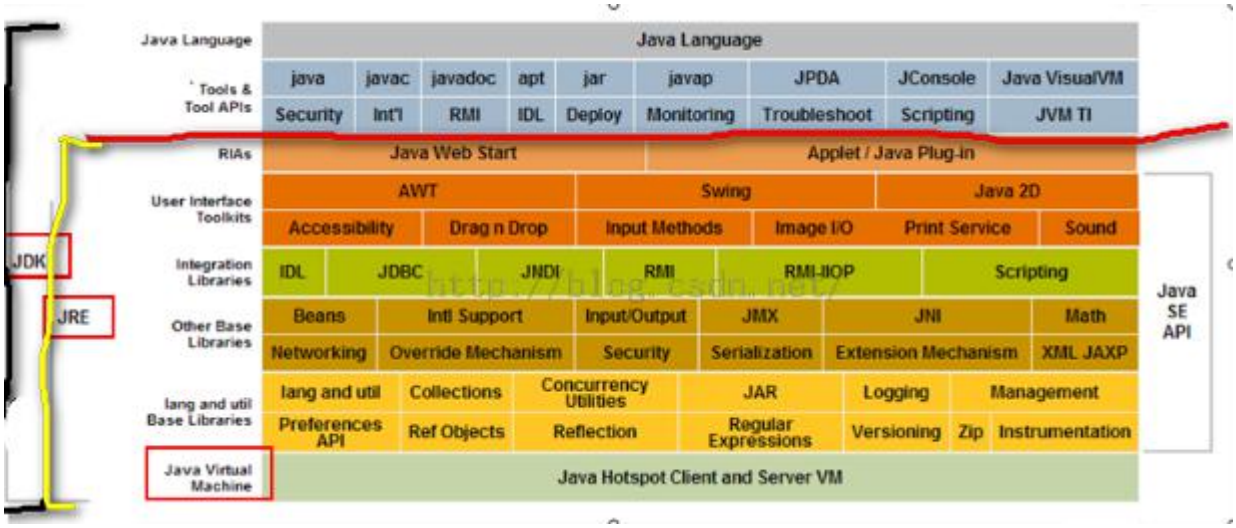
Java 基础学习笔记

1.JRE 和 JDK

JRE (Java Runtime Environment) ：Java 运行环境，包括 Java 虚拟机和 Java 所需的核心类库等。

JDK(Java Development Kit):Java 开发工具包，供开发人员使用，JDK 中包括了 JRE。

其中的开发工具包括：编译工具（javac.exe)、打包工具（jar.exe 等）



2.静态（static）

用法：修饰符用来修饰成员（成员变量、成员函数）

static 修饰内容被对象所共享。

被 static 修饰的成员可以直接被类名调用（类名.静态成员）

特有数据随对象存储

static 特点：1.随着本类的加载而加载，随着类的消失而消失。2.优先于对象存在（静态先存在，对象后存在）。3.被所有对象所共享。4.可以直接被类名所调用。

实例变量和类变量区别：1.存放位置：类变量随类的加载存在于方法区中。实例变量随对象的建立而存在于堆内存中。2.生命周期：类变量生命周期长随着类的消失而消失。实例变量生命周期随着对象的消失而消失。

静态使用注意事项：1.静态方法只能访问静态成员，非静态方法既可以访问静态也可以访问非静态。2.静态方法中不可以定义 this super 关键字，因为静态优先于对象存在所以静态方法中不可以出现 this。3.主函数是静态的。

静态的利弊：利：对对象的共享数据进行单独空间的存储节省空间；可以直接被类名调用；弊：生命周期过长；访问出现局限性。

什么时候使用静态？1.当对象中出现共享数据时，该数据被静态所修饰。对象中的特有数据要定义成非静态存在于堆内存中。

什么时候定义静态函数？当功能内部没有访问到非静态数据(对象的特有数据那么该功能可以定义成静态的)

每一个应用程序中都有共性的功能可以将这些功能进行抽取独立封装以便复用。

将方法都静态后可以方便使用但该类还可以被其他程序建立对象为了严谨强制让该类不能建立对象可以通过将构造器函数私有化完成。

静态代码块：static{ 静态代码块中执行的语句} 特点：随着类的加载而执行用于给类进行初始化。

3.抽象类

当多个类中出现功能相同，但是功能体不同，可以进行向上抽取，这时，只抽取功能定义，不抽取功能主体。

抽象的特点：1.抽象方法一定在抽象类中。2.抽象方法和抽象类都必须被 abstract 关键字修饰。3.抽象类不可以用 new 创建对象，因为调用抽象方法没意义。4.抽象类中的抽象方

法要被调用，必须有子类复写其所有的抽象方法后，建立子类对象调用。如果子类只覆盖了部分抽象方法，那么子类仍然是抽象类。

抽象类比一般类多了抽象函数。抽象类不可以被实例化。抽象只能用来修饰类和方法。

特殊：抽象类可以不创建抽象方法，这样做仅仅是不让该类创建对象。

模板方法设计模式：在定义功能时，功能的一部分是确定的，但是有一部分是不确定的，而确定的部分，在使用不确定的部分，那么这时就将不确定的部分暴漏出去，由该类的子类去完成。

4.this 用法

this 代表它所在函数所属对象的引用（简单说：哪个对象在调用 this 所在函数。this 就代表哪个对象）

this 的应用：当定义类中功能时该函数内部要用到调用该函数的对象时，这时用 this 来表示这个对象但凡本类功能内部使用了本类对象都用 this 表示。

this 语句用于构造函数之间互调。this 语句只能放在构造函数的第一行，因为初始化语句要先执行

5.关键字 final

final:最终，作为一个修饰符可修饰类、方法、变量。被 final 修饰的类不可以被继承，为了避免被继承，被子类复写功能。被 final 修饰的方法不能被复写。被 final 修饰的变量是常量，只能赋值一次，既可以修饰成员变量也可以修饰局部变量。

常量的书写规范：所有字母大写，当有多个单词组成时单词间通过下划线（_）连接。

6.接口（interface）

接口：可以认为是一个特殊的抽象类，当抽象类中的方法都是抽象的，那么该类可以通过接口的形式来表示。

class 用于定义类。

interface 用于定义接口。

接口定义时格式特点：1.接口中常见定义，常量，抽象方法。2.接口中的成员都有固定修饰符。常量：public static final ；方法：public abstract。记住:接口中的成员都是 public 的。

接口不可以创建对象，因为有抽象方法。

需要被子类实现，子类对接口中的抽象方法全都覆盖后，子类才可以实例化，否则子类是一个抽象类。

接口可以被类多实现。

接口之间可以多继承。

接口的特点：1.接口是对外暴露的规则。2.接口是程序的功能扩展。3.接口可以用来多实现。4.类与接口之间是实现关系而且类可以继承一个类的同时实现多个接口。5.接口与接口之间可以有多继承关系。

基本功能定义在类中，扩展功能定义在接口中。

事物之间的关系：聚集：has-a（一个事物中包含另一事物）；聚合：同类事物组成一个集合（如球员与球队的关系）；组合：如手是身体的一部分。

7.构造函数

构造函数特点:1.函数名与类名相同；2.不用定义返回类型；3.不可以写 return 语句；

作用：给对象进行初始化

对象一建立构造函数就被执行

当一个类中没有定义构造函数时，那么系统会默认给该类加入一个空参数的构造函数，当在类中自定义构造函数后默认的构造函数就没有了。

构造代码块：格式：{ 执行语句}；作用：给对象进行初始化，对象一建立就运行而且优于构造函数执行。

构造代码块给所有的对象进行统一初始化而构造函数是给对应的对象初始化。构造代码块中定义的是不同对象共性初始化内容。

构造函数可以被私有化。

8.理解面向对象

面向过程强调的是功能行为。

面向对象：将功能进行封装进对象强调具备了功能的对象。

1.面向对象是相对于面向过程而言的。

2.面向对象和 面向过程都是一种思想。

3.面向过程强调的是功能行为。

4.面向对象将功能封装进对象，强调具备了功能的对象。

5.面向对象是基于面向过程的。

9.单例模式（Singleton）

饿汉式(先初始化对象，Singleton 一进内存就已经创建好了对象)（开发一般用饿汉式）：

[java] class Singleton{

```
 private static Singleton s=new Singleton();
private Singleton() {}
public static Singleton   getInstance()
{
    return s;
}
```

```
}
```

懒汉式（对象时方法调用时才被初始化，也叫做对象的延时加载。Singleton 进内存时对象还没有存在，只有调用了 getInstance 方法时才建立对象）

```
[java] class Singleton{
        private static Singleton s=null;
       private Singleton() {}
       public static synchronized Singleton   getInstance()
       {
            if(s==null)
            {
                s=new Singleton();
            }
            return s;
        }
}
```

改进：

```
[java] class Singleton{
        private static Singleton s=null;
       private Singleton() {}
       public static   Singleton   getInstance()
       {
```

```
            if(s==null)
            {
                synchronized(Singleton.class)
                  {
                            if(s==null)
                                    s=new Singleton();
                    }

            }
            return s;
        }
      }
```

使用原则;定义单例建议使用饿汉式。

10.封装

封装;指隐藏对象属性和实现细节仅对外提供公共访问方式。

好处：1.将变化隔离；2.便于使用；3.提高重用性；4.提高安全性

封装原则：1.将不需要对外提供的内容隐藏起来；2.把属性都隐藏提供公共方法对其访问。

Java 中的函数本身就是一个最小的封装体。

private 私有权限修饰符，私有只在本类中有效。私有不是封装，私有仅仅是封装的一种表现形式。

成员变量都有默认值存储在堆内存中。

11.继承

继承:1.提高代码复用性。2.使类与类之间产生关系，有了这个关系才有了多态的特性。

注意：不要为了获取其他类的功能简化代码而继承，必须是类与类之间有所属关系才可以继承。所属关系 is-a。

java 语言:只支持单继承，不支持多继承，因多继承会带来安全隐患（当多个父类中定义了相同功能，当功能内容不同时，子类不确定要运行哪一个。但是 java 保留了这种机制，并用另一种体现形式来完成表示：多实现。）

java 支持多层继承。

1.变量：如果子类中出现非私有同名成员变量时，子类要访问本类中的变量用 this，子类要访问父类中的同名变量用 super。

this 代表本类对象的引用，super 代表父类对象的引用。

2.函数：当子类出现和父类一模一样的函数时，当子类对象调用该函数，会运行子类函数的内容。如同父类的函数被覆盖一样，这是函数的另一个特性重写（覆盖 override）

3.子父类中的构造函数：先执行父类的构造函数再执行子类的构造函数。在对子类函数进行初始化时父类的构造函数也会运行，是因为子类的构造函数默认第一行有一条隐式语句，super(); super 会访问父类中空参数的构造函数，而且子类中所有的构造函数默认第一行都是：super();

为什么子类一定要访问父类中的构造函数？

因为父类中的数据子类可以直接获取，所以子类对象在建立时需要先查看父类是如何对这些数据初始化的。所以子类在对象初始化时，要先访问以下父类中的构造函数。

如果要访问父类中指定的构造函数，可以通过手动定义 super 语句的方式来指定。

super 语句一定要定义在子类的构造函数的第一行。

子类的所有构造函数，默认都会访问父类中空参数的构造函数。因为子类每一个构造函数内的第一行都有一句隐式 super();

当父类中没有空参数的构造函数时，子类必须手动通过 super 或者 this 语句形式来指定要访问的构造函数。

当然子类的构造函数第一行也可以手动指定 this 语句来访问本类中的构造函数。子类中至少会有一个构造函数访问到父类中的构造函数。

没有条件的递归就是死循环。

覆盖：1.子类覆盖父类，必须保证子类权限大于等于父类权限，才可以覆盖，否则编译失败。

2.静态只能覆盖静态。

重写：子类和父类方法一模一样。

重载：只看同名函数的参数列表。

12.Java 的编译方式

Java 有两种编译方式：（1）直接把源代码编译成和本地机器平台相关的机器语言，叫即时编译；（2）把源文件编译成一种中间的字节码与机器平台无关叫解释型。

13.匿名对象

匿名对象使用方式一：当对象的方法只调用一次时可以用匿名对象来完成。这样写比较简化

如果对一个对象进行对个成员调用必须给这个对象起名字，匿名内部类调用成员变量无任何意义。

匿名对象使用方式二:可以将匿名对象作为实际参数进行传递。

14.泛型

泛型是 JDK1.5 出现的新特性，用于解决安全问题是一个类型安全机制。

好处：1.将运行时期出现的 ClassCastException 问题转移到编译时期，方便程序员解决问题，让运行时期问题减少，安全。2.避免了强制转换的麻烦。

泛型格式：通过<>来定义要操作的引用数据类型。

泛型应用场合：通常在集合框架中很常见，只要见到<>就要定义泛型 ，其实<>就是用来接收类型的。当使用集合时，集合中要存储的数据类型作为参数传递到<>即可。

什么时候定义泛型类？当类中要操作的引用数据类型不确定时，早期定义 Object 来完成扩展，现在定义泛型来完成扩展。

泛型类：

泛型可以定在类上也可以定义在方法上。

泛型类定义的泛型在整个类中有效，如果被方法使用，那么泛型类明确要操作的具体类型后，所有要要操作的类型都已经确定了。为了让不同方法可以操作 不同类型而且类型不确定，那么可以将泛型定义在方法上。

泛型方法：

特殊之处：静态方法不可以访问类上定义的泛型。如果静态方法操作的应用数据类型不确定可以将泛型定义在方法上。

静态泛型方法：

泛型定义在接口上：

<?>表示引用数据类型不明确，？通配符

泛型限定：<? extends E>可以接受 E 类型或者 E 的子类，上限。

<? super E>可以接受 E 类型或者 E 的父类

15.String 类

字符串是一个特殊的对象一旦被初始化就不能改变。

public final class String

equals 方法比较内存地址值。

常见字符串操作：

1.获取

1.1 字符串中包含的字符数也就是字符串的长度。

int length():获取长度

1.2 根据位置获取位置上的某个字符

char chartAt(int index):

1.3 根据字符获取该字符在字符串中的位置

int indexOf(int ch):返回的是在字符串中第一次出现的位置

int indexOf(int ch,int indexFrom):从 fromIndex 指定位置开始获取字符 ch 在 字符串中出现的位置。

int indexOf(String str):返回的是 str 在字符串中第一次出现的位置

int indexOf(String str,int indexFrom):从 fromIndex 指定位置开始获取 str 在 字符串中出现的位置。

int lastIndexOf(String str):返回的是 str 在字符串中最后一次出现的位置

2.判断

2.1 字符串中是否包含指定字串

boolean contains（ ）

特殊之处：indexOf（String str）返回 str 在字符串中第一次出现的位置，若果返回-1 则表示该字符串中不包含 stry 因此也可以用来判断字符串是否包含某个字串。

2.2 字符串中是否有内容

boolean isEmpity（ ）：判断字符串长度是否为 0.

2.3 字符串中是否以指定内容开头

boolean startsWith（String prefix）

2.4 字符串是否以指定内容结尾

boolean endsWith（String str）

2.5 判断字符串的内容是否相同（复写 Object 类中的 equals 方法）

boolean equals（String str）

2.6 判断字符串长度是否相同并忽略大写小

boolean equalsIgnoreCase()

3.转换

3.1 将字符数组转换为字符串

构造函数：Strirng（char[] arry）

String(char[] arry,int offset,int count):将字符数组中的一部分转换为字符串

静态方法：static String copyValueOf（char[] arry）

static String copyValueOf（char[] arry,int offset,int count）

static String ValueOf(char[] arry)

static String ValueOf (char[] arry,int offset,int count)

3.2 将字符串转换为字符数组

char[] toCharArray()

3.3 将字节数组转换为字符串

Strirng (byte[] arry)

String(bytear[] arry,int offset,int count):将字节数组中的一部分转换为字符串

3.4 将字符串转换为字节数组

byte[] getBytes()

3.5 将基本类型转换为字符串类型

static String valueOf (int|double|short|char (基本数据类型) data)

特殊：字符串和字节数组在转换过程中是可以指定编码表的。

4.替换

String replace (char oldChar，char newChar)

String replace (CharSequence target，CharSequence replacement)

5.切割

String[] split(regex)

6.子串 (获取字符串中的一部分)

String subString (int beginIndex)

String subString (intbeginIndex，int endIndex) (包含头不包含尾)

7.转换、去除空格、比较

7.1 将字符串转换为大写或小写

String toUpperCase（ ）

String toLowerCase（ ）

7.2 将字符串两端多余空格去除

String trim()

7.3 对两个字符串进行自然顺序的比较

int compareTo(String anotherString)

StringBuffer 是一个容器它可以对字符串进行增删，StringBuffer 是长度是可变的。

public final class StringBuffer

StringBuffer 特点：1.长度可变。2.可以一次操作多个数据类型。3.通过 toString 方法转化为字符串

1.存储

StringBuffer append():将指定数据作为参数添加到已有数据尾处。

StringBuffer insert(index,data):在指定位置处插入数据。

2.删除

StringBuffer delete(start ,end ）:删除缓冲区中的数据包含 start ,但不包含 end。

StringBuffer deleteCharAt（index）：删除指定位置的字符。

3.获取

StringBuffer charAt（index）

int indexOf()

int lastIndexOf()

String subString(start,end)

4.修改

replace(int start,int end.String str)

void setCharAt(int index,char ch)

5.翻转

StringBuffer reverse（ ）

6.void getChars（int srcBigin，int srcEnd,char[] dest,int dstBegin）将缓冲区中的指定字符保存到字符数组。

StringBuilder：StringBuffer 的简易替换，不保证线程安全。

JDK1.5 出现 StringBuilder，与 StringBuffer 的区别是：StringBuffer 是线程同步的，StringBuilder 是线程不同步的。

升级三个因素：1.提高效率 2.简化书写 3.提高安全性

16.Jar 包

好处：1.方便与项目的携带。2.方便使用，只要在 classpath 中设置 jar 路径即可。3.数据库驱动，SSH 框架等都是以 jar 包体现。

jar 包的使用：

17.异常

异常:程序在运行时出现的不正常情况。

异常的由来：问题也是现实生活中一个具体的事物，也可以通过 java 的类的形式进行描述，并封装成对象。其实就是 java 对不正常情况进行描述后的对象体现。

对于问题的划分：两种:一种是严重的的问题；一种是非严重的问题。对于严重的 java 通过 Error 类进行描述。对于非严重的 java 通过 Exception 进行描述。

对于 Error 一般不编写针对性的代码对其进行描述。对于 Exception 可以使用针对性的处理方式进行处理。

Throwable(直接子类：Error 类　Exception 类)

java 异常处理的机制：

```
try{
    被检查代码块
```

}catch(异常类　变量){

**　处理异常的代码（处理方式）**

}finally{

**　　一定被执行的语句**

}

对捕获到的异常对象常见方法操作：1.String　getMessage()　2.String toString()　3.void printStackTrace()　jvm 默认的异常处理机制就是在调用 printStackTrace()方法，打印异常在堆栈中的跟踪信息。

在函数上声明异常，便于提高安全性让调用者进行处理，不处理编译失败。

对多异常的的处理：1.声明异常时建议声明为更具体的异常。这样处理的可以更具体。2. 声明几个异常就对应几个 catch 块。如果多个 catch 块中的异常出现继承关系，父类异常 catch 块放在最下面。

建议在进行异常处理时，catch 中一定要定义具体处理方式。

当在函数内部出现 throw 抛出异常对象，那么就必须给出处理动作。要么在内部 try-catch 处理要么在函数上声明让调用者处理。

一般情况在函数内出现异常，函数上需要声明。

自定义异常：必须是自定义类继承 Exception。

继承 Exception 的原因：异常体系有一个特点，因为异常类和异常对象都被抛出。他们都具有可抛性，这个可抛性是 Throwable 这个体系独有的特点。只有这个体系中的类和对象才可以被 throws 和 throw 操作。

throws 和 throw 的区别：1.throws 使用在函数上;throw 使用在函数内。2.throws 后面跟的是异常类可以跟多个用逗号隔开；throw 后面跟的是异常对象。

Exception 中有一个特殊的子类异常 RunTimeException 运行时异常。如果在函数内抛出该异常可以不用声明编译一样通过。如果在函数上声明了该异常，调用者可以不用进行处理。编译一样通过。

之所以不用在函数声明，是因为不需要让调用者处理。当该异常发生希望程序停止，因为在运行时出现无法继续运算的情况，希望停止后对代码进行修正。常见的 RunTimeException 异常：Arithmetic Exception、ClassCastExcepton、NullPointerException、IndexOutOfBoundsException、IllegalArgumentException 等。

自定义异常时如果该异常的发生无法再继续进行运算就让自定义异常类继承 RunTimeException。

对于异常分两种：1.编译时被检测的异常。2.编译时不被检测的异常（RunTimeException 及其子类）

finally 一定被执行的代码。finally 代码块定义一定执行的代码，通常用于关闭资源。因为资源必须要释放。

异常处理格式：格式 1.try{}catch(){}。格式 2.try{}catch(){}finally()。格式三：try{}finally{}。

记住：catch 是用来处理异常的，如果没有 catch 就代表异常没被处理过，那么必须声明。

异常在父类覆盖中的体现：1.子类在覆盖父类时如果父类的方法抛出异常，那么子类的覆盖方法只能抛出父类的异常或者该异常的子类。2.如果父类方法抛出多个异常，那么子类在覆盖该方法时只能抛出父类异常的子集。

3.如果父类或者接口的方法中没有异常抛出，那么子类在覆盖方法时也不能抛出异常。如果子类方法发生异常就必须要进行 try 处理，不能抛。

finally 只有一种情况读不到：就是在 finally 语句前存在 System.exit(0);系统退出，jvm 结束。

或者异常可以处理，但需要将异常产生的和本功能相关的问题提供出去，让调用者知道并处理，也可以将捕获异常处理后转换成新的异常。

```
异常的注意事项：
    在子父类覆盖时：
    1，子类抛出的异常必须是父类的异常的子类或者子集。
    2，如果父类或者接口没有异常抛出时，子类覆盖出现异常，只能try不能抛。
```

18.object 类

object 类是所有类的直接或间接父类。

该类中定义的是所有对象都具备的功能。

19.多态

多态：可以理解为事物存在的多种体现形态。

1.多态的体现 :父类的引用指向了自己的子类对象。父类的引用也可以接受自己的子类对象。

2.多态的前提 :必须类与类之间有关系，要么继承要么实现。通常还有一个前提，存在覆盖。

3.多态的好处：多态的出现大大提高了程序的可扩展性。

4.多态的弊端：只能使用父类的引用访问父类中的成员。

类型提升：向上转型。

强制将父类的引用转换为子类类型：向下转型。

```
class Fu{
    ……
  }
class Son extends Fu{
    ……
}
class Cast{
    public static void main(String[] args)
    {
            Fu fu=new Son();              //向上转型，类型提升
            Son son=(Son) fu;          //向下转型，强制将父类的引用，转换成子类类
型。
    }
}
```

20.集合类

集合类：面向对象语言对事物的体现都是以对象的形式，为了方便对多个对象进行操作，就对对象进行存储，集合就是存储对象最基本的一种方式。

数组和集合的区别：数组虽然也可以存储对象但长度是固定的，集合是可变的。数组中可以存储基本数据类型但集合只能存储对象。

集合类的特点：集合只用于存储对象，长度可变，结合可以存储不同类型的对象。

为什么会有那么多容器？因为每个容器对数据的存储方式不同。这个存储方式称为数据结构。

add 方法的参数类型是 Object 以便接受任何类型

集合中存储的都是对象的引用。

Collections 中包含了一些操作集合对象的方法，具体方法请参考： http://tool.oschina.net/apidocs/apidoc?api=jdk-zh

public interface iterator<E>:方法：hasNext()　next()　remove()

迭代器就是集合取出元素的方式。

List 元素是有序的，元素可以重复，因为该集合体系中有索引。

Set 元素是无序的，元素不可以重复。

List 特有方法：凡是可以操作角标的方法都是 List 特有方法。

增

add(index,element)

add(index,Collection)

删

remove(index)

查

get(index)

subList(from,to)

listIterator()

改

set（index，element）

List 集合特有的迭代器 ListIterator 是 Iterator 的子接口，该接口只能通过 List 集合的 listLiterator 方法获取。ListIterator 可以在遍历过程中进行增删查改。

ArrayList：底层的数据结构使用的是数组结构。特点：查询速度很快但是增删稍慢。线程不同步。初始容量为 10，增速为 1.5 倍。

LinkedList：底层使用的是链表数据结构。特点:增删速度很快，查询稍慢。

Vector：底层是数组数据结构。线程同步。初始容量为 10，增速为 2 倍。

Enumeration 是 Vector 特有的取出方式。其实枚举和迭代是一样的.

LinkedList 特有方法：

addFirst()　addLast()

获取元素不删除元素 如果集合中没有元素会出现 NoSuchElementException ：

getFirst()　getLast()

获取元素删除元素如果集合中没有元素会出现 NoSuchElementException：

removeFirst()　removeLast()

JDK1.6 版本出现的替代方法：offerFirst()　offerLast()

获取元素不删除元素 如果集合中没有元素 会返回 nul ：peekFirts()　peekLast()

获取元素删除元素 如果集合中没有元素会返回 null ：pollFirst() pollLast()

堆栈：先进后出

队列：先进先出

在迭代时循环中 next 调用一次，就好 hasNext 判断一次。

Object 类中的 equals 方法比较的是内存。

List 集合判断元素是否相同依据的是 equals 方法。

set：元素是无序（存入或取出的顺序不一定一致。） 元素不可以重复。

set 集合的功能和 collection 是一致的。

hashSet 底层数据结构是哈希表。（哈希表存放一堆哈希值的表，先判断哈西值是否一样，如果一样再判断是否为同一对象），线程是非同步的。

hashSet 保证元素唯一性的原理 :通过元素的两个方法 hashCode 和 equals 方法来完成，如果元素的 hashCode 的值相同才会判断 equals 是否为 true，如果元素的 hashcode 不同不会调用 equals 方法。

对于判断元素是否存在以及删除等操作依赖的时 hashCode 和 equals 方法。

TreeSet 可以对 Set 集合中的元素进行排序。DI 底层数据结构是二叉树，保证数据唯一性的依据是 compareT 方法 return 0。

实现比较的接口：public interface Compare<T> 此接口强行对实现它的每个类的对象进行整体排序，这种排序被称为类的自然排序。

方法：int CompareTo(T o)返回负整数、零、正整数，根据此对象对象是大于、等于、还是小于指定对象。

排序时当主要条件相同时一定要判断一下次要条件。

TreeSet 排序的第一种方式：让元素自身具备比较性，元素需要实现 Compare 接口，覆盖 compareTo 方法

TreeSet 排序的第二种方式：当元素自身不具备比较性，或者具备的比较性不是所需要的，这时就需要让集合自身具备比较性。在集合初始化时就有了比较方式。TreeSet（Comparator <? super E> comparator)实现 Comparator 接口覆盖 compare 方法（int compare(Object obj1,Object obj2)）

当两种比较都存在时，以比较器为主。定义一个类实现 comparator 接口覆盖 compare 方法。

21.基本类型封装类

基本数据类型与包装类型对应关系：

1.short Short 2.byte Byte 3.int Integer 4.char Character 5.boolean Boolean 6.long Long 7.float Float 8.double Double

基本数据类型对象包装类最常见的作用就是：用于基本数据类型和字符串之间转换。

基本数据类型转字符串：基本数据类型+“” 基本数据类型.toString（基本数据类型值）

字符串转基本数据类型： static parseInt（字符串 ）

十进制转其它进制： Integer.toBinaryString(int data) Integer.toHexString(int data)

Integer.toOctalString(int data)

其它进制转十进制：parseInt(string,radix)

自动拆箱自动装箱：

```
class A
{
```

Integer a=new Integer(4);

Integer b=4; //自动装箱（new Integer（4））

b=b/* b.intValue */+2; //b 进行自动拆箱变成 int 型，和 2 进行运算，然后再将结果自动装箱赋给 b。注意 null 值

Integer m=128；

Integer n=128;

Integer a=127;

Integer b=127;

判断 m==n false

判断 a==b true

原因：a 和 b 指向了同一个对象，当数值在 byte 范围内时对于新特性如果该数值应经存在不会开辟新空间。

}

22.内部类

内部类定义形式：

```
class Outer{
    ……
    class Inner{
        ……
    }
}
```