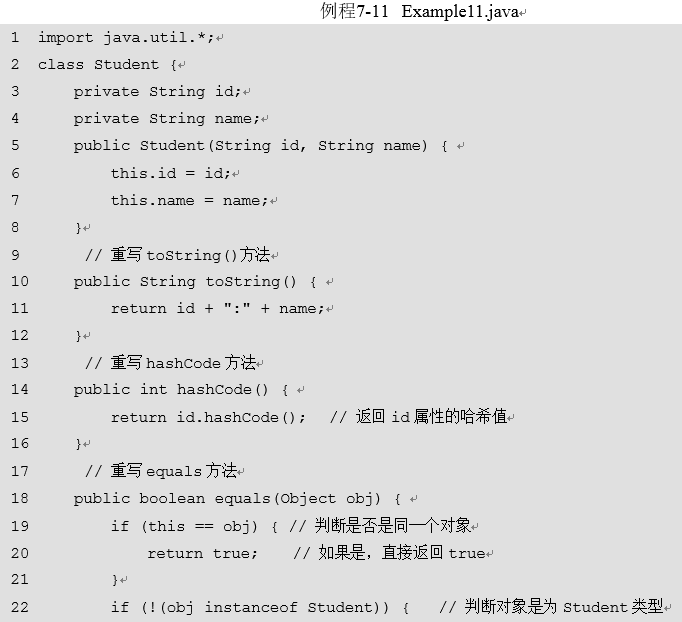
## Set集合

Set集合是无序不重复的集合

### 作业1：HashSet练习

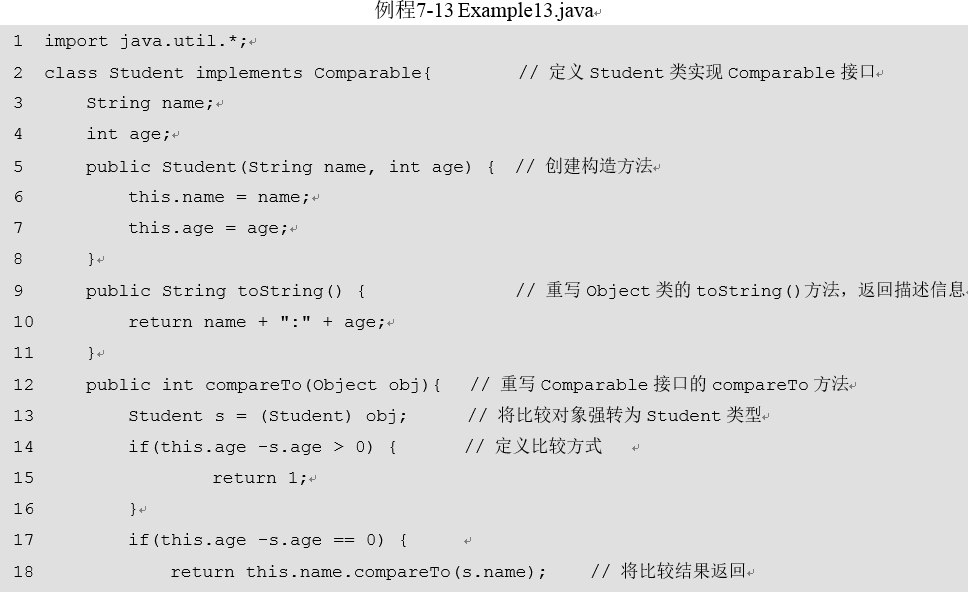


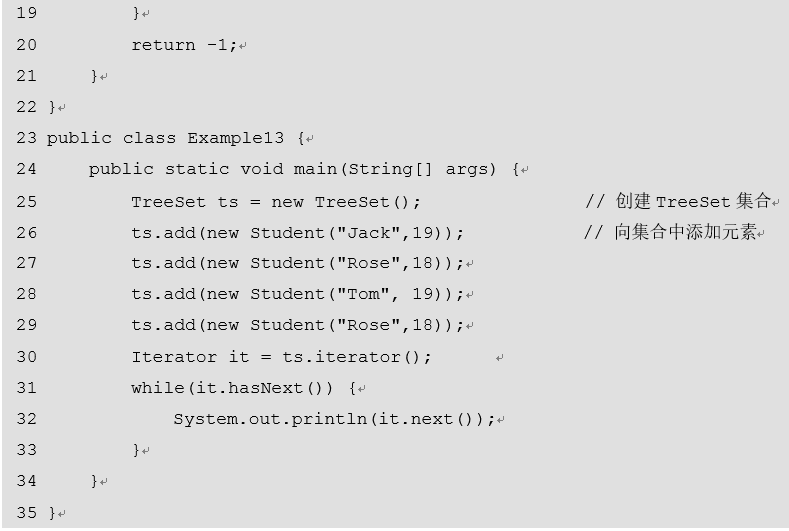
案例：

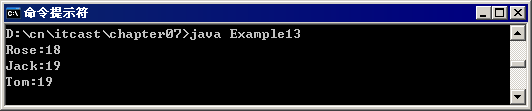


### 作业2：TreeSet练习

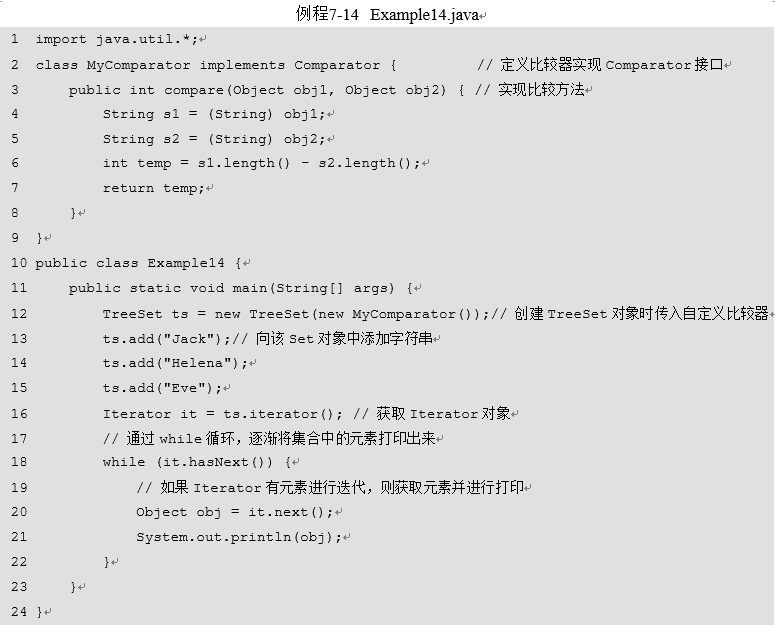
第一种实现方式

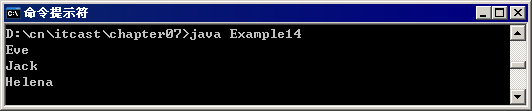






第二种实现方式：



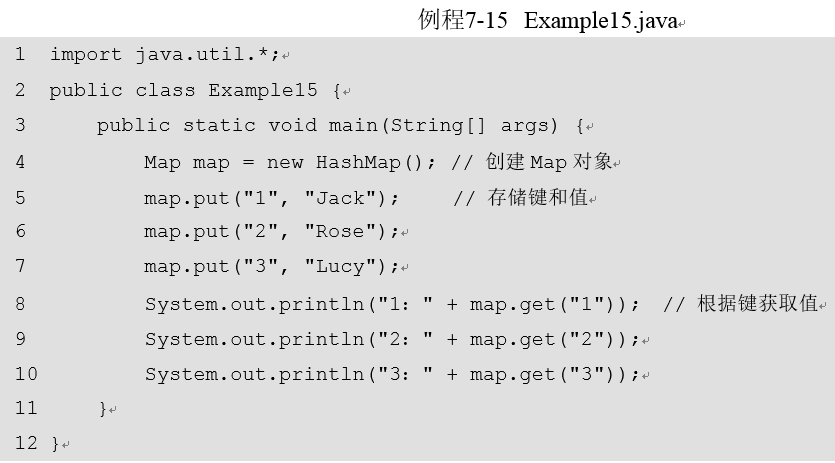


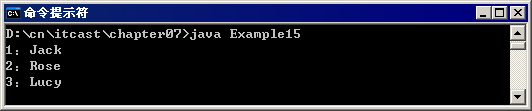
## Map（键值对，双列表）：

保证不出现重复的键



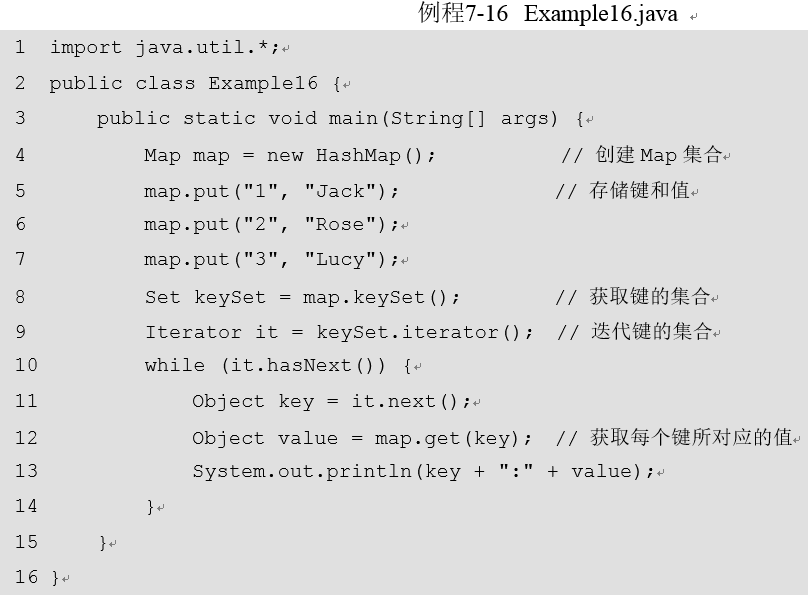
案例：

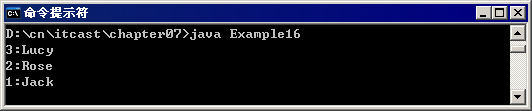




### Map的遍历

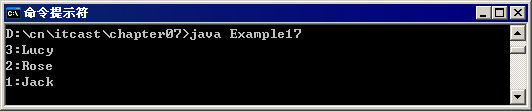
第一种方法：





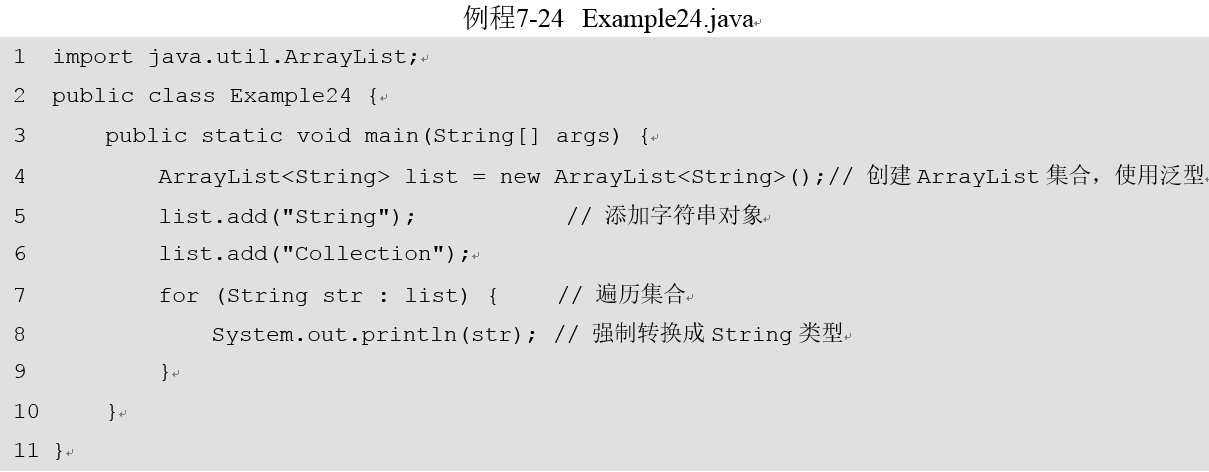
第二种方法（取得一对Key|value）

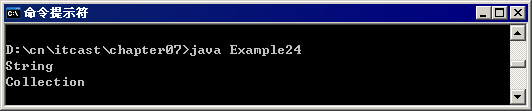




## 泛型

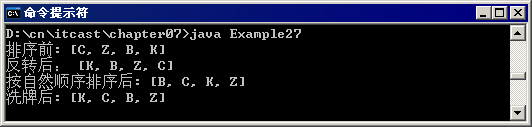
规范集合元素的类型，一旦制定就只能该类型的数据。

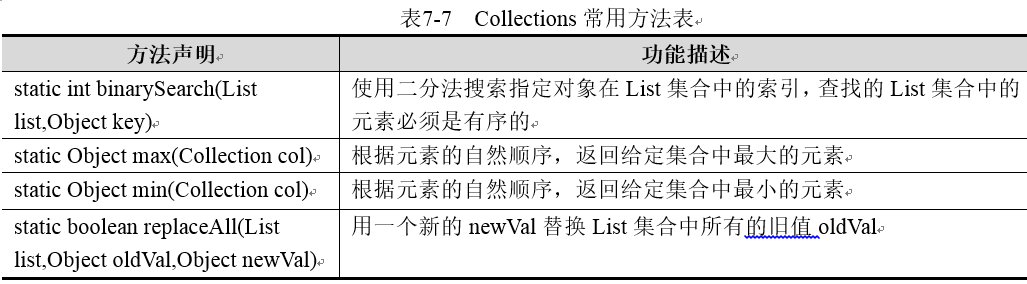


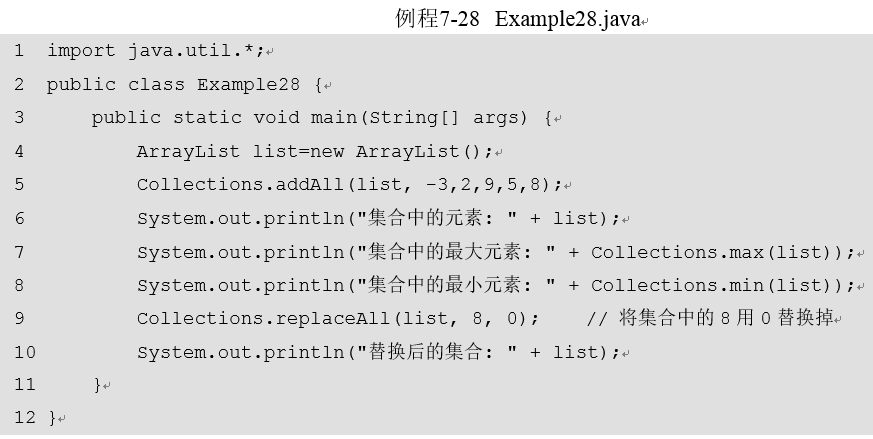


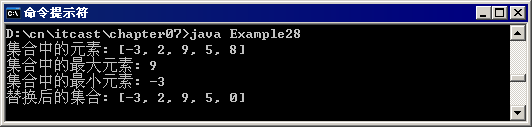
## 集合的操作







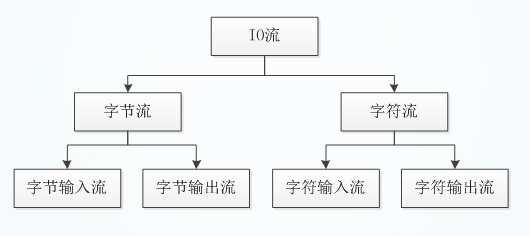




## 5.抽象类和接口的区别和想同点：

* **相同点：**
  + 两者都包含抽象方法。
  + 两者都不能被实例化。
  + 两者都是引用数据类型。
* **不同点：**
  + 抽象类约定多个子类之间共同使用的方法；接口约定多个互不相关类之间共同使用的方法。
  + 抽象类与子类之间采用单重继承机制；一个类实现多个接口则实现了多重继承的功能。
  + 抽象类及其类中成员具有与普通类一样的访问权限；接口的访问权限有public和默认两种，接口中成员的访问权限均是public。
  + 抽象类中可以包含非抽象方法，也可以声明构造方法；接口中的方法全部是抽象方法，不能声明构造方法。
  + 抽象类中可以声明成员变量，子类可以对该成员变量赋值；接口中只能声明常量。

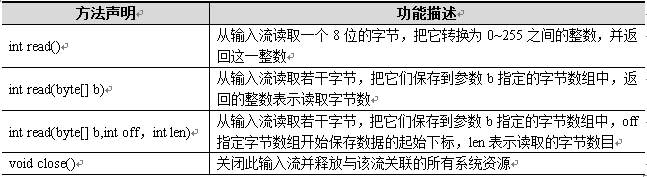
## IO

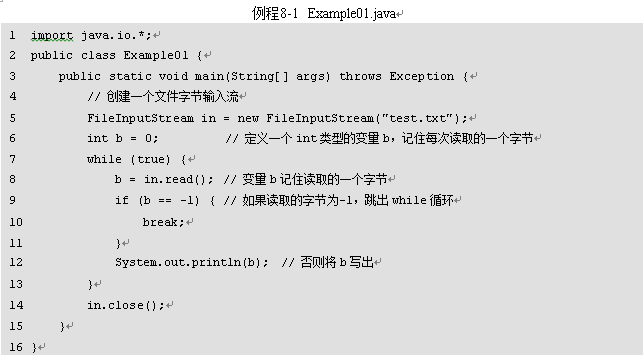


### InputStream

InputStream是所有字节输入流顶级父类

InputStream：

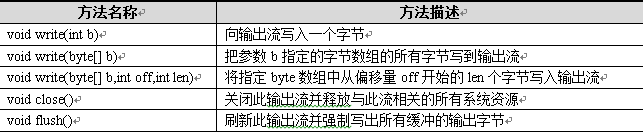




### 2.OutputStream

OutputStream是所有字节输出流顶级父类

OutputStream：



案例：

### 3.尝试：

/\*\*

\* 输入输出流

\*/

public class FileUtil {

/\*\*

\* 写操作

\* @param str

\*/

public static void fileOutput(String content,String fileName){

FileOutputStream stream = null;

try {

stream = new FileOutputStream(fileName);

byte[] bytes = content.getBytes();

for (int i = 0; i < bytes.length; i++) {

stream.write(bytes[i]);

}

} catch (FileNotFoundException e) {

e.printStackTrace();

} catch (IOException e) {

e.printStackTrace();

} finally{

try {

stream.close();

} catch (IOException e) {

e.printStackTrace();

}

}

}

/\*\*

\* 读操作

\* @param fileName

\*/

public static void fileInput(String fileName) {

File file = new File(fileName);

//1、创建输入流

FileInputStream stream = null;

try {

if (!file.exists()) {

file.createNewFile();

}

stream = new FileInputStream(file);

int length = 0;//保存读取的长度

byte[] bytes = new byte[1024];//保存读到的内容

//开始读操作

while ((length = stream.read(bytes)) != -1) {//读取一个字节

System.out.write(bytes,0,length);

}

} catch (FileNotFoundException e) {

e.printStackTrace();

} catch (IOException e) {

e.printStackTrace();

}finally{

try {

stream.close();

} catch (IOException e) {

e.printStackTrace();

}

}

}

}

public static void main(String[] args) {

String str = "学习真是很漫长的过，才第三天呀，还有12天，天啦";

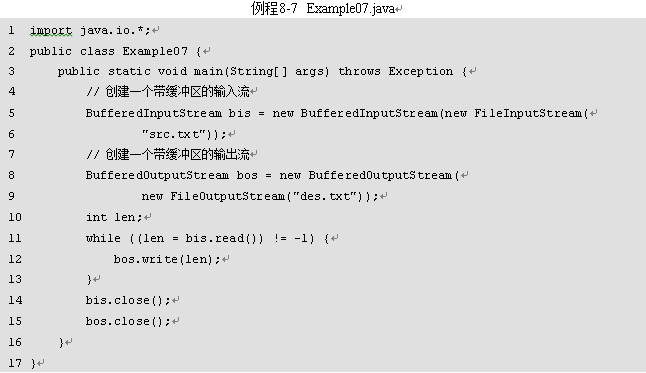
String name = "ddup";

FileUtil.fileOutput(str,"ddup");

FileUtil.fileInput(name);

}

### 4.过滤流：



### 总结：

顶层：InputStream、OutPutstream

直接子类：FileInputStream、FileOutPutstream

包装类：BufferedInputStream、BufferedOutPutstream：首选

### 字符流

顶层：Reader、Writer

子类FileReader、FileWriter

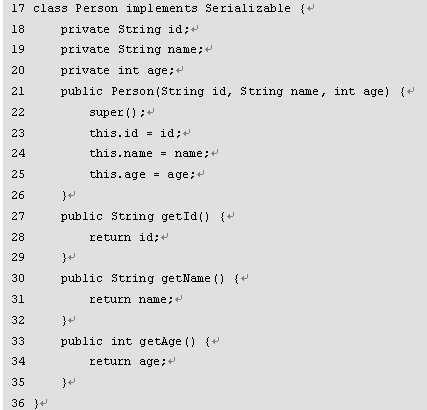
包装类：BufferedReader、BufferedWriter

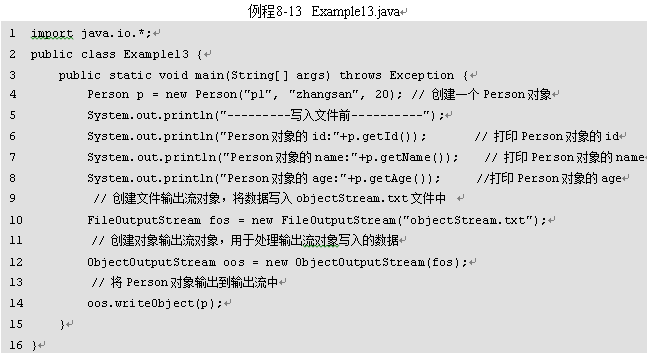
次包装类：LineNumberReader、LineNumberWriter

### 转换流

OutputStreamWriter、InputStreamReader

### 对象流





### File

Root

Tree1 tree2

First.txt second.txt

举例：

public static void create() {

//創建文件甲

File root = new File("root");

root.mkdir();

File tree1 = new File(root,"tree1");

tree1.mkdir();

File first = new File(tree1,"first");

try {

first.createNewFile();

} catch (IOException e) {

e.printStackTrace();

}

}

### 11.文件操作：

* + String getName( )； //得到一个文件的名称（不包括路径）
  + String getPath( )； //得到一个文件的路径名
  + String getAbsolutePath( )；//得到一个文件的绝对路径名
  + String getParent( )； //得到一个文件的上一级目录名
  + File getParentFile()；//返回父文件对象
  + Long length(); //返回文件的字节长度
  + Long lastModified(); //返回文件的最后修改时间
  + boolean exists( )； //测试当前File对象所指示的文件是否存在
  + boolean canWrite( )；//测试当前文件是否可写
  + boolean canRead( )；//测试当前文件是否可读
  + Boolean isHidden(); //判断文件是否是隐藏的
  + boolean isFile( )； //测试当前文件是否是文件（不是目录）
  + boolean isDirectory( )； //测试当前文件是否是目录
  + Boolean setReadOnly(); //设置文件属性为只读
  + Boolean setLastModified(long time)//设置文件的最后修改时间
  + Int compareTo(File pathname);//比较两个文件对象的内容
  + boolean renameTo(File dest ); //文件重命名
    - boolean createNewFile( )throws IOException; //创建新
  + boolean mkdir( ); //根据当前对象生成一个由该对象指定的路径
  + String[ ] list( ); //列出当前目录下的文件
  + File[] listFiles();//返回当前目录下的所有文件对象