常用类

- What?Why?How?

本章概述

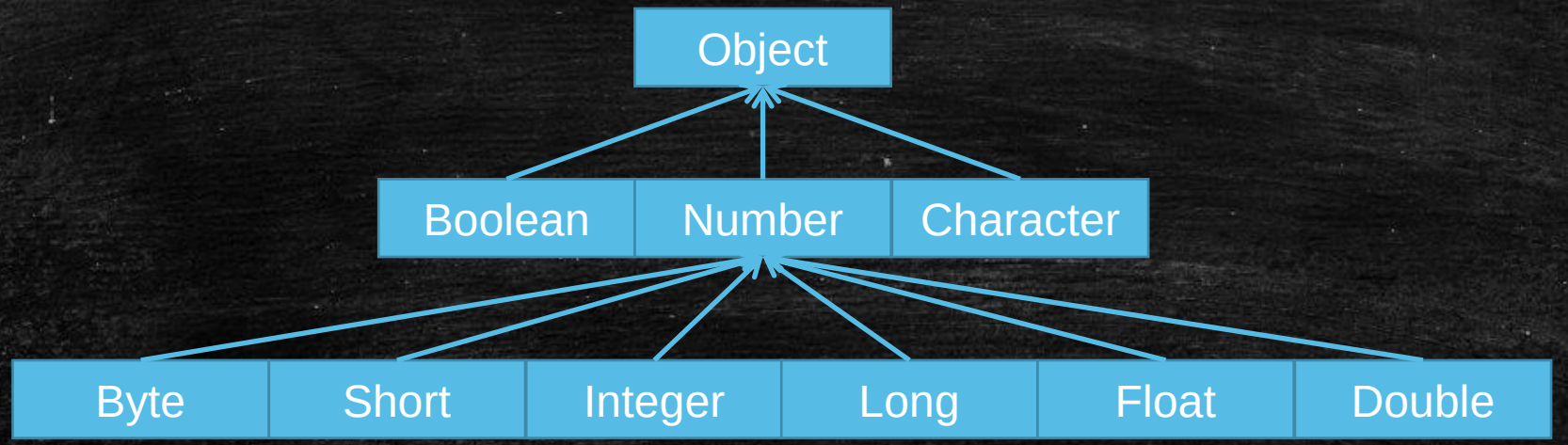
- 基本数据类型的包装类
- 字符串相关类
  - 不可变字符序列：String
  - 可变字符序列：StringBuffer、StringBuilder
- 时间处理相关类
  - Date
  - DateFormat、SimpleDateFormat
  - Calender
- Math类
- File类
- 枚举类：Jdk1.5

包装类

包装类是将基本类型封装到一个类中

包含属性和方法，方便对象操作

包装类位于java.lang包中



包装类和基本类型

- 基本数据类型转换为包装类

```
Integer intValue = new Integer(21);
或Integer intValue = new Integer("21");
Integer intValue = Integer.valueOf("21");
```

- 包装类转换成基本类型

```
Integer integerId=new Integer(25);
int intId=integerId.intValue();
```

- 基本类型和包装类的自动转换

```
Integer intObject = 5;
int intValue = intObject;
```

包装类并不是用来取代基本类型的

自动装箱和自动拆箱 (auto-boxing & unboxing)

- 自动装箱
  - 基本类型就自动地封装到与它相同类型的包装中，如：
  - Integer i = 100;
  - 本质上是，编译器编译时为我们添加了：
  - Integer i = Integer.valueOf(100);
- 自动拆箱
  - 包装类对象自动转换成基本类型数据。如：
  - int a = new Integer(100);
  - 本质上，编译器编译时为我们添加了：
  - int a = new Integer(100).intValue();

1、装箱 与拆箱
装箱: 基本 -->类
new Integer(int)
Integer.valueOf(int i) ;
拆箱: 类 -->基本
intValue()
2、方法
1)、与字符串转换的方法
a)、字符串 -->Integer
Integer(String s)
Integer.parseInt(String s)
Integer.valueOf(String s);
b)、Integer -->字符串
toString()
String.valueOf(Object obj) ;
Integer -->int +""

无处不在的字符串

- 生活中的字符串

频繁使用的字符串

“欢迎进入”

“Hello World”

“教育改变生活”

- 使用String对象存储字符串

```
String s = "Hello World";
```

```
String s = new String();
```

```
String s = new String("Hello World");
```

- String类位于java.lang包中，具有丰富的方法
  - 计算字符串的长度、比较字符串、连接字符串、提取字符串

String(不可变字符序列)

- Java字符串就是**Unicode字符序列**，例如串“Java”就是4个Unicode字符J,a,v,a组成的。
- Java允许使用符号"+"把两个字符串连接起来
  - String s1 = “Hello”;String s2 = “World!”;
  - String s = s1 + s2; //HelloWorld!

String类的常用方法（1）

- char charAt(int index)
  返回字符串中第index个字符。
- boolean equals(String other)
  如果字符串与other相等，返回true
- boolean equalsIgnoreCase(String other)
  如果字符串与other相等（忽略大小写），则返回true
- int indexOf(String str)　**lastIndexOf(String str,int idx)**
- int length()
  返回字符串的长度。
- String replace(char oldChar,char newChar)
  返回一个新串，它是通过用 newChar 替换此字符串中出现的所有oldChar而生成的

String类的常用方法（2）

- boolean startsWith(String prefix)
  如果字符串以prefix开始，则返回true
- boolean endsWith(String prefix)
  如果字符串以prefix结尾，则返回true
- String substring(int beginIndex)
- String substring(int beginIndex,int endIndex)
  返回一个新字符串，该串包含从原始字符串beginIndex到串尾或endIndex-1的所有字符
- String toLowerCase()
  返回一个新字符串，该串将原始字符串中的所有大写字母改成小写字母
- String toUpperCase()
  返回一个新字符串，该串将原始字符串中的所有小写字母改成大写字母
- String trim()
  返回一个新字符串，该串删除了原始字符串头部和尾部的空格

字符串长度

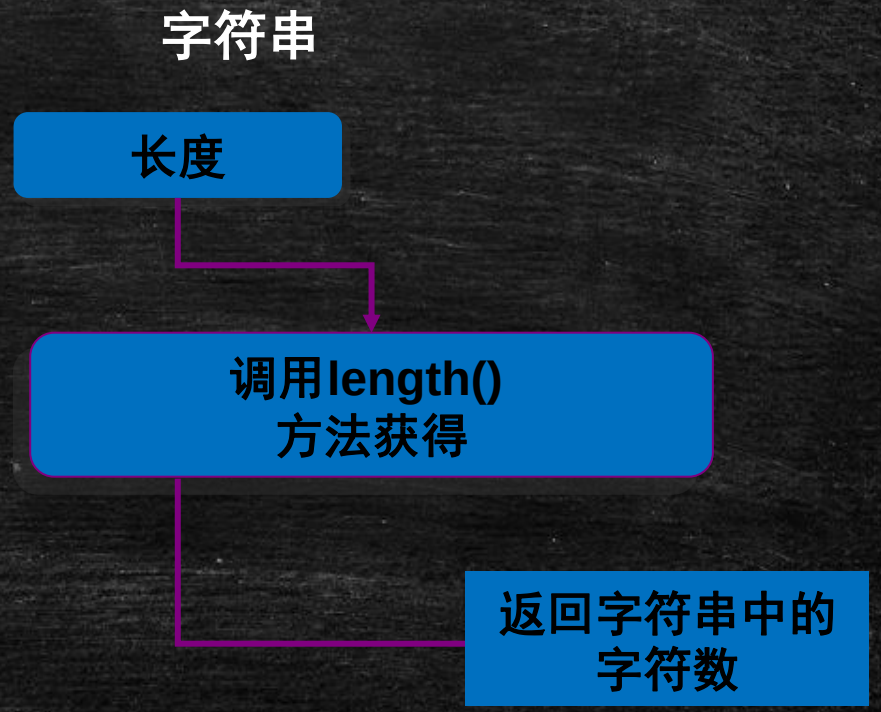
- 计算字符串长度

方法原型:

```
public int length(){
}
```

调用方法:

```
字符串标识符.length();
```

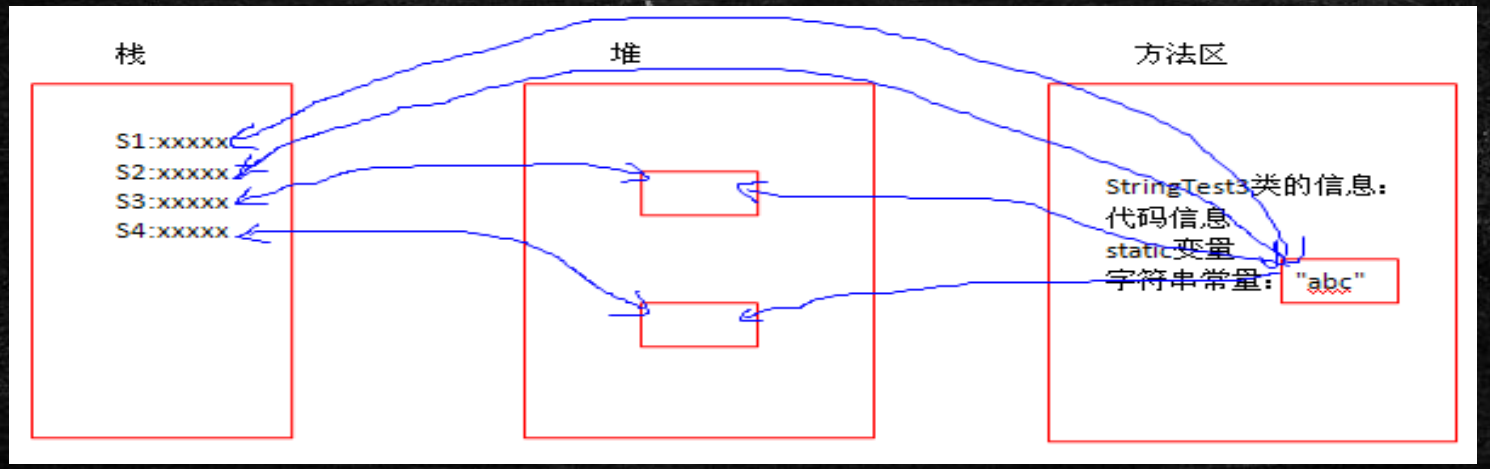


字符串比较

- equals判断字符串值相等，==判断字符串对象引用相等！

```
public class StringTest3 {
        public static void main(String[] args) {
                String s1 = "abc";
                String s2 = "abc";
                String s3 = new String("abc");
                String s4 = new String("abc");
                System.out.println(s1==s2);    //true
                System.out.println(s1==s3);    //false
                System.out.println(s3==s4);    //false
        }
}
```

内存结构图



上机练习1—会员登录

- 训练要点：
  - String类的使用。
  - 带参方法的定义和使用
- 需求说明：
  - 实现会员注册，要求用户名长度不小于3，密码长度不小于6，注册时两次输入密码必须相同
- 实现思路：
  - 1、创建类Register
  - 2、创建验证方法verify()
  - 3、调用方法测试程序
- 难点指导：
  - 创建验证方法verify()

```
***欢迎进入注册系统***

请输入用户名： t
请输入密码： tomcat
请再次输入密码： tomcat
用户名长度不能小于3，密码长度不能小于6!
请输入用户名： tom
请输入密码： tomcat123
请再次输入密码： tomcat12
两次输入的密码不相同!
请输入用户名： tom
请输入密码： tomcat123
请再次输入密码： tomcat123
注册成功！请牢记用户名和密码。
```

字符串连接

- 方法1：使用“+”
- 方法2：使用String类的concat()方法

```
String s = new String("你好，");
String name = new String("张三！");
String sentence = s.concat(name);
System.out.println(sentence);
```

A.concat(B)：
B字符串将被连接到A字符串后面

你好，张三！

字符串常用提取方法

返回出现第一个匹配的位置， 如果没有找到字符或字符串，则返回-1

- 常用提取方法举例

方　法	说　明
public int indexOf(int ch)	搜索第一个出现的字符ch（或字符串value）
public int indexOf(String value)	
public int lastIndexOf(int ch)	搜索最后一个出现的字符ch（或字符串value）
public int lastIndexOf(String value)	

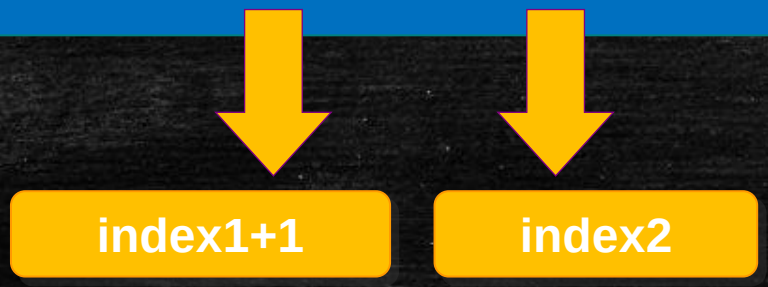
字符串常用提取方法

方 法	说 明
public String substring(int index)	提取从位置索引开始的字符串部分
public String substring(int beginindex, int endindex)	提取beginindex和endindex之间的字符串部分
public String trim()	返回一个前后不含任何空格的调用字符串的副本

小结

如果要打印输出“小鱼儿”，应填入的代码是什么？

```
String word = "Hello,      ";
word = word.trim();
String s = word.concat("小鱼儿!");
int index1 = s.indexOf(',');
int index2 = s.indexOf('!');
System.out.println(s.substring(_______, _________));
```



字符串拆分

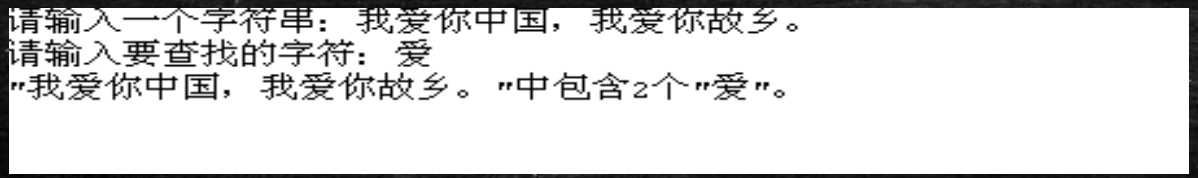
- 有一段歌词，每句都以空格“ ”结尾，请将歌词每句按行输出
- String类提供了split()方法，将一个字符串分割为子字符串，结果作为字符串数组返回

```
***原歌词格式***
长亭外 古道边 芳草碧连天 晚风拂 柳笛声残 夕阳山外山

***拆分后歌词格式***
长亭外
古道边
芳草碧连天
晚风拂
柳笛声残
夕阳山外山
```

上机练习2--判断字符出现次数

- 需求说明：
  - 输入一个字符串，输入一个字符，判断该字符在该字符串中出现的次数



StringBuffer类与StringBuilder类

- StringBuffer：String增强版
- 字符串缓冲区，是一个容器
- StringBuffer声明

```
StringBuffer sb = new StringBuffer();
StringBuffer sb = new StringBuffer("aaa");
```

创建空StringBuffer对象

创建一个变量存储字符串aaa

```
sb.toString();      //转化为String类型
```

- StringBuffer的使用

```
sb.append("**");  //追加字符串
```

StringBuffer类

```
public class sbAppend {
    public static void main(String[ ] args) {
        StringBuffer sb = new StringBuffer("青春无悔");
        int num=110;
        StringBuffer sb1 = sb.append("我心永恒");
        System.out.println(sb1);
        StringBuffer sb2 = sb1.append('啊');
        System.out.println(sb2);
        StringBuffer sb3 = sb2.append(num);
        System.out.println(sb3);
    }
}
```

相当于sb3.toString()

```
青春无悔我心永恒
青春无悔我心永恒啊
青春无悔我心永恒啊110
```

StringBuffer类

- 利用StringBuffer类的length()和insert ()方法实现需求
- 将一个数字字符串转换成逗号分隔的数字串，即从右边开始每三个数字用逗号分隔

```
请输入一串数字：  12345678
12,345,678
```

StringBuffer类

```
public class TestInsert {
	public static void main(String[] args) {
		Scanner input = new Scanner(System.in);
		System.out.print("请输入一串数字： ");
		String nums = input.next();
		StringBuffer str=new StringBuffer(nums);
		for(int i=str.length()-3;i>0;i=i-3){
			str.insert(i,',');
		}
		System.out.print(str);
	}
}
```

上机练习3--实现商品批发总金额显示

- 训练要点：
  - StringBuffer类的使用
  - 方法的定义和使用
- 需求说明：
  - 登录验证通过后，显示批发商品信息;输入批发商品编号和数量，以指定格式显示总金额
- 实现思路：
  - 1、创建类Goods
  - 2、创建方法show()
  - 3、创建方法change()
- 难点指导：
  - 格式化输出

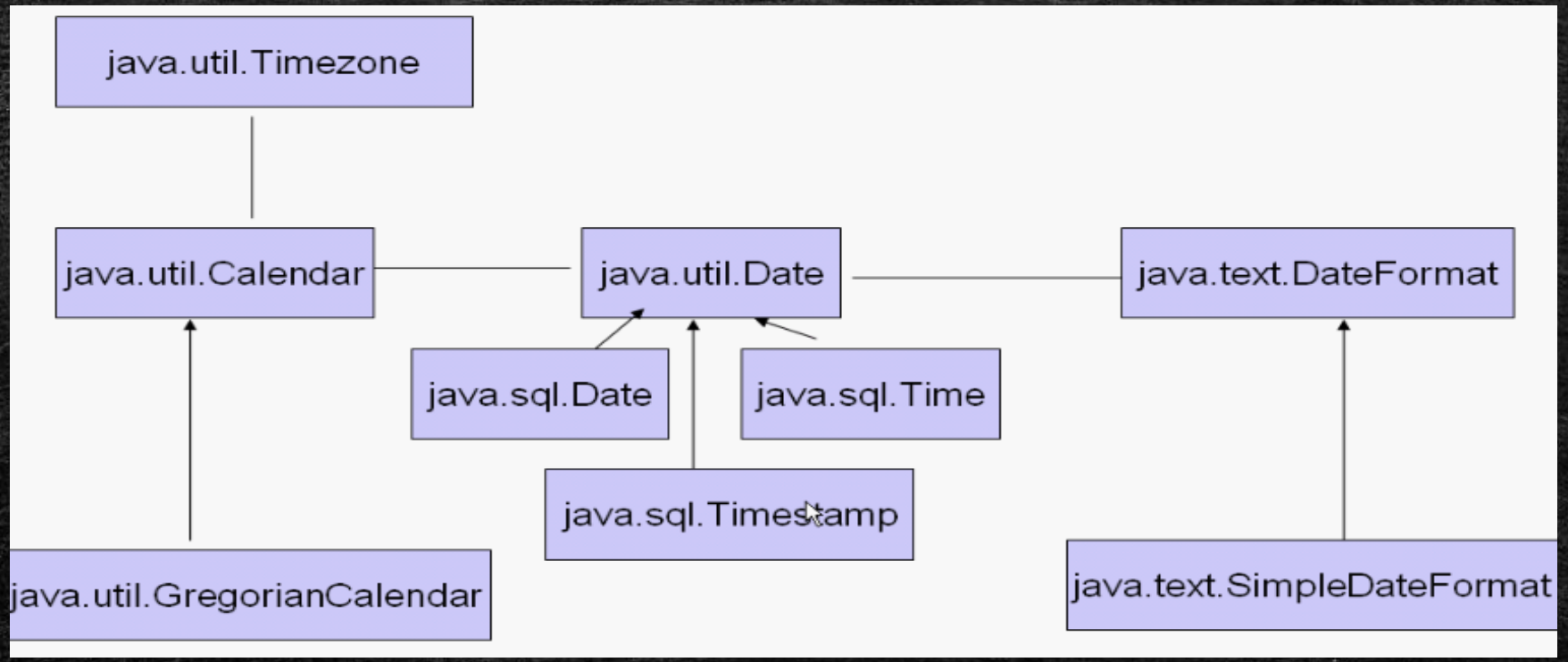
```
请输入用户名： TOM
请输入密码： 123
登录成功！
*********欢迎进入商品批发城*********
        编号      商品      价格
        1         电风扇    124.23
        2         洗衣机    4,500.0
        3         电视机    8,800.9
        4         冰箱      5,000.88
        5         空调机    4,456.0
***********************************
请输入您批发的商品编号： 3
请输入批发数量： 15
您需要付款： 132,013.5
```

字符串选用 A

- String：不可变字符序列
- StringBuilder：可变字符序列、效率高、线程不安全
- StringBuffer：可变字符序列、效率低、线程安全
- String使用陷阱：
  - string s="a"; //创建了一个字符串

    s=s+"b"; //实际上原来的"a"字符串对象已经丢弃了，现在又产生了一个字符串s+"b"。如果多次执行这些改变串内容的操作，会导致大量副本字符串对象存留在内存中，降低效率。如果这样的操作放到循环中，会极大影响程序的性能。

时间处理相关类


java.util.Timezone
java.util.Calendar
java.util.Date
java.text.DateFormat
java.sql.Date
java.sql.Time
java.sql.Timestamp
java.util.GregorianCalendar
java.text.SimpleDateFormat

Date时间类(java.util.Date)

Date类：表示日期和时间

提供操作日期和时间各组成部分的方法

DateFormat类 与SimpleDateFormat类

用于定制日期时间的格式

```
Date date = new Date();    //创建日期对象
SimpleDateFormat formater = new SimpleDateFormat("yyyy-
    MM-dd HH:mm:ss");//定制日期格式
String now = formater.format(date);
System.out.println(now);
```

Calendar

Calendar类：

抽象类

用于设置和获取日期/时间数据的特定部分

Calendar类提供一些方法和静态字段来操作日历

方法或属性	说明
int get(int field)	返回给定日历字段的值
MONTH	指示月
DAY_OF_MONTH	指示一个月中的某天
DAY_OF_WEEK	指示一个星期中的某天

Math类

- 包含了常见的数学运算函数。
- random()→生成[0,1)之间的随机浮点数
- 生成：0-10之间的任意整数：
  - int a = (int)(10*Math.random());
- 生成：20-30之间的任意整数：
  - int b = 20 + (int)(10*Math.random());

枚举

- 枚举指由一组固定的常量组成的类型

```
[Modifier] enum enumName{
    enumContantName1[,
    enumConstantName2...[;]]
    //[field，method]
}
```

使用枚举的好处

- 类型安全
- 易于输入
- 代码清晰

性别枚举

```
public enum Genders{
    Male,Female
```

```
public class Student{
    public  Genders  sex;
```

枚举类型的变量

```
Student stu=new Student();
stu.sex=Genders.Male;
```

```
stu.sex="你好";
```

枚举

- 枚举类型：
  1. 只能够取特定值中的一个
  2. 使用enum关键字
  3. 所有的枚举类型隐性地继承自 java.lang.Enum。（枚举实质上还是类！**而每个被枚举的成员实质就是一个枚举类型的实例**，他们默认都是public static final的。可以直接通过枚举类型名直接使用它们。）
  4. 强烈建议当你需要定义**一组常量时**，使用枚举类型

上机练习6—枚举的使用

需求说明

为JavaSE课程三个单元定义枚举：U1，U2，U3分别表示初级，中级，高级

编程输出每个单元的学习目标

总结

- **字符串**
  - String :字符串处理浪费内存
  - StringBuffer :线程安全
  - StringBuilder :线程不安全
- **日期与时间**
  - Date
  - DateFormat
  - SimpleDateFormat :格式化日期
  - Calendar
- **攻举类型**
  - Enum
- **数学**
  - Math
  - Random