第 12 章异常-Exception

12. 1 看个实际的问题和一段代码

运行下面的代码，看看有什么问题-> 引出异常和异常处理机制 Exception01.java

|  |
| --- |
| public static void main(String[] args) {  int num1 = 10;  int num2 = 0;  int res = num1 / num2;  System.out.println("程序继续运行....");  } |

12.2 解决方案-异常捕获

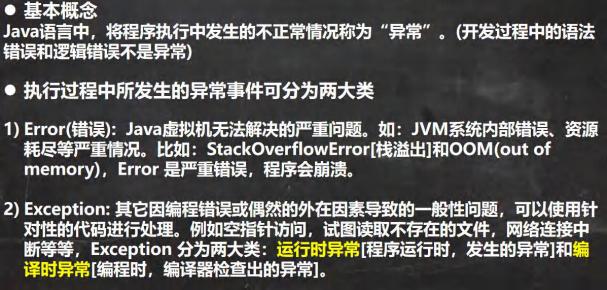
对异常进行捕获，保证程序可以继续运行.

看老师的代码演示 try-catch

|  |
| --- |
| package com.hspedu.exception\_;  /\*\*  \* @author 韩顺平  \* @version 1.0  \*/  public class Exception01 {  public static void main(String[] args) {  int num1 = 10;  int num2 = 0;//Scanner();  //老韩解读 |

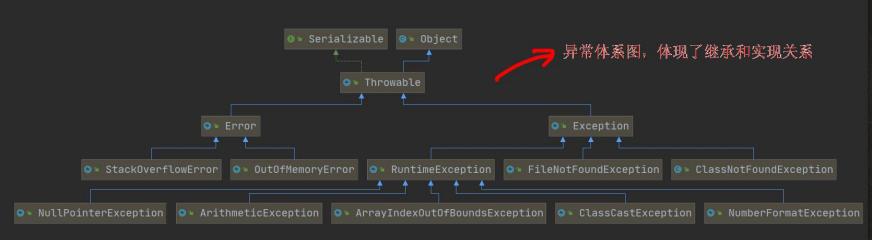
|  |
| --- |
| //1. num1 / num2 => 10 / 0  //2. 当执行到 num1 / num2 因为 num2 = 0, 程序就会出现(抛出)异常 ArithmeticException  //3. 当抛出异常后，程序就退出，崩溃了 , 下面的代码就不在执行  //4. 大家想想这样的程序好吗? 不好，不应该出现了一个不算致命的问题，就导致整个系统崩溃  //5. java 设计者，提供了一个叫 异常处理机制来解决该问题  // int res = num1 / num2;  //如果程序员，认为一段代码可能出现异常/问题，可以使用 try-catch 异常处理机制来解决  //从而保证程序的健壮性  //将该代码块->选中->快捷键 ctrl + alt + t -> 选中 try-catch  //6. 如果进行异常处理，那么即使出现了异常，程序可以继续执行  try {  int res = num1 / num2;  } catch (Exception e) {  //e.printStackTrace();  System.out.println(" 出现异常的原因=" + e.getMessage());//输出异常信息  }  System.out.println("程序继续运行....");  }  } |

12.3 异常介绍

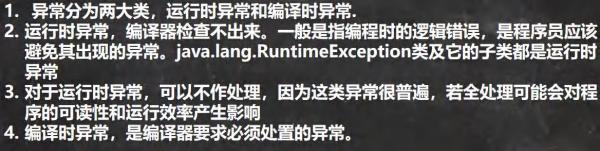


12.4 异常体系图一览

12.4.1 异常体系图



12.4.2 异常体系图的小结



12.5 常见的运行时异常

12.5.1 常见的运行时异常包括

1) NullPointerException 空指针异常

2) ArithmeticException 数学运算异常

3) ArrayIndexOutOfBoundsException 数组下标越界异常

4) ClassCastException 类型转换异常

5) NumberFormatException 数字格式不正确异常[]

12.5.2 常见的运行时异常举例

1) NullPointerException 空指针异常 NullPointerException\_.java

当应用程序试图在需要对象的地方使用 null 时，抛出该异常,看案例演示。



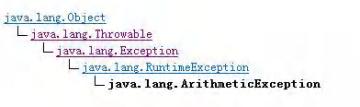
代码:

|  |
| --- |
| package com.hspedu.exception\_;  /\*\*  \* @author 韩顺平 |

|  |
| --- |
| \* @version 1.0  \*/  public class NullPointerException\_ {  public static void main(String[] args) {  String name = null;  System.out.println(name.length());  }  } |

2) ArithmeticException 数学运算异常 ArithmeticException\_.java

当出现异常的运算条件时，抛出此异常。例如，一个整数“除以零 ”时，抛出此类的一个实例, 案例演示



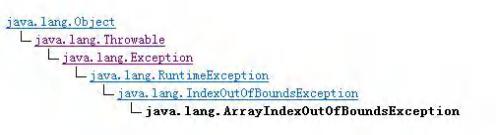
代码:

|  |
| --- |
| package com.hspedu.exception\_;  /\*\*  \* @author 韩顺平  \* @version 1.0  \*/  public class NumberFormatException\_ {  public static void main(String[] args) {  String name = "韩顺平教育";  //将 String 转成 int |

|  |
| --- |
| int num = Integer.parseInt(name);//抛出 NumberFormatException  System.out.println(num);//1234  }  } |

3) ArrayIndexOutOfBoundsException 数组下标越界异常

用非法索引访问数组时抛出的异常。如果索引为负或大于等于数组大小，则该索引为非法索引



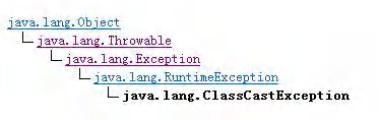
代码

|  |
| --- |
| package com.hspedu.exception\_;  /\*\*  \* @author 韩顺平  \* @version 1.0  \*/  public class ArrayIndexOutOfBoundsException\_ {  public static void main(String[] args) {  int[] arr = {1,2,4};  for (int i = 0; i <= arr.length; i++) {  System.out.println(arr[i]);  } |

|  |
| --- |
| }  } |

4) ClassCastException 类型转换异常

当试图将对象强制转换为不是实例的子类时，抛出该异常。例如，以下代码将生成一个 ClassCastException



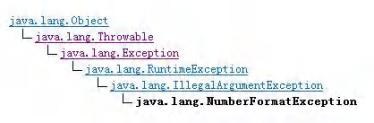
代码:

|  |
| --- |
| package com.hspedu.exception\_;  /\*\*  \* @author 韩顺平  \* @version 1.0  \*/  public class ClassCastException\_ {  public static void main(String[] args) {  Ab = new B(); //向上转型  B b2 = (B)b;//向下转型，这里是 OK  C c2 = (C)b;//这里抛出 ClassCastException  }  }  class A {}  class B extends A {}  class C extends A {} |

5) NumberFormatException 数字格式不正确异常

当应用程序试图将字符串转换成一种数值类型，但该字符串不能转换为适当格式时，抛出该异常 => 使用异常我们

可以确保输入是满足条件数字.



代码:

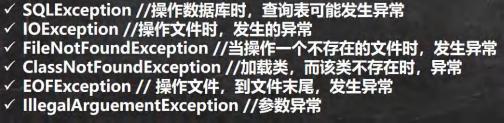
|  |
| --- |
| package com.hspedu.exception\_;  /\*\*  \* @author 韩顺平  \* @version 1.0  \*/  public class NumberFormatException\_ {  public static void main(String[] args) {  String name = "韩顺平教育";  //将 String 转成 int  int num = Integer.parseInt(name);//抛出 NumberFormatException  System.out.println(num);//1234  }  } |

12.6 编译异常

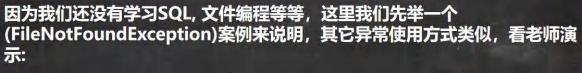
12.6.1 介绍



12.6.2 常见的编译异常



12.6.3 案例说明



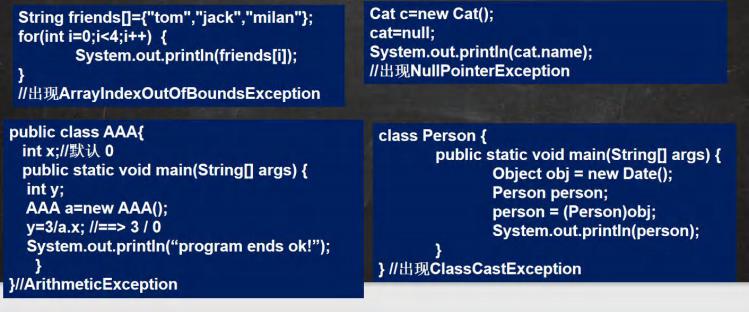
代码

|  |
| --- |
| package com.hspedu.exception\_;  import java.io.FileInputStream;  import java.io.IOException;  /\*\*  \* @author 韩顺平  \* @version 1.0  \*/  public class Exception02 {  public static void main(String[] args) {  try { |

|  |
| --- |
| FileInputStream fis;  fis = new FileInputStream("d:\\aa.jpg");  int len;  while ((len = fis.read()) != - 1) {  System.out.println(len);  }  fis.close();  } catch (IOException e) {  e.printStackTrace();  }  }  } |

12.7 异常课堂练习

看看下面代码是否正确，为什么？ 课堂练习 4 个题 4min

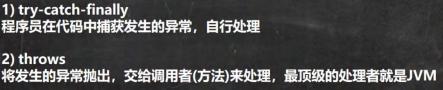


12.8 异常处理

12.8.1 基本介绍

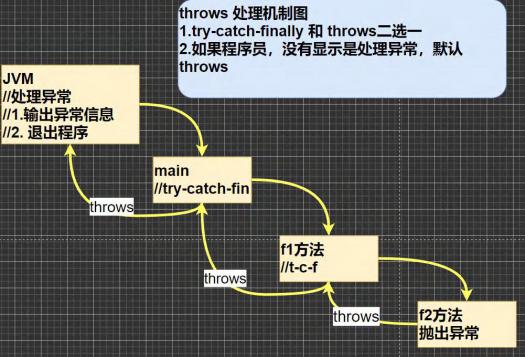


12.8.2 异常处理的方式



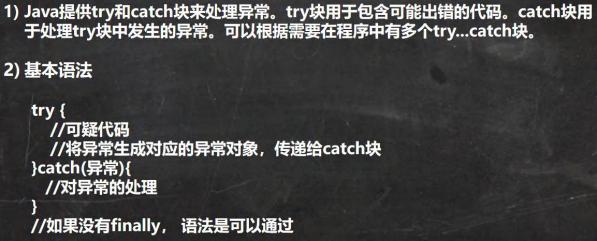
12.8.3 示意图





12.9 try-catch 异常处理

12.9.1 try-catch 方式处理异常说明 TryCatch01.java



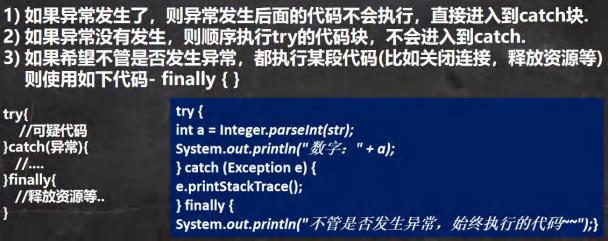
12.9.2 try-catch 方式处理异常-快速入门

代码:

|  |
| --- |
| public static void main(String[] args) { |

|  |
| --- |
| int num1 = 10;  int num2 = 0;  try {  int res = num1 / num2;  } catch (Exception e) {  System.out.println(e.getMessage());  }  } |

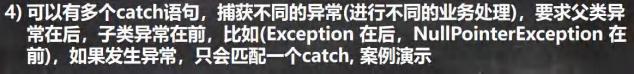
12.9.3 try-catch 方式处理异常-注意事项 TryCatchDetail.java



代码

|  |
| --- |
| package com.hspedu.try\_;  /\*\*  \* @author 韩顺平  \* @version 1.0  \*/ |

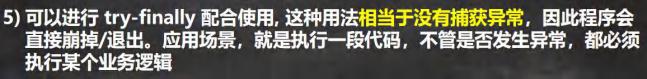
|  |
| --- |
| public class TryCatchDetail {  public static void main(String[] args) {  //ctrl + atl + t  //老韩解读  //1. 如果异常发生了，则异常发生后面的代码不会执行，直接进入到catch 块  //2. 如果异常没有发生，则顺序执行try 的代码块，不会进入到 catch  //3. 如果希望不管是否发生异常，都执行某段代码(比如关闭连接，释放资源等)则使用如下代码- finally  try {  String str = "韩顺平";  int a = Integer.parseInt(str);  System.out.println("数字：" + a);  } catch (NumberFormatException e) {  System.out.println("异常信息=" + e.getMessage());  } finally {  System.out.println("finally 代码块被执行...");  }  System.out.println("程序继续...");  }  } |



代码

|  |
| --- |
| package com.hspedu.try\_;  /\*\*  \* @author 韩顺平  \* @version 1.0  \*/  public class TryCatchDetail02 {  public static void main(String[] args) {  //老韩解读  //1.如果try 代码块有可能有多个异常  //2.可以使用多个catch 分别捕获不同的异常，相应处理  //3.要求子类异常写在前面，父类异常写在后面  try {  Person person = new Person();  //person = null;  System.out.println(person.getName());//NullPointerException  int n1 = 10;  int n2 = 0;  int res = n1 / n2;//ArithmeticException  } catch (NullPointerException e) {  System.out.println("空指针异常=" + e.getMessage()); |

|  |
| --- |
| } catch (ArithmeticException e) {  System.out.println("算术异常=" + e.getMessage());  } catch (Exception e) {  System.out.println(e.getMessage());  } finally {  }  }  }  class Person {  private String name = "jack";  public String getName() {  return name;  }  } |



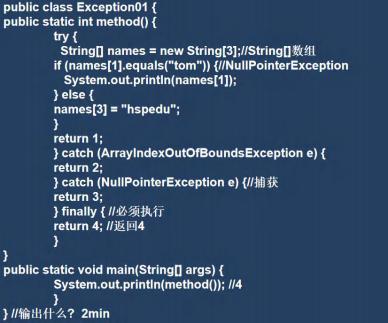
代码:

|  |
| --- |
| package com.hspedu.try\_; |

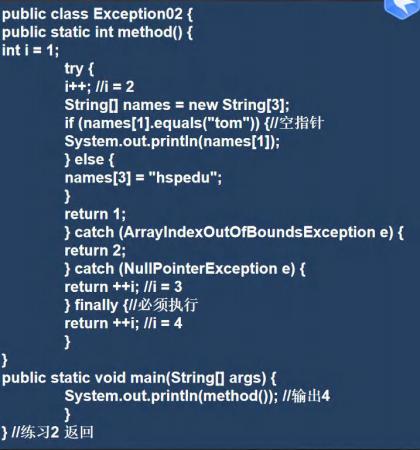
|  |
| --- |
| /\*\*  \* @author 韩顺平  \* @version 1.0  \*/  public class TryCatchDetail03 {  public static void main(String[] args) {  /\*  可以进行 try-finally 配合使用, 这种用法相当于没有捕获异常，  因此程序会直接崩掉/退出。应用场景，就是执行一段代码，不管是否发生异常，  都必须执行某个业务逻辑  \*/  try{  int n1 = 10;  int n2 = 0;  System.out.println(n1 / n2);  }finally {  System.out.println("执行了 finally..");  }  System.out.println("程序继续执行..");  }  } |

12.9.4 异常处理课堂练习

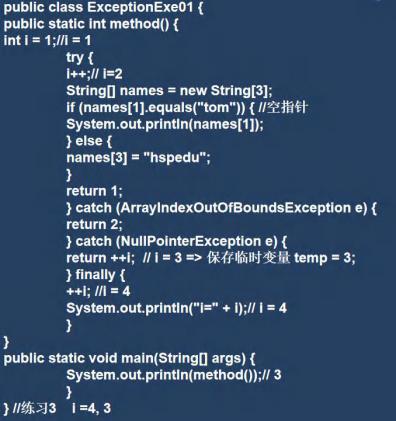
1) 题 1 TryCatchExercise01.java



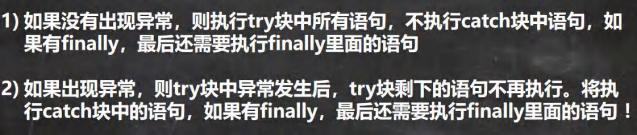
2) 题 2TryCatchExercise02.java



3) 题 3TryCatchExercise03.java



12.9.5 try-catch-finally 执行顺序小结



12.9.6 课后练习题: TryCatchExercise04.java

如果用户输入的不是一个整数，就提示他反复输入，直到输入一个整数为止

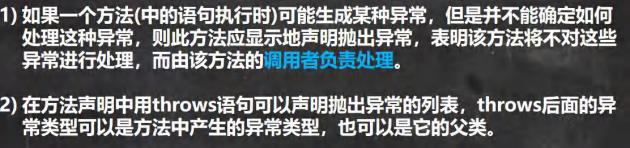
|  |
| --- |
| package com.hspedu.try\_;  import java.util.Scanner; |

|  |
| --- |
| /\*\*  \* @author 韩顺平  \* @version 1.0  \*/  public class TryCatchExercise04 {  public static void main(String[] args) {  //如果用户输入的不是一个整数，就提示他反复输入，直到输入一个整数为止  //思路  //1. 创建 Scanner 对象  //2. 使用无限循环，去接收一个输入  //3. 然后将该输入的值，转成一个 int  //4. 如果在转换时，抛出异常，说明输入的内容不是一个可以转成 int 的内容  //5. 如果没有抛出异常，则 break 该循环  Scanner scanner = new Scanner(System.in);  int num = 0;  String inputStr = "";  while (true) {  System.out.println("请输入一个整数:"); //  inputStr = scanner.next();  try {  num = Integer.parseInt(inputStr); //这里是可能抛出异常  break;  } catch (NumberFormatException e) { |

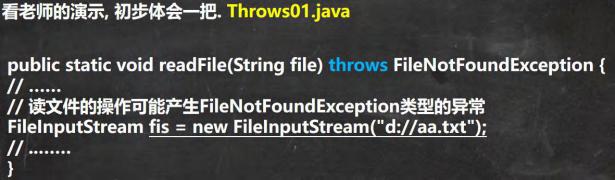
|  |
| --- |
| System.out.println("你输入的不是一个整数:");  }  }  System.out.println("你输入的值是=" + num);  }  } |

12. 10throws 异常处理

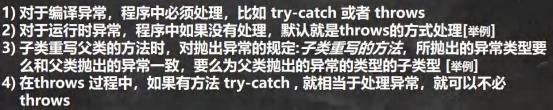
12.10. 1 基本介绍



12.10.2 快速入门案例



12.10.3 注意事项和使用细节 ThrowsDetail.java



代码:

|  |
| --- |
| package com.hspedu.throws\_;  import java.io.FileInputStream;  import java.io.FileNotFoundException;  /\*\*  \* @author 韩顺平  \* @version 1.0  \*/  public class ThrowsDetail {  public static void main(String[] args) {  f2();  }  public static void f2() /\*throws ArithmeticException\*/ {  //1.对于编译异常，程序中必须处理，比如 try-catch 或者 throws  //2.对于运行时异常，程序中如果没有处理，默认就是 throws 的方式处理  int n1 = 10;  int n2 = 0;  double res = n1 / n2; |

|  |
| --- |
| }  public static void f1() throws FileNotFoundException {  //这里大家思考问题 调用 f3() 报错  //老韩解读  //1. 因为 f3() 方法抛出的是一个编译异常  //2. 即这时，就要 f1() 必须处理这个编译异常  //3. 在 f1() 中，要么 try-catch-finally ,或者继续 throws 这个编译异常  f3(); // 抛出异常  }  public static void f3() throws FileNotFoundException {  FileInputStream fis = new FileInputStream("d://aa.txt");  }  public static void f4() {  //老韩解读:  //1. 在 f4()中调用方法 f5() 是 OK  //2. 原因是 f5() 抛出的是运行异常  //3. 而java 中，并不要求程序员显示处理, 因为有默认处理机制  f5();  }  public static void f5() throws ArithmeticException {  }  } |

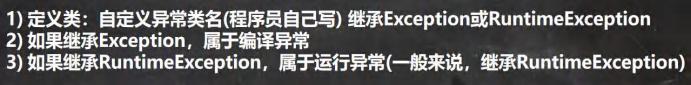
|  |
| --- |
| class Father { //父类  public void method() throws RuntimeException {  }  }  class Son extends Father {//子类  //3. 子类重写父类的方法时，对抛出异常的规定:子类重写的方法，  // 所抛出的异常类型要么和父类抛出的异常一致，要么为父类抛出的异常类型的子类型  //4. 在 throws 过程中，如果有方法 try-catch , 就相当于处理异常，就可以不必 throws  @Override  public void method() throws ArithmeticException {  }  } |

12. 11自定义异常

12.11. 1 基本概念



12.11.2 自定义异常的步骤



12.11.3 自定义异常的应用实例 CustomException.java



代码:

|  |
| --- |
| package com.hspedu.customexception\_;  /\*\*  \* @author 韩顺平  \* @version 1.0  \*/  public class CustomException {  public static void main(String[] args) /\*throws AgeException\*/ {  int age = 180;  //要求范围在 18 – 120 之间，否则抛出一个自定义异常  if(!(age >= 18 && age <= 120)) {  //这里我们可以通过构造器，设置信息  throw new AgeException("年龄需要在 18~ 120 之间");  }  System.out.println("你的年龄范围正确.");  }  }  //自定义一个异常  //老韩解读  //1. 一般情况下，我们自定义异常是继承 RuntimeException  //2. 即把自定义异常做成 运行时异常，好处时，我们可以使用默认的处理机制  //3. 即比较方便 |

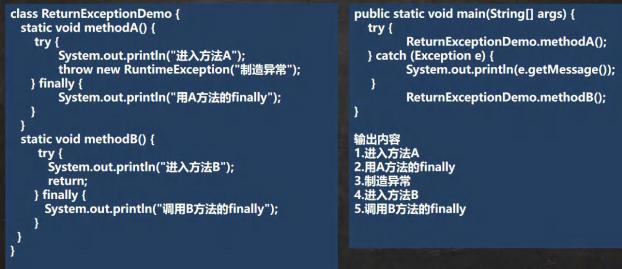
|  |
| --- |
| class AgeException extends RuntimeException {  public AgeException(String message) {//构造器  super(message);  }  } |

12. 12throw 和 throws 的区别

12.12.1 一览表

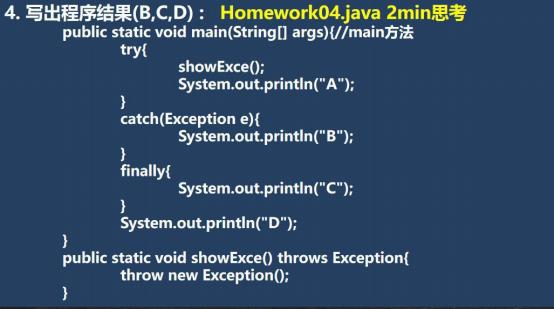


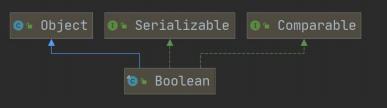
12.12.2 测试题-下面的测试输出什么 ThrowException.java 2min



12. 13本章作业







第 13 章常用类

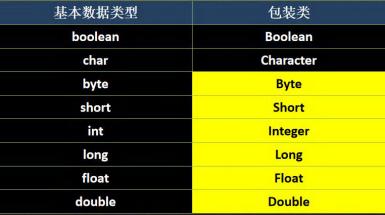
13. 1 包装类

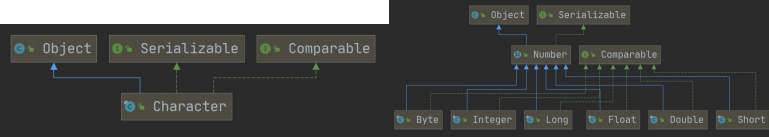
13.1.1 包装类的分类 WrapperType.java

1) 针对八种基本数据类型相应的引用类型—包装类

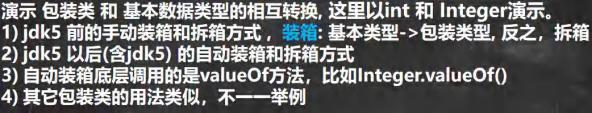
2) 有了类的特点，就可以调用类中的方法。

3) 如图:





13.1.2 包装类和基本数据的转换



13.1.3 案例演示 Integer01.java

|  |
| --- |
| package com.hspedu.wrapper;  /\*\*  \* @author 韩顺平  \* @version 1.0  \*/  public class Integer01 {  public static void main(String[] args) {  //演示 int <--> Integer 的装箱和拆箱  //jdk5 前是手动装箱和拆箱  //手动装箱 int->Integer  int n1 = 100;  Integer integer = new Integer(n1);  Integer integer1 = Integer.valueOf(n1);  //手动拆箱  //Integer -> int  int i = integer.intValue();  //jdk5 后，就可以自动装箱和自动拆箱  int n2 = 200;  //自动装箱 int->Integer  Integer integer2 = n2; //底层使用的是 Integer.valueOf(n2)  //自动拆箱 Integer->int  int n3 = integer2; //底层仍然使用的是 intValue()方法  } |

|  |
| --- |
| } |

13.1.4 课堂测试题 WrapperExercise01.java 2min



13.1.5 包装类型和 String 类型的相互转换 WrapperVSString.java

案例演示, 以 Integer 和 String 转换为例，其它类似:

|  |
| --- |
| package com.hspedu.wrapper;  /\*\*  \* @author 韩顺平  \* @version 1.0  \*/  public class WrapperVSString {  public static void main(String[] args) {  //包装类(Integer)->String  Integer i = 100;//自动装箱 |

|  |
| --- |
| //方式 1  String str1 = i + "";  //方式 2  String str2 = i.toString();  //方式 3  String str3 = String.valueOf(i);  //String -> 包装类(Integer)  String str4 = " 12345";  Integer i2 = Integer.parseInt(str4);//使用到自动装箱  Integer i3 = new Integer(str4);//构造器  System.out.println("ok~~");  }  } |

13.1.6 Integer 类和 Character 类的常用方法



|  |
| --- |
| package com.hspedu.wrapper;  /\*\*  \* @author 韩顺平  \* @version 1.0  \*/ |

|  |
| --- |
| public class WrapperMethod {  public static void main(String[] args) {  System.out.println(Integer.MIN\_VALUE); //返回最小值  System.out.println(Integer.MAX\_VALUE);//返回最大值  System.out.println(Character.isDigit('a'));//判断是不是数字  System.out.println(Character.isLetter('a'));//判断是不是字母  System.out.println(Character.isUpperCase('a'));//判断是不是大写  System.out.println(Character.isLowerCase('a'));//判断是不是小写  System.out.println(Character.isWhitespace('a'));//判断是不是空格  System.out.println(Character.toUpperCase('a'));//转成大写  System.out.println(Character.toLowerCase('A'));//转成小写  }  } |

13.1.7 Integer 类面试题 1 WrapperExercise02.java

看看下面代码，输出什么结果? 为什么? 2min

|  |
| --- |
| package com.hspedu.wrapper;  /\*\*  \* @author 韩顺平  \* @version 1.0  \*/  public class WrapperExercise02 { |

|  |
| --- |
| public static void main(String[] args) {  Integer i = new Integer(1);  Integer j = new Integer(1);  System.out.println(i == j); //False  //所以，这里主要是看范围 - 128 ~ 127 就是直接返回  /\*  老韩解读  //1. 如果 i 在 IntegerCache.low(- 128)~IntegerCache.high(127),就直接从数组返回  //2. 如果不在 - 128~ 127,就直接 new Integer(i)  public static Integer valueOf(int i) {  if (i >= IntegerCache.low && i <= IntegerCache.high)  return IntegerCache.cache[i + (-IntegerCache.low)];  return new Integer(i);  }  \*/  Integer m = 1; //底层 Integer.valueOf(1); -> 阅读源码  Integer n = 1;//底层 Integer.valueOf(1);  System.out.println(m == n); //T  //所以，这里主要是看范围 - 128 ~ 127 就是直接返回  // ，否则，就 new Integer(xx);  Integer x = 128;//底层 Integer.valueOf(1);  Integer y = 128;//底层 Integer.valueOf(1);  System.out.println(x == y);//False  }  } |

13.1.8 Intege 类面试题总结

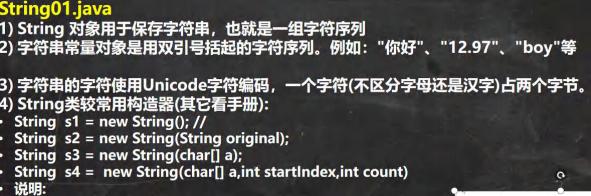
看看下面代码，输出什么结果.WrapperExercise03.java 5min

|  |
| --- |
| package com.hspedu.wrapper;  /\*\*  \* @author 韩顺平  \* @version 1.0  \*/  public class WrapperExercise03 {  public static void main(String[] args) {  //示例一  Integer i1 = new Integer(127);  Integer i2 = new Integer(127);  System.out.println(i1 == i2);//F  //示例二  Integer i3 = new Integer(128);  Integer i4 = new Integer(128);  System.out.println(i3 == i4);//F  //示例三  Integer i5 = 127;//底层 Integer.valueOf(127)  Integer i6 = 127;//- 128~ 127  System.out.println(i5 == i6); //T  //示例四  Integer i7 = 128; |

|  |
| --- |
| Integer i8 = 128;  System.out.println(i7 == i8);//F  //示例五  Integer i9 = 127; //Integer.valueOf(127)  Integer i10 = new Integer(127);  System.out.println(i9 == i10);//F  //示例六  Integer i11= 127;  int i12= 127;  //只有有基本数据类型，判断的是  //值是否相同  System.out.println(i11==i12); //T  //示例七  Integer i13= 128;  int i14= 128;  System.out.println(i13==i14);//T  }  } |

13.2 String 类

13.2.1 String 类的理解和创建对象





代码:

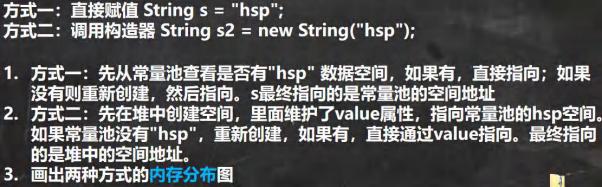
|  |
| --- |
| package com.hspedu.string\_;  /\*\*  \* @author 韩顺平  \* @version 1.0  \*/  public class String01 {  public static void main(String[] args) {  //1.String 对象用于保存字符串，也就是一组字符序列  //2. "jack" 字符串常量, 双引号括起的字符序列  //3. 字符串的字符使用 Unicode 字符编码，一个字符(不区分字母还是汉字)占两个字节  //4. String 类有很多构造器，构造器的重载 |

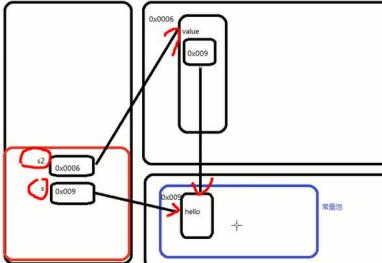
|  |
| --- |
| // 常用的有 String s1 = new String(); //  //String s2 = new String(String original);  //String s3 = new String(char[] a);  //String s4 = new String(char[] a,int startIndex,int count)  //String s5 = new String(byte[] b)  //5. String 类实现了接口 Serializable【String 可以串行化:可以在网络传输】  // 接口 Comparable [String 对象可以比较大小]  //6. String 是 final 类，不能被其他的类继承  //7. String 有属性 private final char value[]; 用于存放字符串内容  //8. 一定要注意：value 是一个 final 类型， 不可以修改(需要功力)：即 value 不能指向  // 新的地址，但是单个字符内容是可以变化  String name = "jack";  name = "tom";  final char[] value = {'a','b','c'};  char[] v2 = {'t','o','m'};  value[0] = 'H';  //value = v2; 不可以修改 value 地址  }  } |

13.2.2 创建 String 对象的两种方式

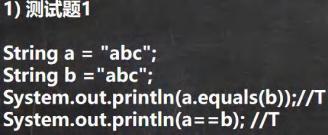


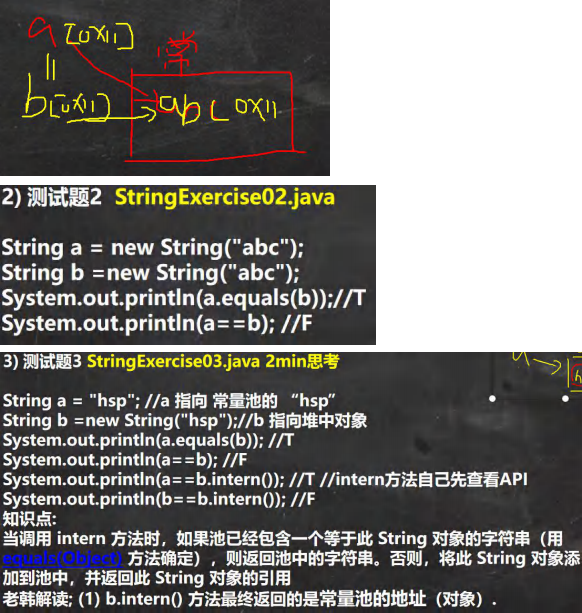
13.2.3 两种创建 String 对象的区别



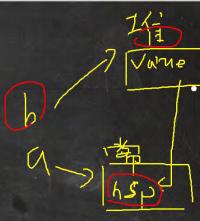


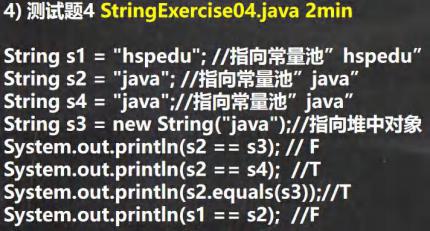
13.2.4 课堂测试题 StringExercise01.java

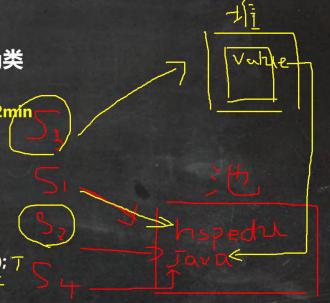


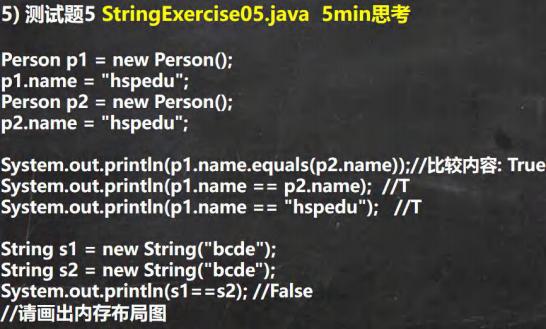


对应的示意图:





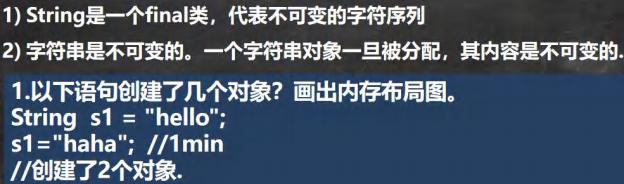






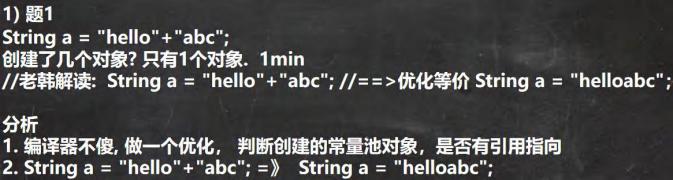
13.3 字符串的特性

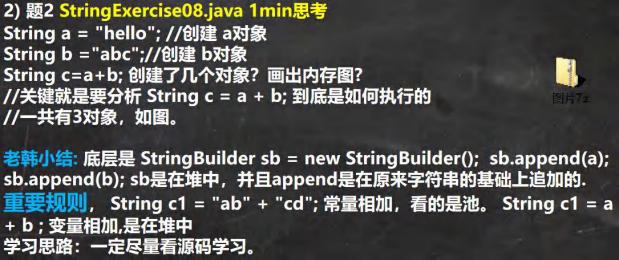
13.3.1 说明 StringExercise06.java

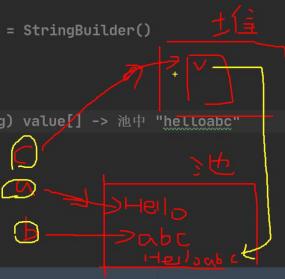


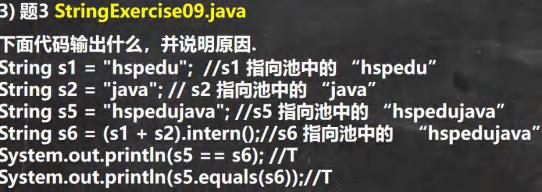


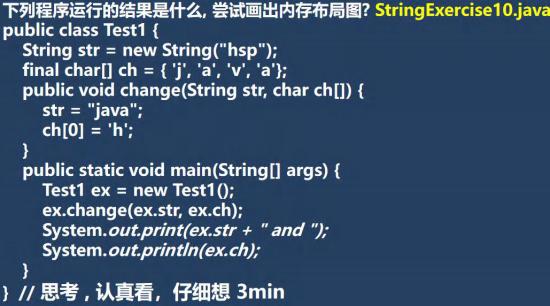
13.3.2 面试题

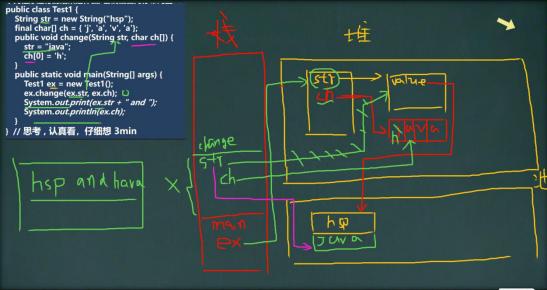






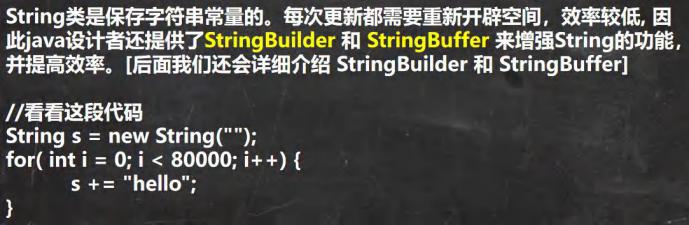




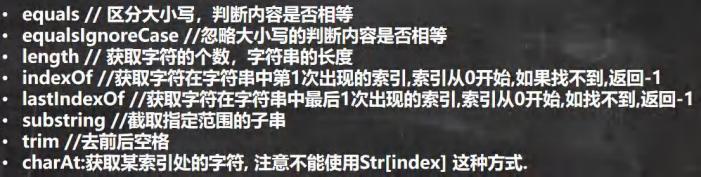


13.4 String 类的常见方法

13.4.1 说明



13.4.2 String 类的常见方法一览

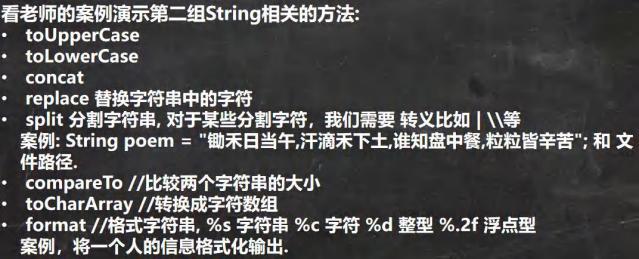


代码:

|  |
| --- |
| package com.hspedu.string\_;  /\*\*  \* @author 韩顺平  \* @version 1.0  \*/  public class StringMethod01 {  public static void main(String[] args) {  //1. equals 前面已经讲过了. 比较内容是否相同，区分大小写  String str1 = "hello"; |

|  |
| --- |
| String str2 = "Hello";  System.out.println(str1.equals(str2));//  // 2.equalsIgnoreCase 忽略大小写的判断内容是否相等  String username = "johN";  if ("john".equalsIgnoreCase(username)) {  System.out.println("Success!");  } else {  System.out.println("Failure!");  }  // 3.length 获取字符的个数，字符串的长度  System.out.println("韩顺平".length());  // 4.indexOf 获取字符在字符串对象中第一次出现的索引，索引从 0 开始，如果找不到，返回- 1  String s1 = "wer@terwe@g";  int index = s1.indexOf('@');  System.out.println(index);// 3  System.out.println("weIndex=" + s1.indexOf("we"));//0  // 5.lastIndexOf 获取字符在字符串中最后一次出现的索引，索引从 0 开始，如果找不到，返回- 1  s1 = "wer@terwe@g@";  index = s1.lastIndexOf('@');  System.out.println(index);//11  System.out.println("ter 的位置=" + s1.lastIndexOf("ter"));//4  // 6.substring 截取指定范围的子串  String name = "hello,张三";  //下面 name.substring(6) 从索引 6 开始截取后面所有的内容  System.out.println(name.substring(6));//截取后面的字符 |

|  |
| --- |
| //name.substring(0,5)表示从索引 0 开始截取，截取到索引 5- 1=4 位置  System.out.println(name.substring(2,5));//llo  }  } |



|  |
| --- |
| package com.hspedu.string\_;  /\*\*  \* @author 韩顺平  \* @version 1.0  \*/  public class StringMethod02 {  public static void main(String[] args) {  // 1.toUpperCase 转换成大写  String s = "heLLo";  System.out.println(s.toUpperCase());//HELLO  // 2.toLowerCase |

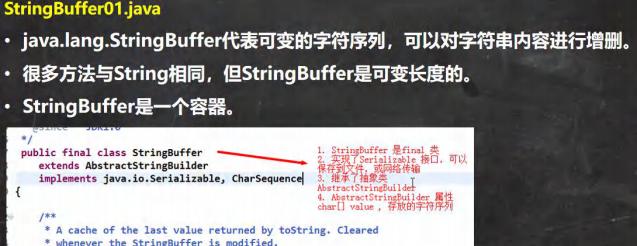
|  |
| --- |
| System.out.println(s.toLowerCase());//hello  // 3.concat 拼接字符串  String s1 = "宝玉";  s1 = s1.concat("林黛玉").concat("薛宝钗").concat("together");  System.out.println(s1);//宝玉林黛玉薛宝钗 together  // 4.replace 替换字符串中的字符  s1 = "宝玉 and 林黛玉 林黛玉 林黛玉";  //在s1 中，将 所有的 林黛玉 替换成薛宝钗  // 老韩解读: s1.replace() 方法执行后，返回的结果才是替换过的.  // 注意对 s1 没有任何影响  String s11 = s1.replace("宝玉", "jack");  System.out.println(s1);//宝玉 and 林黛玉 林黛玉 林黛玉  System.out.println(s11);//jack and 林黛玉 林黛玉 林黛玉  // 5.split 分割字符串, 对于某些分割字符，我们需要 转义比如 | \\等  String poem= "锄禾日当午,汗滴禾下土,谁知盘中餐,粒粒皆辛苦";  //老韩解读：  // 1. 以 , 为标准对 poem 进行分割 , 返回一个数组  // 2. 在对字符串进行分割时，如果有特殊字符，需要加入 转义符 \  String[] split = poem.split(",");  poem = "E:\\aaa\\bbb";  split = poem.split("\\\\");  System.out.println("==分割后内容===");  for (int i = 0; i < split.length; i++) {  System.out.println(split[i]);  }  // 6.toCharArray 转换成字符数组 |

|  |
| --- |
| s = "happy";  char[] chs = s.toCharArray();  for (int i = 0; i < chs.length; i++) {  System.out.println(chs[i]);  }  // 7.compareTo 比较两个字符串的大小，如果前者大，  // 则返回正数，后者大，则返回负数，如果相等，返回 0  // 老韩解读  // (1) 如果长度相同，并且每个字符也相同，就返回 0  // (2) 如果长度相同或者不相同，但是在进行比较时，可以区分大小  // 就返回 if (c1 != c2) {  // return c1 - c2;  // }  // (3) 如果前面的部分都相同，就返回 str1.len - str2.len  String a = "jcck";// len = 3  String b = "jack";// len = 4  System.out.println(a.compareTo(b)); // 返回值是 'c' - 'a' = 2 的值  // 8.format 格式字符串  /\* 占位符有:  \* %s 字符串 %c 字符 %d 整型 %.2f 浮点型  \*  \*/  String name = "john";  int age = 10;  double score = 56.857;  char gender = '男'; |

|  |
| --- |
| //将所有的信息都拼接在一个字符串.  String info =  "我的姓名是" + name + "年龄是" + age + ",成绩是" + score + "性别是" + gender + " 。希望大家喜欢我！  "  ;  System.out.println(info);  //老韩解读  //1. %s , %d , %.2f %c 称为占位符  //2. 这些占位符由后面变量来替换  //3. %s 表示后面由 字符串来替换  //4. %d 是整数来替换  //5. %.2f 表示使用小数来替换，替换后，只会保留小数点两位, 并且进行四舍五入的处理  //6. %c 使用 char 类型来替换  String formatStr = "我的姓名是%s 年龄是%d ，成绩是%.2f 性别是%c.希望大家喜欢我！";  String info2 = String.format(formatStr, name, age, score, gender);  System.out.println("info2=" + info2);  }  } |

13.5 StringBuffer 类

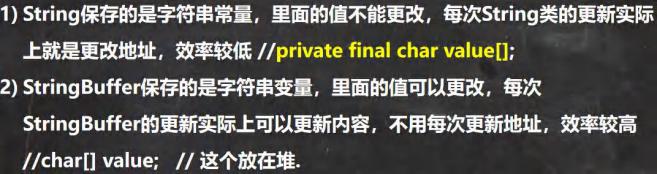
13.5.1 基本介绍



|  |
| --- |
| package com.hspedu.stringbuffer\_;  /\*\*  \* @author 韩顺平  \* @version 1.0  \*/  public class StringBuffer01 {  public static void main(String[] args) {  //老韩解读  //1. StringBuffer 的直接父类 是 AbstractStringBuilder  //2. StringBuffer 实现了 Serializable, 即 StringBuffer 的对象可以串行化  //3. 在父类中 AbstractStringBuilder 有属性 char[] value,不是 final  // 该 value 数组存放 字符串内容，引出存放在堆中的  //4. StringBuffer 是一个 final 类，不能被继承  //5. 因为 StringBuffer 字符内容是存在 char[] value, 所有在变化(增加/删除)  // 不用每次都更换地址(即不是每次创建新对象) ， 所以效率高于 String  StringBuffer stringBuffer = new StringBuffer("hello"); |

|  |
| --- |
| }  } |

13.5.2 String VS StringBuffer



13.5.3 String 和 StringBuffer 相互转换



|  |
| --- |
| package com.hspedu.stringbuffer\_;  /\*\*  \* @author 韩顺平  \* @version 1.0  \*/  public class StringAndStringBuffer {  public static void main(String[] args) {  //看 String——>StringBuffer |

|  |
| --- |
| String str = "hello tom";  //方式 1 使用构造器  //注意： 返回的才是 StringBuffer 对象，对 str 本身没有影响  StringBuffer stringBuffer = new StringBuffer(str);  //方式 2 使用的是 append 方法  StringBuffer stringBuffer1 = new StringBuffer();  stringBuffer1 = stringBuffer1.append(str);  //看看 StringBuffer ->String  StringBuffer stringBuffer3 = new StringBuffer("韩顺平教育");  //方式 1 使用 StringBuffer 提供的 toString 方法  String s = stringBuffer3.toString();  //方式 2: 使用构造器来搞定  String s1 = new String(stringBuffer3);  }  } |

13.5.4 StringBuffer 类常见方法

|  |
| --- |
| package com.hspedu.stringbuffer\_;  /\*\*  \* @author 韩顺平  \* @version 1.0  \*/ |

|  |
| --- |
| public class StringBufferMethod {  public static void main(String[] args) {  StringBuffer s = new StringBuffer("hello");  //增  s.append(',');// "hello,"  s.append("张三丰");//"hello,张三丰"  s.append("赵敏").append(100).append(true).append(10.5);//"hello,张三丰赵敏 100true10.5"  System.out.println(s);//"hello,张三丰赵敏 100true10.5"  //删  /\*  \* 删除索引为>=start && <end 处的字符  \* 解读: 删除 11~ 14 的字符 [11, 14)  \*/  s.delete(11, 14);  System.out.println(s);//"hello,张三丰赵敏 true10.5"  //改  //老韩解读，使用 周芷若 替换 索引 9- 11 的字符 [9, 11)  s.replace(9, 11, "周芷若");  System.out.println(s);//"hello,张三丰周芷若 true10.5"  //查找指定的子串在字符串第一次出现的索引，如果找不到返回- 1  int indexOf= s.indexOf("张三丰");  System.out.println(indexOf);//6  //插 |

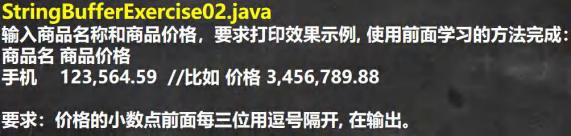
|  |
| --- |
| //老韩解读，在索引为 9 的位置插入 "赵敏",原来索引为 9 的内容自动后移  s.insert(9, "赵敏");  System.out.println(s);//"hello,张三丰赵敏周芷若 true10.5"  //长度  System.out.println(s.length());//22  System.out.println(s);  }  } |

13.5.5 StringBuffer 类课堂测试题 1 StringBufferExercise01.java

|  |
| --- |
| package com.hspedu.stringbuffer\_;  /\*\*  \* @author 韩顺平  \* @version 1.0  \*/  public class StringBufferExercise01 {  public static void main(String[] args) {  String str = null;// ok  StringBuffer sb = new StringBuffer(); //ok  sb.append(str);//需要看源码 , 底层调用的是 AbstractStringBuilder 的 appendNull  System.out.println(sb.length());//4  System.out.println(sb);//null |

|  |
| --- |
| //下面的构造器，会抛出 NullpointerException  StringBuffer sb1 = new StringBuffer(str);//看底层源码 super(str.length() + 16);  System.out.println(sb1);  }  } |

13.5.6 StringBuffer 类课后练习 2



代码

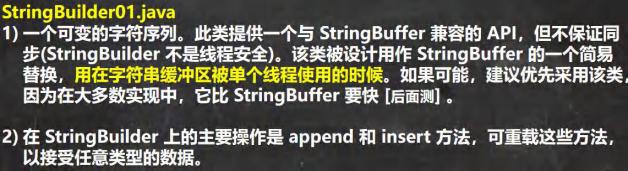
|  |
| --- |
| package com.hspedu.stringbuffer\_;  import java.util.Scanner;  /\*\*  \* @author 韩顺平  \* @version 1.0  \*/  public class StringBufferExercise02 {  public static void main(String[] args) {  /\*  输入商品名称和商品价格，要求打印效果示例, 使用前面学习的方法完成：  商品名 商品价格 |

|  |
| --- |
| 手机 123,564.59 //比如 价格 3,456,789.88  要求：价格的小数点前面每三位用逗号隔开, 在输出。  思路分析  1. 定义一个 Scanner 对象，接收用户输入的 价格(String)  2. 希望使用到 StringBuffer 的 insert ，需要将 String 转成 StringBuffer  3. 然后使用相关方法进行字符串的处理  代码实现  \*/  //new Scanner(System.in)  String price = "8123564.59";  StringBuffer sb = new StringBuffer(price);  //先完成一个最简单的实现 123,564.59  //找到小数点的索引，然后在该位置的前 3 位，插入, 即可  // int i = sb.lastIndexOf(".");  // sb = sb.insert(i - 3, ",");  //上面的两步需要做一个循环处理,才是正确的  for (int i = sb.lastIndexOf(".") - 3; i > 0; i -= 3) {  sb = sb.insert(i, ",");  }  System.out.println(sb);//8, 123,564.59 |

|  |
| --- |
| }  } |

13.6 StringBuilder 类

13.6.1 基本介绍

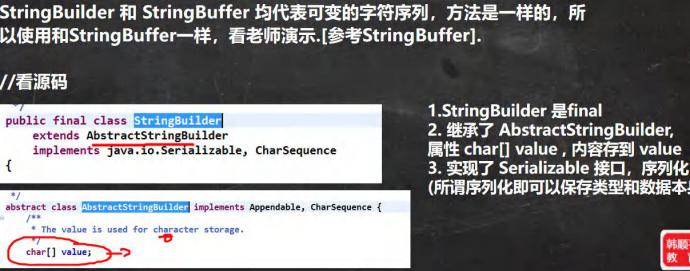


代码

|  |
| --- |
| package com.hspedu.stringbuffer\_;  /\*\*  \* @author 韩顺平  \* @version 1.0  \*/  public class StringBuffer01 {  public static void main(String[] args) {  //老韩解读  //1. StringBuffer 的直接父类 是 AbstractStringBuilder  //2. StringBuffer 实现了 Serializable, 即 StringBuffer 的对象可以串行化  //3. 在父类中 AbstractStringBuilder 有属性 char[] value,不是 final  // 该 value 数组存放 字符串内容，引出存放在堆中的 |

|  |
| --- |
| //4. StringBuffer 是一个 final 类，不能被继承  //5. 因为 StringBuffer 字符内容是存在 char[] value, 所有在变化(增加/删除)  // 不用每次都更换地址(即不是每次创建新对象) ， 所以效率高于 String  StringBuffer stringBuffer = new StringBuffer("hello");  }  } |

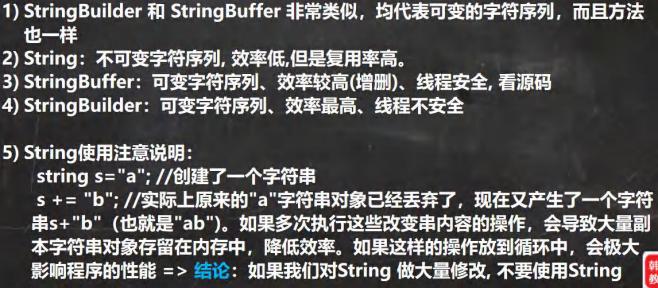
13.6.2 StringBuilder 常用方法



|  |
| --- |
| package com.hspedu.stringbuilder\_;  /\*\*  \* @author 韩顺平  \* @version 1.0  \*/  public class StringBuilder01 {  public static void main(String[] args) { |

|  |
| --- |
| //老韩解读  //1. StringBuilder 继承 AbstractStringBuilder 类  //2. 实现了 Serializable ,说明 StringBuilder 对象是可以串行化(对象可以网络传输,可以保存到文件)  //3. StringBuilder 是 final 类, 不能被继承  //4. StringBuilder 对象字符序列仍然是存放在其父类 AbstractStringBuilder 的 char[] value;  // 因此，字符序列是堆中  //5. StringBuilder 的方法，没有做互斥的处理, 即没有 synchronized 关键字, 因此在单线程的情况下使用  // StringBuilder  StringBuilder stringBuilder = new StringBuilder();  }  } |

13.6.3 String 、StringBuffer 和 StringBuilder 的比较



13.6.4 String 、StringBuffer 和 StringBuilder 的效率测试

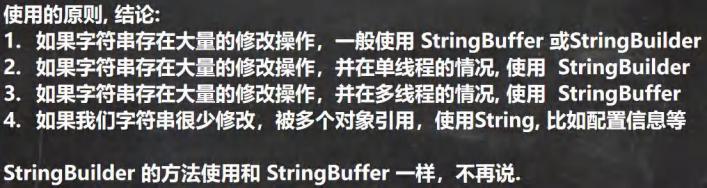
StringVsStringBufferVsStringBuilder.java 效率 ： StringBuilder > StringBuffer > String

|  |
| --- |
| package com.hspedu.stringbuilder\_; |

|  |
| --- |
| /\*\*  \* @author 韩顺平  \* @version 1.0  \*/  public class StringVsStringBufferVsStringBuilder {  public static void main(String[] args) {  long startTime = 0L;  long endTime = 0L;  StringBuffer buffer = new StringBuffer("");  startTime = System.currentTimeMillis();  for (int i = 0; i < 80000; i++) {//StringBuffer 拼接 20000 次  buffer.append(String.valueOf(i));  }  endTime = System.currentTimeMillis();  System.out.println("StringBuffer 的执行时间：" + (endTime - startTime));  StringBuilder builder = new StringBuilder("");  startTime = System.currentTimeMillis();  for (int i = 0; i < 80000; i++) {//StringBuilder 拼接 20000 次 |

|  |
| --- |
| builder.append(String.valueOf(i));  }  endTime = System.currentTimeMillis();  System.out.println("StringBuilder 的执行时间：" + (endTime - startTime));  String text = "";  startTime = System.currentTimeMillis();  for (int i = 0; i < 80000; i++) {//String 拼接 20000  text = text + i;  }  endTime = System.currentTimeMillis();  System.out.println("String 的执行时间：" + (endTime - startTime));  }  } |

13.6.5 String 、StringBuffer 和 StringBuilder 的选择

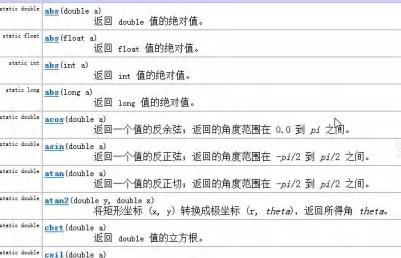


13.7 Math 类

13.7.1 基本介绍

Math 类包含用于执行基本数学运算的方法，如初等指数、对数、平方根和三角函数。

13.7.2 方法一览(均为静态方法)



13.7.3 Math 类常见方法应用案例

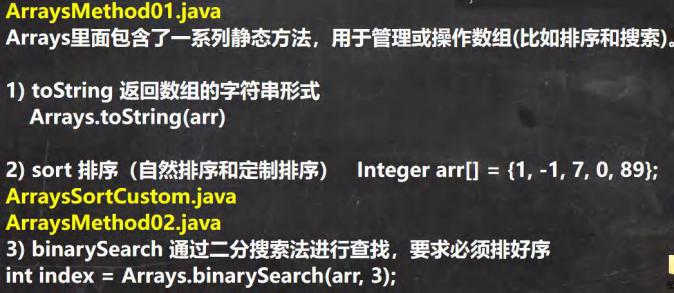
|  |
| --- |
| package com.hspedu.math\_;  /\*\*  \* @author 韩顺平  \* @version 1.0  \*/  public class MathMethod {  public static void main(String[] args) {  //看看 Math 常用的方法(静态方法)  //1.abs 绝对值  int abs = Math.abs(-9);  System.out.println(abs);//9 |

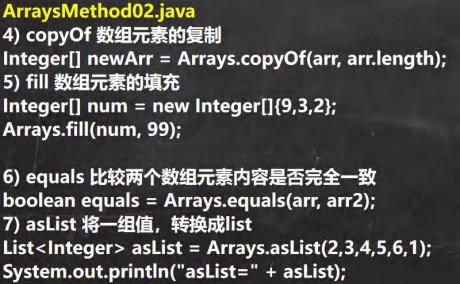
|  |
| --- |
| //2.pow 求幂  double pow = Math.pow(2, 4);//2 的 4 次方  System.out.println(pow);//16  //3.ceil 向上取整,返回>=该参数的最小整数(转成 double);  double ceil = Math.ceil(3.9);  System.out.println(ceil);//4.0  //4.floor 向下取整，返回<=该参数的最大整数(转成 double)  double floor = Math.floor(4.001);  System.out.println(floor);//4.0  //5.round 四舍五入 Math.floor(该参数+0.5)  long round = Math.round(5.51);  System.out.println(round);//6  //6.sqrt 求开方  double sqrt = Math.sqrt(9.0);  System.out.println(sqrt);//3.0  //7.random 求随机数  // random 返回的是 0 <= x < 1 之间的一个随机小数  // 思考：请写出获取 a-b 之间的一个随机整数,a,b 均为整数 ，比如 a = 2, b=7  // 即返回一个数 x 2 <= x <= 7  // 老韩解读 Math.random() \* (b-a) 返回的就是 0 <= 数 <= b-a  // (1) (int)(a) <= x <= (int)(a + Math.random() \* (b-a +1) )  // (2) 使用具体的数给小伙伴介绍 a = 2 b = 7  // (int)(a + Math.random() \* (b-a +1) ) = (int)( 2 + Math.random()\*6)  // Math.random()\*6 返回的是 0 <= x < 6 小数  // 2 + Math.random()\*6 返回的就是 2<= x < 8 小数 |

|  |
| --- |
| // (int)(2 + Math.random()\*6) = 2 <= x <= 7  // (3) 公式就是 (int)(a + Math.random() \* (b-a +1) )  for(int i = 0; i < 100; i++) {  System.out.println((int)(2 + Math.random() \* (7 - 2 + 1)));  }  //max , min 返回最大值和最小值  int min = Math.min(1, 9);  int max = Math.max(45, 90);  System.out.println("min=" + min);  System.out.println("max=" + max);  }  } |

13.8 Arrays 类

13.8.1 Arrays 类常见方法应用案例





|  |
| --- |
| package com.hspedu.arrays\_;  import java.util.Arrays;  import java.util.Comparator;  /\*\* |

|  |
| --- |
| \* @author 韩顺平  \* @version 1.0  \*/  public class ArraysMethod01 {  public static void main(String[] args) {  Integer[] integers = {1, 20, 90};  //遍历数组  // for(int i = 0; i < integers.length; i++) {  // System.out.println(integers[i]);  // }  //直接使用 Arrays.toString 方法，显示数组  // System.out.println(Arrays.toString(integers));//  //演示 sort 方法的使用  Integer arr[] = {1, - 1, 7, 0, 89};  //进行排序  //老韩解读  //1. 可以直接使用冒泡排序 , 也可以直接使用 Arrays 提供的 sort 方法排序  //2. 因为数组是引用类型，所以通过 sort 排序后，会直接影响到 实参 arr  //3. sort 重载的，也可以通过传入一个接口 Comparator 实现定制排序  //4. 调用 定制排序 时，传入两个参数 (1) 排序的数组 arr  // (2) 实现了 Comparator 接口的匿名内部类 , 要求实现 compare 方法  //5. 先演示效果，再解释  //6. 这里体现了接口编程的方式 , 看看源码，就明白 |

|  |
| --- |
| // 源码分析  //(1) Arrays.sort(arr, new Comparator()  //(2) 最终到 TimSort 类的 private static <T> void binarySort(T[] a, int lo, int hi, int start,  // Comparator<? super T> c)()  //(3) 执行到 binarySort 方法的代码, 会根据动态绑定机制 c.compare()执行我们传入的  // 匿名内部类的 compare ()  // while (left < right) {  // int mid = (left + right) >>> 1;  // if (c.compare(pivot, a[mid]) < 0)  // right = mid;  // else  // left = mid + 1;  // }  //(4) new Comparator() {  // @Override  // public int compare(Object o1, Object o2) {  // Integer i1 = (Integer) o1;  // Integer i2 = (Integer) o2;  // return i2 - i1;  // }  // }  //(5) public int compare(Object o1, Object o2) 返回的值>0 还是 <0  // 会影响整个排序结果, 这就充分体现了 接口编程+动态绑定+匿名内部类的综合使用  // 将来的底层框架和源码的使用方式，会非常常见  //Arrays.sort(arr); // 默认排序方法  //定制排序 |

|  |
| --- |
| Arrays.sort(arr, new Comparator() {  @Override  public int compare(Object o1, Object o2) {  Integer i1 = (Integer) o1;  Integer i2 = (Integer) o2;  return i2 - i1;  }  });  System.out.println("===排序后===");  System.out.println(Arrays.toString(arr));//  }  } |
| package com.hspedu.arrays\_;  import java.util.Arrays;  import java.util.Comparator;  /\*\*  \* @author 韩顺平  \* @version 1.0  \*/  public class ArraysSortCustom { |

|  |
| --- |
| public static void main(String[] args) {  int[] arr = {1, - 1, 8, 0, 20};  //bubble01(arr);  bubble02(arr, new Comparator() {  @Override  public int compare(Object o1, Object o2) {  int i1 = (Integer) o1;  int i2 = (Integer) o2;  return i2 - i1;// return i2 - i1;  }  });  System.out.println("==定制排序后的情况==");  System.out.println(Arrays.toString(arr));  }  //使用冒泡完成排序  public static void bubble01(int[] arr) {  int temp = 0;  for (int i = 0; i < arr.length - 1; i++) {  for (intj = 0; j < arr.length - 1 - i; j++) {  //从小到大  if (arr[j] > arr[j + 1]) { |

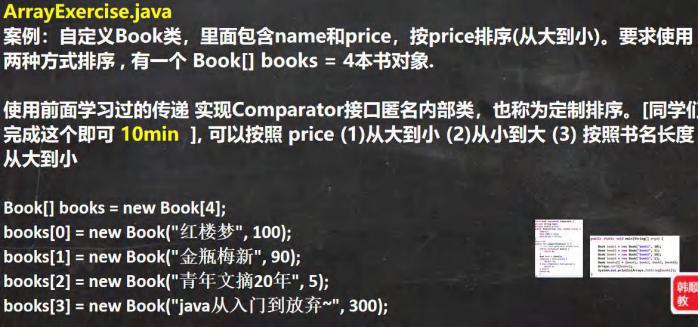
|  |
| --- |
| temp = arr[j];  arr[j] = arr[j + 1];  arr[j + 1] = temp;  }  }  }  }  //结合冒泡 + 定制  public static void bubble02(int[] arr, Comparator c) {  int temp = 0;  for (int i = 0; i < arr.length - 1; i++) {  for (intj = 0; j < arr.length - 1 - i; j++) {  //数组排序由 c.compare(arr[j], arr[j + 1])返回的值决定  if (c.compare(arr[j], arr[j + 1]) > 0) {  temp = arr[j];  arr[j] = arr[j + 1];  arr[j + 1] = temp;  }  }  }  }  } |
| package com.hspedu.arrays\_; |

|  |
| --- |
| import java.util.Arrays;  import java.util.List;  /\*\*  \* @author 韩顺平  \* @version 1.0  \*/  public class ArraysMethod02 {  public static void main(String[] args) {  Integer[] arr = {1, 2, 90, 123, 567};  // binarySearch 通过二分搜索法进行查找，要求必须排好  // 老韩解读  //1. 使用 binarySearch 二叉查找  //2. 要求该数组是有序的. 如果该数组是无序的，不能使用 binarySearch  //3. 如果数组中不存在该元素，就返回 return -(low + 1); // key not found.  int index = Arrays.binarySearch(arr, 567);  System.out.println("index=" + index);  //copyOf 数组元素的复制  // 老韩解读  //1. 从 arr 数组中，拷贝 arr.length 个元素到 newArr 数组中  //2. 如果拷贝的长度 > arr.length 就在新数组的后面 增加 null  //3. 如果拷贝长度 < 0 就抛出异常 NegativeArraySizeException  //4. 该方法的底层使用的是 System.arraycopy()  Integer[] newArr = Arrays.copyOf(arr, arr.length);  System.out.println("==拷贝执行完毕后=="); |

|  |
| --- |
| System.out.println(Arrays.toString(newArr));  //ill 数组元素的填充  Integer[] num = new Integer[]{9,3,2};  //老韩解读  //1. 使用 99 去填充 num 数组，可以理解成是替换原理的元素  Arrays.fill(num, 99);  System.out.println("==num 数组填充后==");  System.out.println(Arrays.toString(num));  //equals 比较两个数组元素内容是否完全一致  Integer[] arr2 = {1, 2, 90, 123};  //老韩解读  //1. 如果 arr 和 arr2 数组的元素一样，则方法 true;  //2. 如果不是完全一样，就返回 false  boolean equals = Arrays.equals(arr, arr2);  System.out.println("equals=" + equals);  //asList 将一组值，转换成 list  //老韩解读  //1. asList 方法，会将 (2,3,4,5,6, 1)数据转成一个 List 集合  //2. 返回的 asList 编译类型 List(接口)  //3. asList 运行类型 java.util.Arrays#ArrayList, 是 Arrays 类的  // 静态内部类 private static class ArrayList<E> extends AbstractList<E>  // implements RandomAccess, java.io.Serializable  List asList = Arrays.asList(2,3,4,5,6, 1); |

|  |
| --- |
| System.out.println("asList=" + asList);  System.out.println("asList 的运行类型" + asList.getClass());  }  } |

13.8.2 Arrays 类课堂练习



代码

|  |
| --- |
| package com.hspedu.arrays\_;  import java.util.Arrays;  import java.util.Comparator; |

|  |
| --- |
| /\*\*  \* @author 韩顺平  \* @version 1.0  \*/  public class ArrayExercise {  public static void main(String[] args) {  /\*  案例： 自定义 Book 类，里面包含 name 和 price ，按 price 排序(从大到小)。  要求使用两种方式排序 , 有一个 Book[] books= 4 本书对象.  使用前面学习过的传递 实现 Comparator接口匿名内部类，也称为定制排序。  [同学们完成这个即可 10min ],  可以按照 price (1)从大到小 (2)从小到大 (3) 按照书名长度从大到小  \*/  Book[] books = new Book[4];  books[0] = new Book("红楼梦", 100);  books[1] = new Book("金瓶梅新", 90);  books[2] = new Book("青年文摘 20 年", 5);  books[3] = new Book("java 从入门到放弃~", 300);  //(1)price 从大到小  // Arrays.sort(books, new Comparator() { |

|  |
| --- |
| // //这里是对 Book 数组排序，因此 o1 和 o2 就是 Book 对象  // @Override  // public int compare(Object o1, Object o2) {  // Book book1 = (Book) o1;  // Book book2 = (Book) o2;  // double priceVal = book2.getPrice() - book1.getPrice();  // //这里老师进行了一个转换  // //如果发现返回结果和我们输出的不一致，就修改一下返回的 1 和 - 1  // if(priceVal > 0) {  // return 1;  // } else if(priceVal < 0) {  // return - 1;  // } else {  // return 0;  // }  // }  // });  //(2)price 从小到大  // Arrays.sort(books, new Comparator() {  // //这里是对 Book 数组排序，因此 o1 和 o2 就是 Book 对象  // @Override  // public int compare(Object o1, Object o2) {  // Book book1 = (Book) o1;  // Book book2 = (Book) o2;  // double priceVal = book2.getPrice() - book1.getPrice(); |

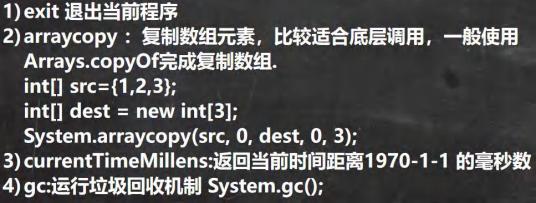
|  |
| --- |
| // //这里老师进行了一个转换  // //如果发现返回结果和我们输出的不一致，就修改一下返回的 1 和 - 1  // if(priceVal > 0) {  // return - 1;  // } else if(priceVal < 0) {  // return 1;  // } else {  // return 0;  // }  // }  // });  //(3)按照书名长度从大到小  Arrays.sort(books, new Comparator() {  //这里是对 Book 数组排序，因此 o1 和 o2 就是 Book 对象  @Override  public int compare(Object o1, Object o2) {  Book book1 = (Book) o1;  Book book2 = (Book) o2;  //要求按照书名的长度来进行排序  return book2.getName().length() - book1.getName().length();  }  }); |

|  |
| --- |
| System.out.println(Arrays.toString(books));  }  }  class Book {  private String name;  private double price;  public Book(String name, double price) {  this.name = name;  this.price = price;  }  public String getName() {  return name;  }  public void setName(String name) {  this.name = name;  }  public double getPrice() {  return price;  } |

|  |
| --- |
| public void setPrice(double price) {  this.price = price;  }  @Override  public String toString() {  return "Book{" +  "name='" + name + '\'' +  ", price=" + price +  '}';  }  } |

13.9 System 类

13.9.1 System 类常见方法和案例



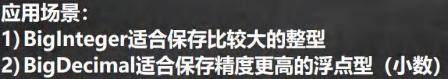
|  |
| --- |
| package com.hspedu.system\_;  import java.util.Arrays; |

|  |
| --- |
| /\*\*  \* @author 韩顺平  \* @version 1.0  \*/  public class System\_ {  public static void main(String[] args) {  //exit 退出当前程序  // System.out.println("ok1");  // //老韩解读  // //1. exit(0) 表示程序退出  // //2. 0 表示一个状态 , 正常的状态  // System.exit(0);//  // System.out.println("ok2");  //arraycopy ：复制数组元素，比较适合底层调用，  // 一般使用 Arrays.copyOf 完成复制数组  int[] src={1,2,3};  int[] dest = new int[3];// dest 当前是 {0,0,0}  //老韩解读  //1. 主要是搞清楚这五个参数的含义  //2.  // 源数组 |

|  |
| --- |
| // \* @param src the source array.  // srcPos： 从源数组的哪个索引位置开始拷贝  // \* @param srcPos starting position in the source array.  // dest : 目标数组，即把源数组的数据拷贝到哪个数组  // \* @param dest the destination array.  // destPos: 把源数组的数据拷贝到 目标数组的哪个索引  // \* @param destPos starting position in the destination data.  // length: 从源数组拷贝多少个数据到目标数组  // \* @param length the number of array elements to be copied.  System.arraycopy(src, 0, dest, 0, src.length);  // int[] src={1,2,3};  System.out.println("dest=" + Arrays.toString(dest));//[1, 2, 3]  //currentTimeMillens:返回当前时间距离 1970- 1- 1 的毫秒数  // 老韩解读:  System.out.println(System.currentTimeMillis());  }  } |

13. 10BigInteger 和 BigDecimal 类

13.10. 1 BigInteger 和 BigDecimal 介绍



13.10.2 BigInteger 和 BigDecimal 常见方法



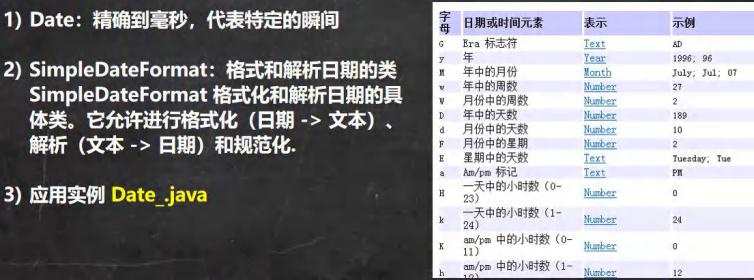
|  |
| --- |
| package com.hspedu.bignum;  import java.math.BigInteger;  /\*\*  \* @author 韩顺平  \* @version 1.0  \*/  public class BigInteger\_ {  public static void main(String[] args) {  //当我们编程中，需要处理很大的整数，long 不够用  //可以使用 BigInteger 的类来搞定  // long l = 23788888899999999999999999999l;  // System.out.println("l=" + l);  BigInteger bigInteger = new BigInteger("23788888899999999999999999999");  BigInteger bigInteger2 = new |

|  |
| --- |
| BigInteger(" 10099999999999999999999999999999999999999999999999999999999999999999999999999999999");  System.out.println(bigInteger);  //老韩解读  //1. 在对 BigInteger 进行加减乘除的时候，需要使用对应的方法，不能直接进行 + - \* /  //2. 可以创建一个 要操作的 BigInteger 然后进行相应操作  BigInteger add = bigInteger.add(bigInteger2);  System.out.println(add);//  BigInteger subtract = bigInteger.subtract(bigInteger2);  System.out.println(subtract);//减  BigInteger multiply = bigInteger.multiply(bigInteger2);  System.out.println(multiply);//乘  BigInteger divide = bigInteger.divide(bigInteger2);  System.out.println(divide);//除  }  } |
| package com.hspedu.bignum;  import java.math.BigDecimal;  /\*\*  \* @author 韩顺平  \* @version 1.0  \*/ |

|  |
| --- |
| public class BigDecimal\_ {  public static void main(String[] args) {  //当我们需要保存一个精度很高的数时，double 不够用  //可以是 BigDecimal  // double d = 1999. 11111111111999999999999977788d;  // System.out.println(d);  BigDecimal bigDecimal = new BigDecimal(" 1999.11");  BigDecimal bigDecimal2 = new BigDecimal("3");  System.out.println(bigDecimal);  //老韩解读  //1. 如果对 BigDecimal 进行运算，比如加减乘除，需要使用对应的方法  //2. 创建一个需要操作的 BigDecimal 然后调用相应的方法即可  System.out.println(bigDecimal.add(bigDecimal2));  System.out.println(bigDecimal.subtract(bigDecimal2));  System.out.println(bigDecimal.multiply(bigDecimal2));  //System.out.println(bigDecimal.divide(bigDecimal2));//可能抛出异常 ArithmeticException  //在调用 divide 方法时，指定精度即可. BigDecimal.ROUND\_CEILING  //如果有无限循环小数，就会保留 分子 的精度  System.out.println(bigDecimal.divide(bigDecimal2, BigDecimal.ROUND\_CEILING));  }  } |

13. 11日期类

13.11. 1 第一代日期类



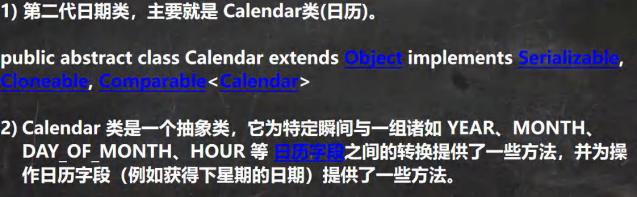
代码

|  |
| --- |
| package com.hspedu.date\_;  import java.text.ParseException;  import java.text.SimpleDateFormat;  import java.util.Date;  /\*\*  \* @author 韩顺平  \* @version 1.0  \*/  public class Date01 {  public static void main(String[] args) throws ParseException { |

|  |
| --- |
| //老韩解读  //1. 获取当前系统时间  //2. 这里的 Date 类是在java.util 包  //3. 默认输出的日期格式是国外的方式, 因此通常需要对格式进行转换  Date d1 = new Date(); //获取当前系统时间  System.out.println(" 当前日期=" + d1);  Date d2 = new Date(9234567); //通过指定毫秒数得到时间  System.out.println("d2=" + d2); //获取某个时间对应的毫秒数  //  //老韩解读  //1. 创建 SimpleDateFormat 对象，可以指定相应的格式  //2. 这里的格式使用的字母是规定好，不能乱写  SimpleDateFormat sdf= new SimpleDateFormat("yyyy 年 MM 月 dd 日 hh:mm:ss E");  String format = sdf.format(d1); // format:将日期转换成指定格式的字符串  System.out.println(" 当前日期=" + format);  //老韩解读  //1. 可以把一个格式化的 String 转成对应的 Date  //2. 得到 Date 仍然在输出时，还是按照国外的形式，如果希望指定格式输出，需要转换  //3. 在把 String -> Date ， 使用的 sdf 格式需要和你给的 String 的格式一样，否则会抛出转换异常  String s = " 1996 年 01 月 01 日 10:20:30 星期一";  Date parse = sdf.parse(s);  System.out.println("parse=" + sdf.format(parse)); |

|  |
| --- |
| }  } |

13.11.2 第二代日期类

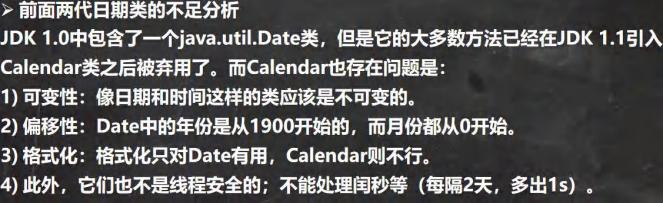


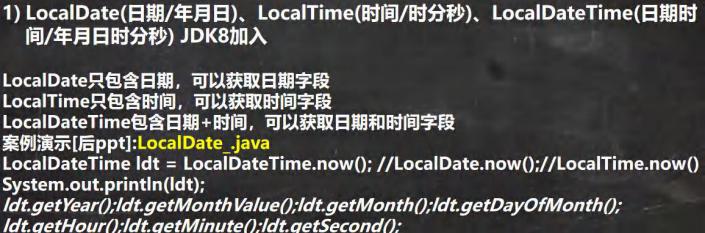
代码

|  |
| --- |
| package com.hspedu.date\_;  import java.util.Calendar;  /\*\*  \* @author 韩顺平  \* @version 1.0  \*/  public class Calendar\_ {  public static void main(String[] args) {  //老韩解读  //1. Calendar 是一个抽象类， 并且构造器是 private  //2. 可以通过 getInstance() 来获取实例  //3. 提供大量的方法和字段提供给程序员 |

|  |
| --- |
| //4. Calendar 没有提供对应的格式化的类，因此需要程序员自己组合来输出(灵活)  //5. 如果我们需要按照 24 小时进制来获取时间， Calendar.HOUR ==改成=> Calendar.HOUR\_OF\_DAY  Calendar c = Calendar.getInstance(); //创建日历类对象//比较简单， 自由  System.out.println("c=" + c);  //2.获取日历对象的某个日历字段  System.out.println("年：" + c.get(Calendar.YEAR));  // 这里为什么要 + 1, 因为 Calendar 返回月时候，是按照 0 开始编号  System.out.println("月：" + (c.get(Calendar.MONTH) + 1));  System.out.println(" 日：" + c.get(Calendar.DAY\_OF\_MONTH));  System.out.println("小时：" + c.get(Calendar.HOUR));  System.out.println("分钟：" + c.get(Calendar.MINUTE));  System.out.println("秒：" + c.get(Calendar.SECOND));  //Calender 没有专门的格式化方法，所以需要程序员自己来组合显示  System.out.println(c.get(Calendar.YEAR) + "-" + (c.get(Calendar.MONTH) + 1) + "-" +  c.get(Calendar.DAY\_OF\_MONTH) +  " " + c.get(Calendar.HOUR\_OF\_DAY) + ":" + c.get(Calendar.MINUTE) + ":" + c.get(Calendar.SECOND) );  }  } |

13.11.3 第三代日期类



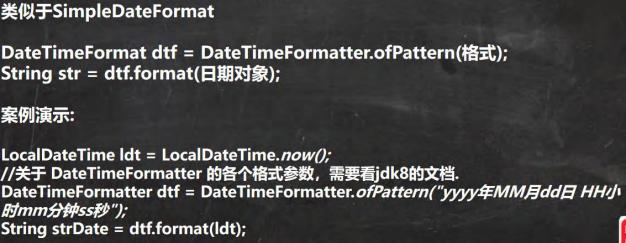


|  |
| --- |
| package com.hspedu.date\_;  import java.time.Instant;  import java.time.LocalDate;  import java.time.LocalDateTime;  import java.time.LocalTime;  import java.time.format.DateTimeFormatter;  import java.util.ArrayList;  import java.util.Collection;  /\*\* |

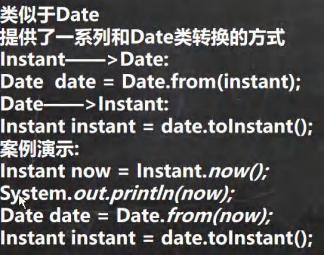
|  |
| --- |
| \* @author 韩顺平  \* @version 1.0  \*/  public class LocalDate\_ {  public static void main(String[] args) {  //第三代日期  //老韩解读  //1. 使用 now() 返回表示当前日期时间的 对象  LocalDateTime ldt = LocalDateTime.now(); //LocalDate.now();//LocalTime.now()  System.out.println(ldt);  //2. 使用 DateTimeFormatter 对象来进行格式化  // 创建 DateTimeFormatter 对象  DateTimeFormatter dateTimeFormatter = DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm:ss");  String format = dateTimeFormatter.format(ldt);  System.out.println("格式化的日期=" + format);  System.out.println("年=" + ldt.getYear());  System.out.println("月=" + ldt.getMonth());  System.out.println("月=" + ldt.getMonthValue());  System.out.println(" 日=" + ldt.getDayOfMonth());  System.out.println("时=" + ldt.getHour());  System.out.println("分=" + ldt.getMinute());  System.out.println("秒=" + ldt.getSecond());  LocalDate now = LocalDate.now(); //可以获取年月日 |

|  |
| --- |
| LocalTime now2 = LocalTime.now();//获取到时分秒  //提供 plus 和 minus 方法可以对当前时间进行加或者减  //看看 890 天后，是什么时候 把 年月日-时分秒  LocalDateTime localDateTime = ldt.plusDays(890);  System.out.println("890 天后=" + dateTimeFormatter.format(localDateTime));  //看看在 3456 分钟前是什么时候，把 年月日-时分秒输出  LocalDateTime localDateTime2 = ldt.minusMinutes(3456);  System.out.println("3456 分钟前 日期=" + dateTimeFormatter.format(localDateTime2));  }  } |

13.11.4 DateTimeFormatter 格式日期类



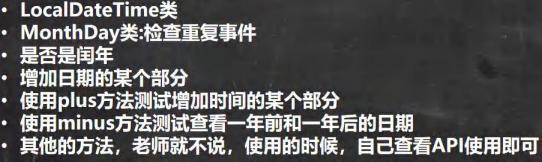
13.11.5 Instant 时间戳



|  |
| --- |
| package com.hspedu.date\_;  import java.time.Instant;  import java.util.Date;  /\*\*  \* @author 韩顺平  \* @version 1.0  \*/  public class Instant\_ {  public static void main(String[] args) {  //1.通过 静态方法 now() 获取表示当前时间戳的对象  Instant now = Instant.now();  System.out.println(now);  //2. 通过 from 可以把 Instant 转成 Date  Date date = Date.from(now); |

|  |
| --- |
| //3. 通过 date 的 toInstant() 可以把 date 转成 Instant 对象  Instant instant = date.toInstant();  }  } |

13.11.6 第三代日期类更多方法



13. 12本章作业

