Java

1. Java概述
   1. 什么是Java

特性：免费且开源、平台无关性

诞生：1995.5.23

* 1. Java平台分类
* Java SE——Java Standard Edition，Java标准版（Java Core）
* Java EE——Java Enterprise Edition，Java企业版（Java 14种技术应用集合）
* Java ME——Java Micro Edition，Java微小版
  1. Java开发环境的搭建

步骤：

* 安装JDK
* 配置环境变量 path——指明Java开发工具的位置（bin文件夹）

classpath——指明Java运行环境的位置（rt.jar文件）

* 1. Java程序的开发过程

1. Java应用程序的开发步骤

编辑 字节码 结果

（.java） 编译操作（javac） （.class） 解释运行操作（java）

1. Java Applet的开发步骤

编辑 字节码 结果

（.java） 编译操作（javac） （.class） 解释运行操作（html文件）

**注意：**（1）Java程序的源文件名必须与public类名保持一致

（2）一个Java源文件中，可以有多个类，但最多只能有一个public类，若无public类，则源文件名与任意类名一致即可

（3）含有main()方法的类被称为应用程序的主类，继承Applet类的类被称为小应用程序的主类

第二章 Java语法基础

2.1 标识符、关键字和保留字

1、标识符

规则：（1）由字符、数字、下划线和$所构成

（2）只能用字符、下划线和$开头

（3）不能使用关键字和保留字

2、关键字和保留字

由系统预留，不能由用户支配

2.2 变量——数据类型

1、整型——整数

（1）字节型——byte

例：byte a;

空间：1个字节

（2）短整型——short

例：short a;

空间：2个字节

（3）基本整型——int

例：int a;

空间：4个字节

（4）长整型——long

例：long a;

空间：8个字节

2、浮点型——带有小数部分的数据

（1）单精度——float

例：float f = 3.14f;

空间：4个字节

（2）双精度——double

例：double a = 3.14;

空间：8个字节

注意：在Java中，浮点型常量默认当作双精度类型进行处理

3、逻辑型——boolean

例：boolean f = true/false

4、字符型——char

例：char c = ‘x’;

char cc = ‘/n’;

空间：2个字节

2.3 精度排序

byte、short、char、int、long、float、double

低 高

2.4 不同精度之间的数据处理

1、较低进度数据赋值给较高精度变量

原则：直接赋值

例如：double x = 100;

影响：精度扩展

2、较高精度数据赋值给较低精度变量

原则：使用强制类型转换

例如： int a = (int)3.14;

影响：精度缺失

2.5 输入与输出操作

1、输入

步骤：（1）引入Scanner类——import java.util.\*/Scanner;

（2）Scanner scan = new Scanner(System.in);

相关方法：nextInt()、nextFloat()、next().charAt(0)……

2、输出

* System.out.println()
* System.out.print()

2.6 运算符和表达式

1、算术运算符

种类：+、-、\*、/、%

注意：运算结果的类型以最高精度为准

2、++、--运算符

3、关系运算符

4、逻辑运算符

5、位运算符

6、赋值运算符

7、条件运算符

8、instanceof运算符

格式：对象名 instanceof 类名

2.7 语句

1、条件控制语句

（1）if语句

（2）if…else语句（是否）

（3）if…else if…else

（4）switch语句（判断语句）

格式：switch(产生byte、short、int、char型数值的表达式){

case byte/short/int/char型值：执行内容;[break;]

……

[default：执行内容;]

}

2、循环控制语句

（1）while语句

（2）do…while语句

（3）for语句

格式：for(表达式1; 表达式2;表达式3){

循环语句;

} 循环主体

3、流程控制语句

（1）break语句——switch或循环中，立刻结束

（2）continue语句——循环中，立刻结束本次循环，转而开始一次新的循环

（3）return语句——方法执行结束

2.8 数组——具有相同类型的数据构成的大小已定的集合

（1）创建数组

格式：数组类型[] 数组名 = new 数组类型[元素个数];

数组声明 数组内存空间分配

例：int[] a = new int[5];//int a[5];

数组的内存空间表示：

0

0x1101

0

0x1101

a

0

（2）数组的使用

格式：数组名[下标]

0~数组长度-1

注意：数组长度——数组名.length

（3）数组的数组

（4）增强型for循环——针对集合进行遍历操作

格式：for(变量定义:集合名){

……//定义的变量将代表每一个集合元素

}

例：int[] a = new int[10];

for(int x:a){

System.out.println(x);

}

第三章 Java面向对象编程

1、什么是面向对象

OO，Object-Oriented

2、面向对象三大特性

* 封装性——隐藏细节，暴露接口
* 继承性
* 多态性

3、类——Java程序的基本单元

（1）定义

格式：class 类名{

声明 类体

}

（2）类中的内容

类 属性——成员变量

功能——方法

（3）成员变量——在类中、方法之外定义的变量

（4）方法

格式：返回值类型 方法名([参数列表]){

方法声明 方法体

}

注意：

* 方法中也可定义变量，这些变量被称为局部变量
* 如果成员变量与局部变量命名重合，则局部变量覆盖成员变量作用域，此时，若破除局部变量的作用域，需要使用this关键字——this.成员变量

（5）方法重载

定义：在一个类中，出现同名方法——名称一致，但这些方法的参数不同，即参数的个数不同或参数的类型不同

（6）构造方法

定义：方法名与类名相同，但方法无返回值类型声明

注意：

* 构造方法用于完成成员变量的初始化处理
* 一个类中如果没有自定义的构造方法，系统会自动分配一个无参且由public修饰的构造方法；如果自定义了构造方法，则系统不再分配

4、对象——对象是类的实例化表现，类是对象的抽象

（1）定义

格式：类名 对象名 = new 构造方法([参数列表]);

例：Triangle t = new Triangle();

对象声明 对象内存空间分配

（2）对象的使用

格式：对象名.成员变量/方法([参数列表])

（3）对象的引用

地址

引用类型的变量

例：Rect r1 = new Rect();

Rect r2 = new Rect();

r1.length = 100;

r2 = r1;

System.out.println(r2.length);

r2.length = 50;

System.out.println(r1.length);

**注意：**传递对象，即为传递对象的引用，导致对象实体共享

5、static关键字——在类加载到内存中即分配空间

静态的、类的

修饰 成员变量——类成员变量，简称类变量——无修饰的成员变量是实例变量

方法——类方法——无修饰的方法是实例方法

特点：

* 类变量可以通过类名调用，类变量被该类所定义的对象共享
* 类方法可以通过类名调用
* 类方法中只能使用类变量和类方法，实例方法中既可以使用类变量、类方法，也可以使用实例变量、实例方法

静态代码块：

static{

……//先于其它语句的执行，与类的加载一起完成

}

6、this关键字

this关键字使用在实例方法中，代表使用当前方法的那个对象

this所在的方法

7、包——是对类的封装

（1）打包

格式：package 包名；

标识符.标识符

注意：位于源程序首行

（2）引包

格式：import 包名.\*/.类名；

引入包中所有的类

8、访问权限——使用权限

（1）私有权限

关键字：private

修饰内容：成员变量和方法

特点：仅限于一个类中的访问

（2）友好权限

关键字：无

修饰内容：成员变量、方法、类

特点：仅限于一个包中的访问

（3）受保护权限

关键字：protected

修饰内容：成员变量和方法

特点：同一包中和不同包的子类中

（4）公有权限

关键字：public

修饰内容：成员变量、方法、类

特点：任意位置

第四章 深入面向对象

1、继承

格式：class 子类 extends 父类{

……//语句 表示继承关系

}

2、继承性

* 子类和父类在同一包中时，子类继承父类中非private权限的成员变量和方法——相当于在子类中直接定义
* 子类和父类不在同一包中时，子类继承父类中public权限及protected权限的成员变量和方法

**注意：**protected权限修饰的成员变量和方法仅限于子类中使用

3、继承中成员变量的创建

例：class X{

private int a;

}

class Y extends X{

int b;

int operation(){

return b+a++;

}

}

……

Y y = new Y();

System.out.println(y.operation());

**注意：**在创建子类对象时，会将父类的成员变量一并开辟空间

4、Object类

Java中所有类的总父类，继承关系默认加载

5、继承的特性

* Java只支持单继承
* 继承具有传递性

6、成员变量的隐藏和方法重写

（1）成员变量的隐藏

子类中定义的成员变量名与父类中定义的成员变量名相一致，此时，子类成员变量隐藏父类成员变量。

（2）方法重写

子类中定义的某个方法，其方法名、方法的返回值类型、方法的参数个数和参数类型均与父类中某个方法相一致，此时，产生子类方法对父类方法的重写。

**注意：**进行重写时，不可降低方法的访问权限

7、super关键字

（1）解除成员变量隐藏和方法重写

格式：super.成员变量/方法()

（2）调用父类构造方法

位置：在子类构造方法第一行，若省略，则自动完成

格式：super([参数])

**注意：**在子类构造方法中，一定会进行父类构造方法的调用，或显示完成（写明super的调用），或隐式完成（默认执行super()）。

8、final关键字

修饰：变量——仅可进行一次赋值

方法——不可被重写

类——不可被继承

9、abstract关键字

修饰：类

方法

特点：

* 抽象类仅可进行对象声明，不可进行对象创建
* 抽象方法位于抽象类中，仅有方法声明，无方法体

例：abstract class A{

abstract void xxx();

}

* 抽象类中可以有非抽象方法
* 抽象类的非抽象子类必须将抽象类中的抽象方法全部进行重写

写明方法体

* 与final互斥

10、向上造型

定义：父类对象获得了子类对象的引用

上转型对象

特点：

* 上转型对象可以使用子类继承的成员变量和方法
* 上转型对象可以使用父类中被隐藏的成员变量和子类中重写的方法
* 上转型对象不可以使用子类新增的成员变量和方法
* 可将上转型对象进行强制类型转换，还原成子类对象

向下造型

11、多态

定义：相同的功能，在不同的情况下，由同一对象使用时产生不同的结果

静态造型——对象使用什么样的成员变量，由对象的类型决定

动态绑定——对象调用什么样的方法，由对象获得的引用决定

12、接口

（1）格式 接口声明

interface 接口名{

//接口体

}

（2）使用特点

* 接口中只能定义抽象方法和类常量，格式：

interface X{

[public static final] double A = 3.1;

[public abstract] double getA();

}

* 接口需通过类的使用，发挥作用，格式：

class Y implements X 或 class Y extends Z implements X

* 非抽象类使用了接口，必须将接口中的接口方法进行重写，格式：

class Y implements X{

public double getA(){

……

}

}

* 一个类使用了接口，则该类继承接口中的类常量，该类对象可以调用类中重写的接口方法，例：

Y y = new Y();

y.getA();

System.out.println(Y.A);

（3）接口回调

定义：接口声明的对象获得了使用该接口的类定义的对象的引用，此时，该接口对象可以调用类中重写的接口方法

例：X x;//声明一个接口对象

Y y = new Y();//使用X接口的Y类定义了对象y

x = y;

x.getA();//接口回调

第五章 异常处理

1、什么是异常处理

程序运行时出现的错误

2、异常处理的结构

分类： 运行时异常——RuntimeException类

Exception类

非运行时异常

3、异常处理格式

try{

//可能产生异常的语句

}catch(异常类对象声明){

//完成一种异常的捕获和处理

}

……

[finally{

//无论try还是catch，最后均需要执行的内容

}]

注意：父类异常对象可以接收抛出的子类异常对象

4、Exception类简介

* public String toString()——获取异常详细信息
* public String getMessage()——获取异常简要信息
* public void printStackTrace()——输出异常栈路径

5、throw关键字和throws关键字

* throw关键字——抛出异常对象，格式：throw 异常对象; 例：throw new Exception();
* throws关键字——为方法声明异常，格式：方法返回值类型 方法名()throws 异常类名

例：int getX()throws Exception

注意：利用throws进行异常声明，则throws所添加的方法，其调用可置于try…catch中，进行声明的异常的捕获。

6、自定义异常

步骤：

* 创建继承Exception类的自定义异常类，重写toString()/getMessage()方法
* 创建抛出异常的方法，使用throws进行声明。在方法体内，利用throw抛出自定义将异常对象

注意：当利用throws声明自定义异常时，throws所设置的方法的调用必须位于try…catch中，或者该方法所在的方法声明自定义异常。

第六章 常用实用类

1、Object类——总父类

2、Class类——反射类

3、System类

4、String类——字符串类

（1）创建对象

* String s = new String();
* String s = “abc”;

（2）常用方法——不会更改自身字符串的内容

* public int length()
* public boolean equals(String s)——判断内容相等
* public boolean startsWith(String s)
* public boolean endsWith(String s)
* public int compareTo(String s) end-1
* public String substring(int start,int end)
* public int indexOf(char c)
* public char charAt(int position)

int:Integer byte:Byte long:Long short:Short

float:Float double:Double Boolean:Boolean

char:Character

5、字符串与基本数据类型值之间的转换

封装类.parseXxx(字符串值)

字符串 基本数据类型值

数值+“”/String.valueOf(数值)

6、字符串对象的创建

* public String(byte[] b,String set)

例：String s = new String(new byte[]{0,1,2,4},”utf-8”);

* public String(char[] c)

例：String s = new String(new char[]{‘a’,’b’,’c’});

注意：

* 字符串转成字节数组——public byte[] getBytes()
* 字符串转成字符数组——public char[] toCharArray()

7、StringBuffer类

特点：操作方法将改变自身的内容

（1）构造方法

* public StringBuffer()
* public StringBuffer(String s)
* public StirngBuffer(int capacity)

（2）常用方法

* public int length()
* public int capacity()
* public StringBuffer reverse()
* public StringBuffer delete(int index)

注意：StringBuffer转成String，需使用String的构造方法——public String(StringBuffer sf)

8、StringTokenizer类——字符串分析器类

（1）构造方法

* public StringTokenizer(String s)——默认分隔符（空格、Tab）
* public StringTokenizer(String s,String x)

指定的分隔符

（2）常用方法

* public int countTokens()
* public String nextToken()
* public boolean hasMoreTokens()

9、Date类和Calendar类

注意：在java.util类包中

（1）Date常用方法

* 构造方法——public Date() //获取当前系统时间

——public Date(long millis) //利用millis代表的毫秒值创建Date对象

* 输出Date，进行格式化——首先，创建SimpleDateFormat类的对象，定义格式，之后，利用其format(Date date)方法完成Date对象的格式化处理

（2）Calendar常用方法

设定日期及时间，以及获取Calendar所表示的日期和时间

* public static Calendar getInstance() //创建Calnedar类的对象
* public void set(year,month,day) //利用参数设定Calendar对象所表示的时间
* public void setTime(Date date) //利用参数date对象的时间设定Calendar
* public int get(Calendar.类常亮) //利用参数获取时间值

10、Math类

（1）类常量

* public static final double PI //圆周率
* public static final double E //数学常数

（2）类方法

* public static double sqrt(int i)
* public static double random() //产生一个开区间0~1之间的随机值

11、正则表达式

定义：用于进行格式设定的字符串

格式构成：

（1）元字符

* Java的基本字符
* 转译字符——‘/d’、‘/D’、‘/w’、‘/W’、‘/s’、‘/S’……
* []——表示区间，[a,b,c]、[a-c]、[a-c[m-z]]、[^abc]、[a-f&&[e-m]]
* .——表示任意一个字符

（2）限定符

* ？——X？，指定元字符X出现一次或0次
* +——X+，指定元字符X出现一次或多次
* \*——X\*，指定元字符X出现0次或多次
* {n/n,/n,m}——X{n}，元字符出现n次；X{n,}，元字符至少出现n次；X{n,m}，元字符至

少出现n次，至多出现m次

* |——X|Y，指定出现X或Y

判断符合正则表达式的方法——public boolean matches(“正则表达式”)

第七章 泛型和集合框架

1、泛型简介

广泛的类型

格式：<E>

2、集合框架——位于java.util包中，长度可变

分类：List——双链表

Collection的子接口

Set——数据集

Map——映射

（1）List<E>接口

指定集合中元素的类型

List<E>

LinkedList<E> ArrayList<E>

常用方法：

* public int size()——获取集合中元素个数
* public void add(E o)­——向集合中添加元素
* public E remove(int index)——将index位置的元素从集合中删除
* public void set(int index,E o)——将index位置的元素设定为对象o
* public E get(int index)——获取index位置的元素

从0计

* public Iterator<E> iterator()——将List集合转成一个迭代对象

（2）Set<E>接口——必须保证Set<E>中数据唯一

注意：存储于Set<E>中的元素，其所属类必须使用Comparable接口

Set<E>

TreeSet<E> HashSet<E>

常用方法：

* public int size()
* public boolean add(E o)
* public boolean remove(E o)
* public Iterator<E> iterator()

（3）Map<K,V>接口——散列映射

键的类型 值得类型

Map<K,V>

HashMap<K,V> HashTable<K,V>

常用方法：

* public int size()
* public V put(K key,V value)
* public V remove(K key)
* public V get(K key)
* public Collection values()

第八章 Java多线程

1、什么是多线程

* 进程：一个程序从开始到结束的一次完整执行过程
* 线程：一个进程中，较为独立的程序执行单元
* 多线程机制：多个线程在同一时刻所模拟出的同时执行的效果

2、线程的生命周期

阶段： 新建——创建线程对象

运行——占据资源

中断——暂时释放资源（系统控制、线程阻塞、休眠、等待）

死亡——线程执行完毕

死亡状态

等待队列

运行状态

新建状态

中断状态

3、线程的使用

（1）创建Thread类的子类

注意：重写Thread类中的public void run()方法，完成线程执行内容的实现

（2）定义使用Runnable接口的类

注意：

* 在类中重写接口方法——public void run()
* 创建Thread类的对象，构造方法参数为使用Runnable接口的类定义的对象（目标对象）

4、线程中的常用方法

Thread类

* public void start()——将线程放入等待队列
* public void run()——线程运行
* public static void sleep(long mills)——使线程休眠millis毫秒（转入中断状态）
* public void interrupt()——将正在休眠的线程叫醒，将导致该线程抛出

InterrupedException异常

* public static Thread currentThread()——获取正在运行的线程对象
* public void setName(String name)——为线程设定名字
* public String getName()——获取线程的名字
* public boolean isAlive()——判断线程是否“活着”（未进入死亡状态）

5、线程中的共享和独享

（1）共享

同一目标对象的成员变量

（2）独享

run()方法中的局部变量

6、线程同步——锁操作

定义：当多个线程使用同一目标对象时，将共享该对象的成员变量，此时，通过同步操作将使该共享情况即使在线程休眠（sleep()）时也无法进行，即为线程同步

实现方法：

* 为方法添加synchronized修饰，例：public synchronized void xxx()
* 利用同步语句块，格式：

synchronized(对象){

……//同步操作语句

}

解除线程同步的方法：

* 线程执行结束
* 线程对象执行wait()方法，使同步中断

使用notifyAll()方法解除wait()引起的中断

第九章 JDBC

1、什么是JDBC

JDBC（Java DataBase Connectivity，Java数据库连接）是Java EE中14个技术规范之一，用以完成Java数据库访问操作的类和接口的集合

2、JDBC的使用

步骤：

（1）加载驱动

Class.forName(“驱动串”);

com.mysql.jdbc.Driver

（2）建立连接

DriverManager.getConnection(“连接串”, “用户名”, “密码”);

jdbc:mysql://localhost:3306/数据库名 root 0000

（3）创建执行SQL语句的对象

Connection con = DriverManger.getConnection(“”, “”, “”);

Statement st = con.createStatement();

（4）执行SQL语句

* 增：insert into 表名[(字段名1,字段名2…)] values(值1,值2…)
* 删：delete from 表名 where 字段名 = 值 and 字段名 = 值 ……
* 改：update 表名 set 字段名 = 值,字段名 = 值…… where 子句
* 查：select \*/字段名1,字段名2,…… from 表名 where 子句

int count = st.executeUpdate(“增/删/改”);

SQL语句执行成功后，所涉及的记录的条数

ResultSet rs = st.executeQuery(“查”);

while(rs.next){

System.out.println(rs.getString(1));

System.out.println(rs.getString(2));

……

}

（5）释放资源

if(rs != null){

rs.close();

}

st.close();

con.close();

注意：引入mysql的jar包

3、预处理操作

对Statement对象进行替换，更改为PreparedStatement对象，格式：

PreparedStatement pre = con.prepareStatement(“sql”);

pre.setXxx(指定’?’位置，替换的值);

从1记

……

pre.executeUpdate()/executeQuery();

4、模糊查询——关键字查询

格式：select \*/字段名1,字段名2…… from 表名 where 字段名 like ‘%\_关键字内容’

代表0个或多个字符 一个字符

注意：若使用通配符（?）替换模糊查询的格式，则需在setString(…)方法中使用%和+进行设置，例：pre.setString(1,”%”+name+”%”);