

千锋Java学院出品

# 常用类

Java Platform Standard Edition

# 课程目标

## CONTENTS

ITEMS **1** 内部类

ITEMS **2** Object类

ITEMS **3** 包装类

ITEMS **4** String

ITEMS **5** BigDecimal类

ITEMS **6** 时间类型

# 内部类

Java Platform Standard Edition





# 什么是内部类

- 概念：在一个类的内部再定义一个完整的类。
- 特点：
  - 编译之后可生成独立的字节码文件。
  - 内部类可直接访问外部类的私有成员，而不破坏封装。
  - 可为外部类提供必要的内部功能组件。

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

编译

 Outer\$Inner.class  
 Outer.class

- 在类的内部定义，与实例变量、实例方法同级别的类。
- 外部类的一个实例部分，创建内部类对象时，必须依赖外部类对象。
  - `Outer out = new Outer();`
  - `Outer.Inner in = out.new Inner();`
- 当外部类、内部类存在重名属性时，会优先访问内部类属性。
- 成员内部类不能定义静态成员。

- 不依赖外部类对象，可直接创建或通过类名访问，可声明静态成员。
- 只能直接访问外部类的静态成员（实例成员需实例化外部类对象）。
  - `Outer.Inner inner = new Outer.Inner();`
  - `Outer.Inner.show();`

- 定义在外部类方法中，作用范围和创建对象范围仅限于当前方法。
- 局部内部类访问外部类当前方法中的局部变量时，因无法保障变量的生命周期与自身相同，变量必须修饰为final。
- 限制类的使用范围。



- 没有类名的局部内部类（一切特征都与局部内部类相同）。
- 必须继承一个父类或者实现一个接口。
- 定义类、实现类、创建对象的语法合并，只能创建一个该类的对象。
- 优点：减少代码量。
- 缺点：可读性较差。

# Object类

Java Platform Standard Edition

- 超类、基类，所有类的直接或间接父类，位于继承树的最顶层。
- 任何类，如没有书写extends显示继承某个类，都默认直接继承Object类，否则为间接继承。
- Object类中所定义的方法，是所有对象都具备的方法。
- Object类型可以存储任何对象。
  - 作为参数，可接受任何对象。
  - 作为返回值，可返回任何对象。

# getClass()方法

---

- `public final Class<?> getClass(){}`
- 返回引用中存储的实际对象类型。
- 应用：通常用于判断两个引用中实际存储对象类型是否一致。

# hashCode()方法

- `public int hashCode() {}`
- 返回该对象的十进制的哈希码值。
- 哈希算法根据对象的地址或字符串或数字计算出来的int类型的数值。
- 哈希码并不唯一，可保证相同对象返回相同哈希码，尽量保证不同对象返回不同哈希码。

# toString()方法

---

- `public String toString(){}`
- 返回该对象的字符串表示（表现形式）。
- 可以根据程序需求覆盖该方法，如：展示对象各个属性值。

# equals()方法

---

- `public boolean equals(Object obj){}`
- 默认实现为(`this == obj`), 比较两个对象地址是否相同。
- 可进行覆盖, 比较两个对象的内容是否相同。

# equals()方法覆盖步骤

- 比较两个引用是否指向同一个对象。
- 判断obj是否为null。
- 判断两个引用指向的实际对象类型是否一致。
- 强制类型转换。
- 依次比较各个属性值是否相同。



# finalize()方法

- 当对象被判定为垃圾对象时，由JVM自动调用此方法，用以标记垃圾对象，进入回收队列。
- 垃圾对象：没有有效引用指向此对象时，为垃圾对象。
- 垃圾回收：由GC销毁垃圾对象，释放数据存储空间。
- 自动回收机制：JVM的内存耗尽，一次性回收所有垃圾对象。
- 手动回收机制：使用System.gc(); 通知JVM执行垃圾回收。

# 包 装 类

Java Platform Standard Edition

# 什么是包装类?

- 基本数据类型所对应的引用数据类型。
- Object可统一所有数据，包装类的默认值是null。

# 包装类对应

基本数据类型	包装类型
byte	Byte
short	Short
int	Integer
long	Long
float	Float
double	Double
boolean	Boolean
char	Character

# 类型转换与装箱、拆箱

- 8种包装类提供不同类型间的转换方式：
  - Number父类中提供的6个共性方法。
  - parseXXX()静态方法（除了Character）。
  - valueOf()静态方法。
- 注意：需保证类型兼容，否则抛出NumberFormatException异常。
- JDK 5.0之后，自动装箱、拆箱。基本数据类型和包装类自动转换。

# 整数缓冲区

- Java预先创建了256个常用的整数包装类型对象。
- 在实际应用当中，对已创建的对象进行复用。

# String类

Java Platform Standard Edition

# String

- Java程序中的所有字符串文本（例如“abc”）都是此类的实例。
- 字符串字面值是常量，创建之后不可改变。
- 常用创建方式：
  - `String str1 = "Hello" ;`
  - `String str2 = new String( "World" );`



- `public char charAt(int index)`: 根据下标获取字符。
- `public boolean contains(String str)`: 判断当前字符串中是否包含str。
- `public char[] toCharArray()`: 将字符串转换成数组。
- `public int indexOf(String str)`: 查找str首次出现的下标, 存在, 则返回该下标; 不存在, 则返回-1。
- `public int length()`: 返回字符串的长度。
- `public String trim()`: 去掉字符串前后的空格。
- `public String toUpperCase()`: 将小写转成大写。
- `public boolean endsWith(String str)`: 判断字符串是否以str结尾。
- `public String replace(char oldChar,char newChar)`: 将旧字符串替换成新字符串
- `public String[] split(String str)`: 根据str做拆分。
- `public String substring(int beginIndex,int endIndex)`: 在字符串中截取出一个子字符串

- 概念：可在内存中创建可变的缓冲空间，存储频繁改变的字符串。
- 常用方法：
  - `public StringBuilder append(String str)`
- **StringBuilder**：可变长字符串，JDK5.0提供，运行效率高、线程不安全。
- **StringBuffer**：可变长字符串，JDK1.0提供，运行效率慢、线程安全。

# BigDecimal

- 思考：以下程序输出结果是多少？

```
public class TestBigDecimal {  
    public static void main(String[] args) {  
        double d1=1.0;  
        double d2=0.9;  
        System.out.println(d1-d2);  
    }  
}
```

输出结果：

0.099999999999999998



很多实际应用中需要精确运算，而double是近似值存储，不在符合要求，需要借助BigDecimal。

# BigDecimal

- 位置: java.math包中。
- 作用: 精确计算浮点数。
- 创建方式: `BigDecimal bd=new BigDecimal( "1.0" );`
- 方法:
  - `BigDecimal add(BigDecimal bd)` 加
  - `BigDecimal subtract(BigDecimal bd)` 减
  - `BigDecimal multiply(BigDecimal bd)` 乘
  - `BigDecimal divide(BigDecimal bd)` 除

- 利用BigDecimal可以进行数值计算：

```
public class TestBigDecimal {  
    public static void main(String[] args) {  
  
        BigDecimal bd1 = new BigDecimal("1.0");  
        BigDecimal bd2 = new BigDecimal("0.9");  
  
        BigDecimal result1 = bd1.add(bd2);  
        System.out.println("bd1+bd2="+result1);  
  
        BigDecimal result2 = bd1.subtract(bd2);  
        System.out.println("bd1-bd2="+result2);  
  
        BigDecimal result3 = bd1.multiply(bd2);  
        System.out.println("bd1*bd2="+result3);  
  
        BigDecimal result4 = bd1.divide(bd2);  
        System.out.println("bd1/bd2="+result4);  
    }  
}
```

输出结果：

bd1+bd2=1.9

bd1-bd2=0.1

bd1\*bd2=0.90

执行除法运算时，抛出错误

进行除法运算时，如果不能准确的计算出结果时需要指定保留的位数和取舍方式。

- 除法： `BigDecimal(BigDecimal bd,int scal,RoundingMode mode)`
- 参数 `scale` ： 指定精确到小数点后几位。
- 参数 `mode` ：
  - 指定小数部分的取舍模式，通常采用四舍五入的模式，
  - 取值为 `BigDecimal.ROUND_HALF_UP`。

- Date表示特定的瞬间，精确到毫秒。
- Date类中的大部分方法都已经被Calendar类中的方法所取代。
- 时间单位
  - 1秒=1000毫秒
  - 1毫秒=1000微秒
  - 1微秒=1000纳秒

- **Calendar**提供了获取或设置各种日历字段的方法。
- **protected Calendar()** 构造方法为protected修饰，无法直接创建该对象。
- 其他方法

方法名	说明
static Calendar getInstance()	使用默认时区和区域获取日历
void set(int year,int month,int date,int hourofday,int minute,int second)	设置日历的年、月、日、时、分、秒。
int get(int field)	返回给定日历字段的值。字段比如年、月、日等
void setTime(Date date)	用给定的Date设置此日历的时间。Date-Calendar
Date getTime()	返回一个Date表示此日历的时间。Calendar-Date
void add(int field,int amount)	按照日历的规则，给指定字段添加或减少时间量
long getTimeInMillis()	毫秒为单位返回该日历的时间值



# SimpleDateFormat

- SimpleDateFormat是以与语言环境有关的方式来格式化和解析日期的类。
- 进行格式化（日期 -> 文本）、解析（文本 -> 日期）。
- 常用的时间模式字母

字母	日期或时间	示例
y	年	2019
M	年中月份	08
d	月中天数	10
H	1天中小时数(0-23)	22
m	分钟	16
s	秒	59
S	毫秒	367

- **System系统类**，主要用于获取系统的属性数据和其他操作。

方法名	说明
<code>static void arraycopy(...)</code>	复制数组
<code>static long currentTimeMillis();</code>	获取当前系统时间，返回的是毫秒值
<code>static void gc();</code>	建议JVM赶快启动垃圾回收器回收垃圾
<code>static void exit(int status);</code>	退出jvm，如果参数是0表示正常退出jvm，非0表示异常退出jvm。

- 内部类：
  - 在一个类的内部再定义一个完整的类。包含：成员内部类、静态内部类、局部内部类、匿名内部类。
- Object类：
  - 所有类的直接或间接父类，可存储任何对象。
- 包装类：
  - 基本数据类型所对应的引用数据类型，可以使Object统一所有数据。
- String类：
  - Java程序中的所有字符串文本（例如“abc”）都是此类的实例。字符串字面值是常量，创建之后不可改变。
- BigDecimal：
  - 可精确计算浮点数。
- 时间相关类：Date、Calendar、SimpleDateFormat、System

# THANK YOU



做真实的自己，用良心做教育