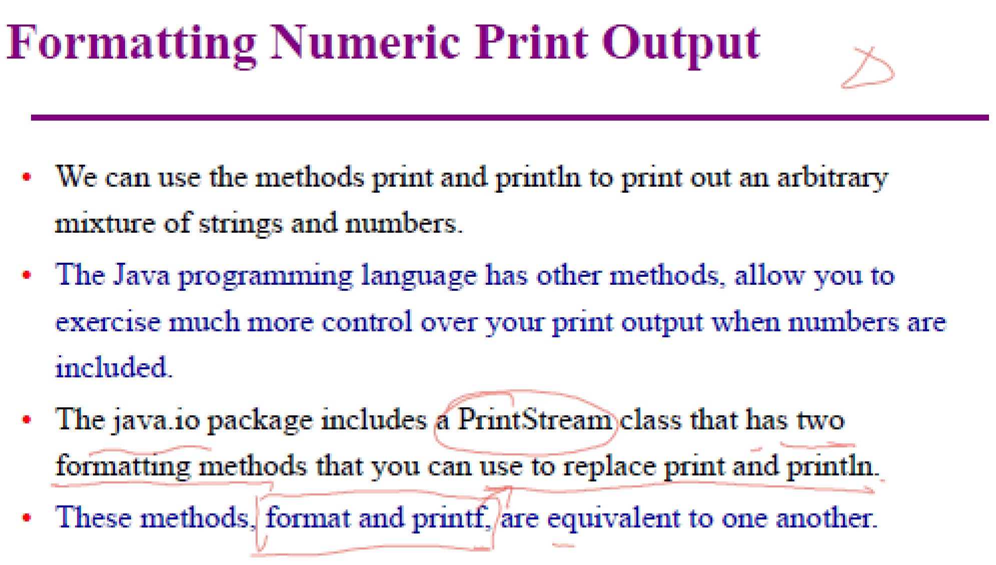
10.23

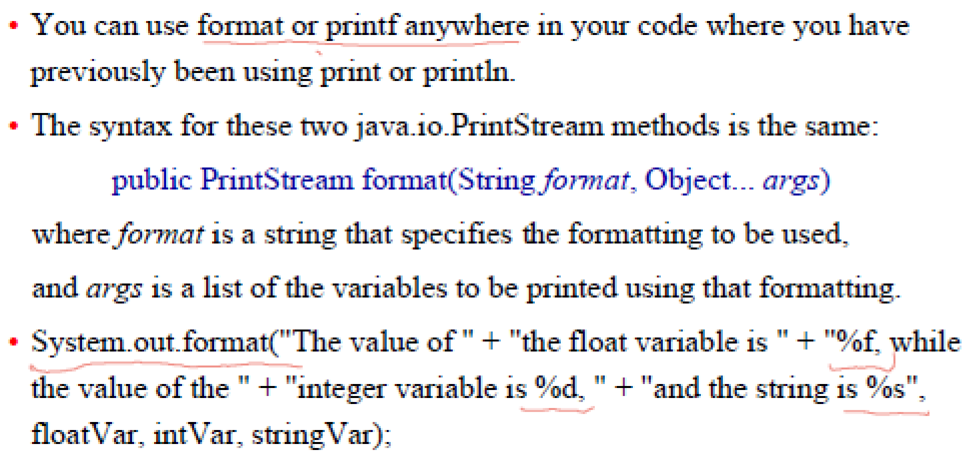
数和字符串

数



Formatting Numeric Print Output

- We can use the methods print and println to print out an arbitrary mixture of strings and numbers.
- The Java programming language has other methods, allow you to exercise much more control over your print output when numbers are included.
- The java.io package includes a PrintStream class that has two formatting methods that you can use to replace print and println.
- These methods, format and printf, are equivalent to one another.

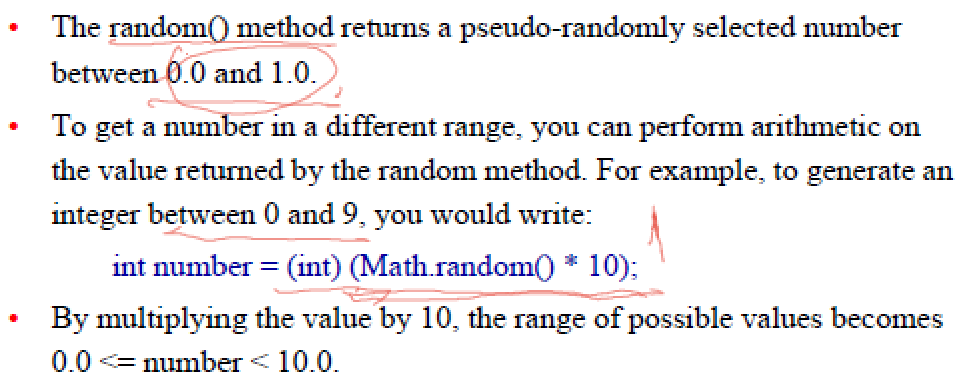


- You can use format or printf anywhere in your code where you have previously been using print or println.
- The syntax for these two java.io.PrintStream methods is the same:

  public PrintStream format(String *format*, Object... *args*)

  where *format* is a string that specifies the formatting to be used, and *args* is a list of the variables to be printed using that formatting.
- System.out.format("The value of " + "the float variable is " + "%f, while the value of the " + "integer variable is %d, " + "and the string is %s", floatVar, intVar, stringVar);

Random ( )



- The random() method returns a pseudo-randomly selected number between 0.0 and 1.0.
- To get a number in a different range, you can perform arithmetic on the value returned by the random method. For example, to generate an integer between 0 and 9, you would write:

  int number = (int) (Math.random() * 10);
- By multiplying the value by 10, the range of possible values becomes 0.0 <= number < 10.0.

- Using Math.random works well when you need to generate a single random number.
- If you need to generate a series of random numbers, you should create an instance of java.util.Random and invoke methods on that object to generate numbers.

Summary of Numbers

- You use one of the [illegible] wrapper classes – Byte, Double, Float, Integer, Long, or Short – to wrap a number of primitive type in an object.
- The Java compiler automatically wraps (boxes) primitives for you when necessary and unboxes them, again when necessary.
- The Number classes include constants and useful class methods.
  - The MIN_VALUE and MAX_VALUE constants contain the smallest and largest values that can be contained by an object of that type.
- The byteValue, shortValue, and similar methods convert one numeric type to another.
- The valueOf method converts a string to a number, and the toString method converts a number to a string.

- To format a string containing numbers for output, you can use the printf() or format() methods in the PrintStream class.
- Alternatively, you can use the NumberFormat class to customize numerical formats using patterns.
- The Math class contains a variety of class methods for performing mathematical functions, including exponential, logarithmic, and trigonometric methods.
- Math also includes basic arithmetic functions, such as absolute value and rounding, and a method, random(), for generating random numbers.

Characters

- There are times, however, when you need to use a char as an object—for example, as a method argument where an object is expected.
- The Java programming language provides a *wrapper* class that "wraps" the char in a Character object for this purpose.
- An object of type Character contains a single field, whose type is char.
- This Character class also offers a number of useful class (i.e., static) methods for manipulating characters.
- You can create a Character object with the Character constructor:

  Character ch = new Character('a');

- The Java compiler will also create a Character object for you under some circumstances.
- For example, if you pass a primitive char into a method that expects an object, the compiler automatically converts the char to a Character for you. This feature is called autoboxing—or unboxing.
- The Character class is immutable, so that once it is created, a Character object cannot be changed.

Strings

不可修改的

- The String class is immutable, so that once it is created a String object cannot be changed.
- The String class has a number of methods that appear to modify strings.
- Since strings are immutable, what these methods really do is create and return a new string that contains the result of the operation.

String palindrome = "Dot saw I was Tod";
int len = palindrome.length();

getChars() 方法将字符从字符串复制到目标字符数组。

语法

```
public void getChars(int srcBegin, int srcEnd, char[] dst,  int dstBegin)
```

参数

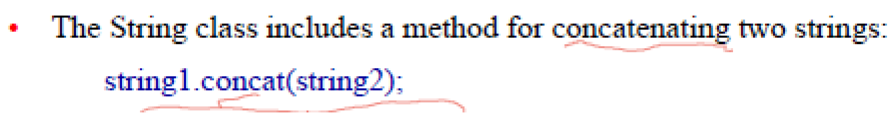
- **srcBegin** -- 字符串中要复制的第一个字符的索引。
- **srcEnd** -- 字符串中要复制的最后一个字符之后的索引。
- **dst** -- 目标数组。
- **dstBegin** -- 目标数组中的起始偏移量。

实例

```
public class Test {
    public static void main(String args[]) {
        String Str1 = new String("www.runoob.com");
        char[] Str2 = new char[6];

        try {
            Str1.getChars(4, 10, Str2, 0);
            System.out.print("拷贝的字符串为: " );
            System.out.println(Str2 );
        } catch( Exception ex) {
            System.out.println("触发异常...");
        }
    }
}
```

以上程序执行结果为：

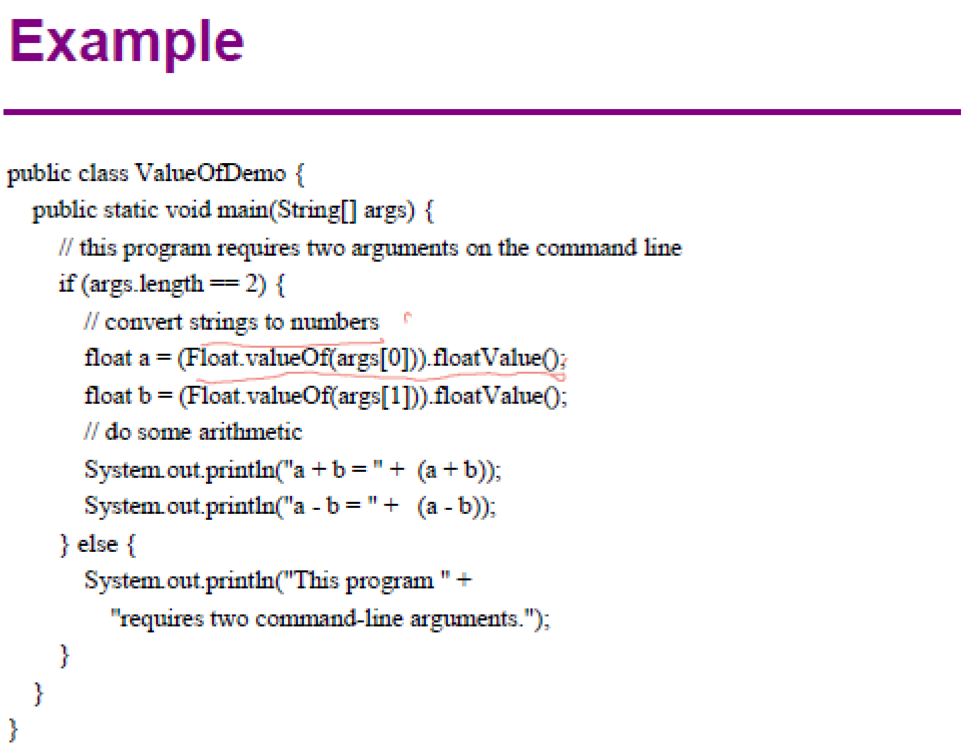
```
拷贝的字符串为：runoob
```



- The String class includes a method for concatenating two strings:

  string1.concat(string2);

将字符串转化为数字 valueOf()



Example

```
public class ValueOfDemo {
   public static void main(String[] args) {
      // this program requires two arguments on the command line
      if (args.length == 2) {
         // convert strings to numbers
         float a = (Float.valueOf(args[0])).floatValue();
         float b = (Float.valueOf(args[1])).floatValue();
         // do some arithmetic
         System.out.println("a + b = " +  (a + b));
         System.out.println("a - b = " +  (a - b));
      } else {
         System.out.println("This program " +
            "requires two command-line arguments.");
      }
   }
}
```

Converting Numbers to Strings

There are several easy ways to convert a number to a string:

```
int i;
// Concatenate "i" with an empty string; conversion is handled for you.
String s1 = "" + i;
```

or

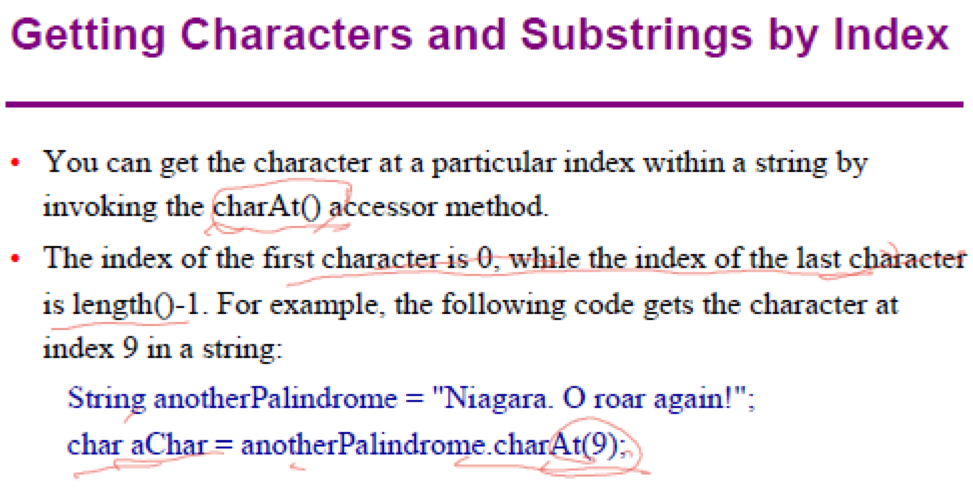
```
// The valueOf class method.
String s2 = String.valueOf(i);
```

```
int i;
double d;
String s3 = Integer.toString(i);
String s4 = Double.toString(d);
```

```
public class ToStringDemo {
   public static void main(String[] args) {
      double d = 858.48;
      String s = Double.toString(d);
      int dot = s.indexOf('.');
      System.out.println(dot + " digits " +
         "before decimal point.");
      System.out.println( (s.length() - dot - 1) +
         " digits after decimal point.");
   }
}
```

字符串对象
不是基本数据!

The output of this program is:
3 digits before decimal point.
2 digits after decimal point.


Getting Characters and Substrings by Index

- You can get the character at a particular index within a string by invoking the charAt() accessor method.
- The index of the first character is 0, while the index of the last character is length()-1. For example, the following code gets the character at index 9 in a string:

```
String anotherPalindrome = "Niagara. O roar again!";
char aChar = anotherPalindrome.charAt(9);
```

regionMatches() 方法用于检测两个字符串在一个区域内是否相等。

语法

```
public boolean regionMatches(int toffset,
                             String other,
                             int ooffset,
                             int len)

或

public boolean regionMatches(boolean ignoreCase,
                             int toffset,
                             String other,
                             int ooffset,
                             int len)
```

参数

- **ignoreCase** -- 如果为 true，则比较字符时忽略大小写。
- **toffset** -- 此字符串中子区域的起始偏移量。
- **other** -- 字符串参数。
- **ooffset** -- 字符串参数中子区域的起始偏移量。
- **len** -- 要比较的字符数。

返回值

如果字符串的指定子区域匹配字符串参数的指定子区域，则返回 true；否则返回 false。是否完全匹配或考虑大小写取决于 ignoreCase 参数。

实例

```
public class Test {
    public static void main(String args[]) {
        String Str1 = new String("www.runoob.com");
        String Str2 = new String("runoob");
        String Str3 = new String("RUNOOB");
        System.out.print("返回值 :" );
        System.out.println(Str1.regionMatches(4, Str2, 0, 5));

        System.out.print("返回值 :" );
        System.out.println(Str1.regionMatches(4, Str3, 0, 5));

        System.out.print("返回值 :" );
        System.out.println(Str1.regionMatches(true, 4, Str3, 0, 5));
    }
}
```

以上程序执行结果为：

```
返回值 :true
返回值 :false
返回值 :true
```

一个很常见的问题：问一个字符串中某个子串出现了多少次？就可以使用上面的方法解决，具体代码示例如下：

```
public class RegionMatcher {

    public static void main(String[] args) {
        int number = 0;
        String str = "fdafdadfadf";

        for (int i = 0; i < str.length(); i++) {

            if (str.regionMatches(i, "da", 0, 2)) {
                number++;
            }
        }
        System.out.println(number);
    }
}
```

上面例子计算出了在字符串str=“fdafdadfadf”中“da”出现的次数。

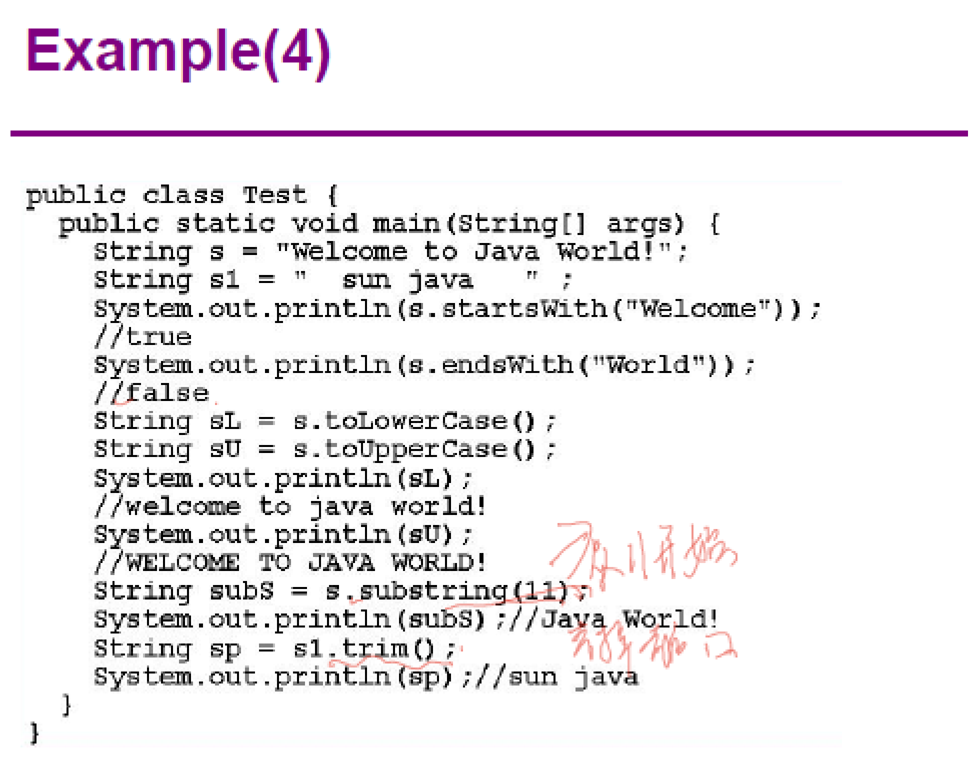
Example(2)

```
public class Test {
  public static void main(String[] args) {
    String s1 = "hello"; String s2 = "world";
    String s3 = "hello";
    System.out.println(s1 == s3); //true

    s1 = new String ("hello");
    s2 = new String("hello");
    System.out.println(s1 == s2); //false
    System.out.println(s1.equals(s2)); //true

    char c[]= {'s','u','n',' ','j','a','v','a'};
    String s4 = new String(c);
    String s5 = new String(c,4,4);
    System.out.println(s4); //sun java
    System.out.println(s5); //java
  }
}
```

Example(3)

```
public class Test {
  public static void main(String[] args) {
    String s1 = "sun java",s2 = "Sun Java";
    System.out.println(s1.charAt(1));//u
    System.out.println(s2.length());//8
    System.out.println(s1.indexOf("java"));//4
    System.out.println(s1.indexOf("Java"));//-1
    System.out.println(s1.equals(s2));//false
    System.out.println(s1.equalsIgnoreCase(s2));
    //true

    String s = "我是程序员，我在学java";
    String sr = s.replace('我','你');
    System.out.println(sr);
    //你是程序员，你在学java
  }
}
```

old. new

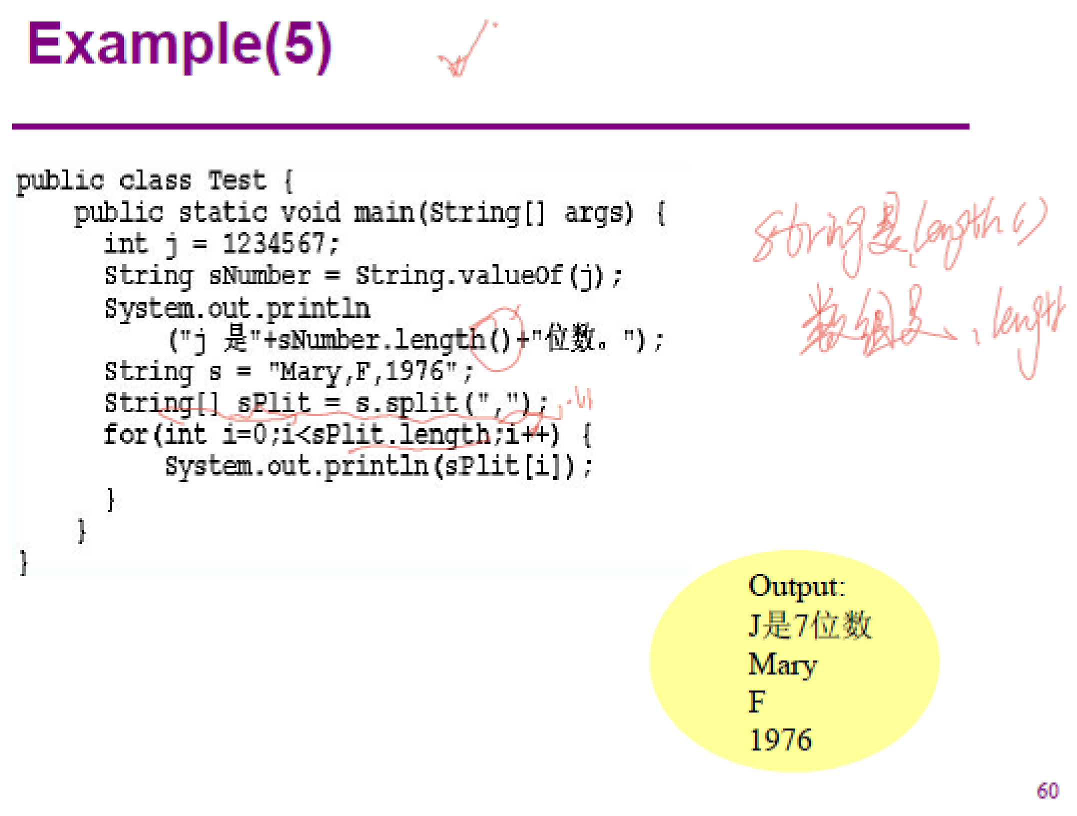


Example(4)

```
public class Test {
  public static void main(String[] args) {
    String s = "Welcome to Java World!";
    String s1 = "  sun java    " ;
    System.out.println(s.startsWith("Welcome"));
    //true
    System.out.println(s.endsWith("World"));
    //false
    String sL = s.toLowerCase();
    String sU = s.toUpperCase();
    System.out.println(sL);
    //welcome to java world!
    System.out.println(sU);
    //WELCOME TO JAVA WORLD!
    String subS = s.substring(11);
    System.out.println(subS);//Java World!
    String sp = s1.trim();
    System.out.println(sp);//sun java
  }
}
```

从11开始

去掉两端空白



Example(5) ✓

```
public class Test {
    public static void main(String[] args) {
      int j = 1234567;
      String sNumber = String.valueOf(j);
      System.out.println
          ("j 是"+sNumber.length()+"位数。");
      String s = "Mary,F,1976";
      String[] sPlit = s.split(",");
      for(int i=0;i<sPlit.length;i++) {
          System.out.println(sPlit[i]);
      }
    }
}
```

String是length()

数组是→，length

```
Output:
J是7位数
Mary
F
1976
```

StringBuilder

- StringBuilder objects are like String objects, except that they can be modified.
- Internally, these objects are treated like variable-length arrays that contain a sequence of characters.

Constructor	Description
StringBuilder()	Creates an empty string builder with a capacity of 16 (16 empty elements).
StringBuilder(CharSequence cs)	Constructs a string builder containing the same characters as the specified CharSequence, plus an extra 16 empty elements trailing the CharSequence.
StringBuilder(int initCapacity)	Creates an empty string builder with the specified initial capacity.
StringBuilder(String s)	Creates a string builder whose value is initialized by the specified string, plus an extra 16 empty elements trailing the string.

Example

```
public class StringBuilderDemo {
   public static void main(String[] args) {
      String palindrome = "Dot saw I was Tod";
      StringBuilder sb = new StringBuilder(palindrome);
      sb.reverse();  // reverse it
      System.out.println(sb);
   }
}
```

StringBuffer()

- There is also a StringBuffer class that is exactly the same as the StringBuilder class, except that it is thread-safe by virtue of having its methods synchronized.

Summary of Characters and Strings

> • Most of the time, if you are using a single character value, you will use the primitive char type.
> • There are times, however, when you need to use a char as an object—for example, as a method argument where an object is expected.
> • The Java programming language provides a wrapper class that "wraps" the char in a Character object for this purpose.
> • An object of type Character contains a single field whose type is char.
> • This Character class also offers a number of useful class (i.e., static) methods for manipulating characters.
>
> • Strings are a sequence of characters and are widely used in Java programming.
> • In the Java programming language, strings are objects.
> • The String class has over 60 methods and 13 constructors.
> • Most commonly, you create a string with a statement like

> String s = "Hello world!";
> rather than using one of the String constructors.
> • The String class has many methods to find and retrieve substrings; these can then be easily reassembled into new strings using the + concatenation operator.
>
> • The String class also includes a number of utility methods, among them split(), toLowerCase(), toUpperCase(), and valueOf().
> • The latter method is indispensable in converting user input strings to numbers.
> • The Number subclasses also have methods for converting strings to numbers and vice versa.
>
> • In addition to the String class, there is also a StringBuilder class. Working with StringBuilder objects can sometimes be more efficient than working with strings.
> • The StringBuilder class offers a few methods that can be useful for strings, among them reverse().
> • In general, however, the String class has a wider variety of methods.
> • A string can be converted to a string builder using a StringBuilder constructor.
> • A string builder can be converted to a string with the toString() method.

自动装箱和拆箱Autoboxing and Unboxing

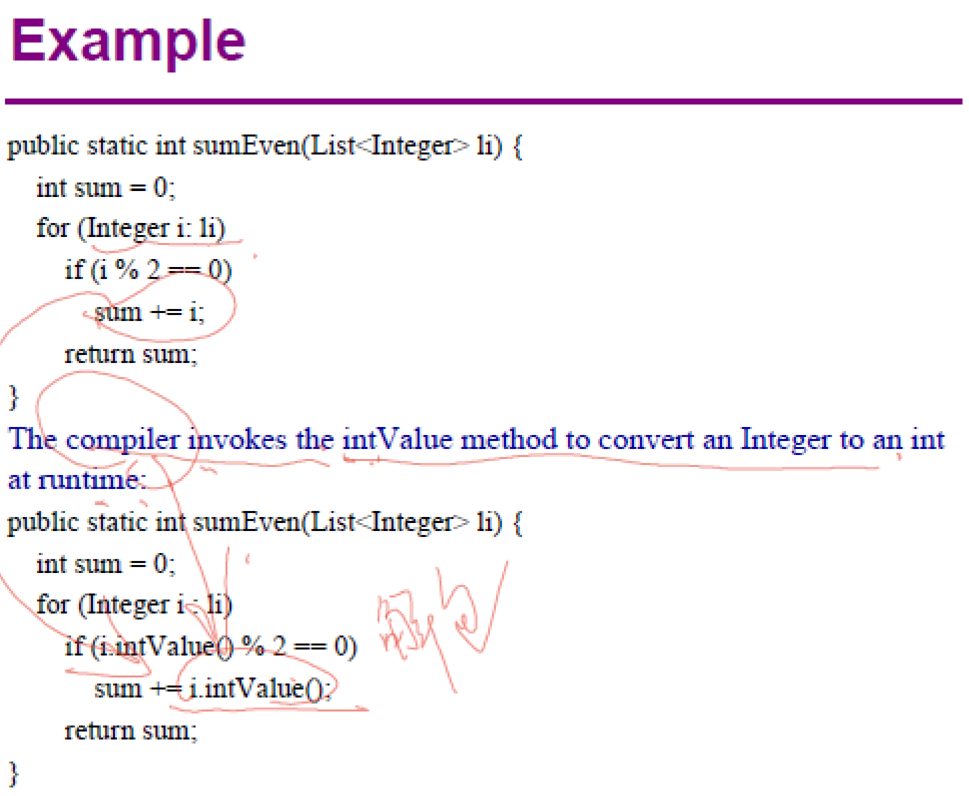
• Autoboxing is the automatic conversion that the Java compiler makes between the primitive types and their corresponding object wrapper classes. For example, converting an int to an Integer, a double to a Double, and so on.
• If the conversion goes the other way, this is called unboxing.
• Here is the simplest example of autoboxing:
Character ch = 'a';

The Java compiler applies autoboxing when a primitive value is:
– Passed as a parameter to a method that expects an object of the corresponding wrapper class.
– Assigned to a variable of the corresponding wrapper class.

当原始值为下面时Java编译器应用自动装箱:

-作为参数传递给需要相应包装类对象的方法。

-分配给相应包装类的一个变量。



Example

```
public static int sumEven(List<Integer> li) {
    int sum = 0;
    for (Integer i: li)
        if (i % 2 == 0)
            sum += i;
        return sum;
}
```

The compiler invokes the intValue method to convert an Integer to an int at runtime:

```
public static int sumEven(List<Integer> li) {
    int sum = 0;
    for (Integer i : li)
        if (i.intValue() % 2 == 0)
            sum += i.intValue();
        return sum;
}
```

[illegible]

Unboxing
• Converting an object of a wrapper type (Integer) to its corresponding primitive (int) value is called unboxing.

The Java compiler applies unboxing when an object of a wrapper class is:
– Passed as a parameter to a method that expects a value of the corresponding primitive type.
– Assigned to a variable of the corresponding primitive type.

拆箱条件 当包装类的对象为:

-作为参数传递给需要相应基元类型值的方法。

-赋给相应原语类型的变量。

Example

[illegible]

```
import java.util.ArrayList;
import java.util.List;
public class Unboxing {
   public static void main(String[] args) {
      Integer i = new Integer(-8);
      // 1. Unboxing through method invocation
      int absVal = absoluteValue(i);
      System.out.println("absolute value of " + i + " = " + absVal);
      List<Double> ld = new ArrayList<>();
      ld.add(3.1416);    // Π is autoboxed through method invocation.
      // 2. Unboxing through assignment
      double pi = ld.get(0);
      System.out.println("pi = " + pi);
   }
```

```
Output:
absolute value of -8 = 8
pi = 3.1416
```

8

Autoboxing and unboxing

p.23
end

Primitive type	Wrapper class
boolean	Boolean
byte	Byte
char	Character
float	Float
int	Integer
long	Long
short	Short
double	Double