目录

Day10. Java

1 回顾

- 继承
  - 作用：代码重用、复用
  - 单继承
    - 只能继承一个父类
  - 子类对象
    - 父类对象和子类对象绑定，整体作为一个对象
    - 调用成员，先找子类，再找父类
  - 重写
  - super
    - super.xxxx()
      - 重写时，调用父类同一个方法的代码
    - super(....)
      - 调用父类构造方法
      - 默认 super()
      - 手动 super(参数)
- 多态
  - 作用：一致的类型
    所有子类型对象，都可以被当做一致的父类型来处理

    - 向上转型
    - 向下转型
    - instanceof
        - 运行期类型识别
        - 对真实类型，和父类型，都返回 true
- 抽象类
    - 半成品
    - 抽象方法：
        - 作为通用方法，在父类中定义
        - 要求子类必须实现
- final
    - 常量
    - 方法，不能重写
    - 类，没有子类
- static
    - 静态，属于类，而不属于对象
    - 共享的数据、工具方法
    - static {
      }
- 访问控制符
    - public,protected,[default],private

- 对象的创建过程（10步）
- 接口
    - 作用：结构设计工具，解耦合，隔离实现
    - interface
    - implements
    - 类继承接口，可以多继承
    - 接口之间继承，用extends
- 内部类
    - 非静态内部类
        - 对一个复杂对象的局部数据、局部运算进行封装
    - 静态内部类
        - 和普通的类没区别
    - 局部内部类
        - 外面不能使用局部的类型
        - 对象可以作为父类型向外传递
    - 匿名内部类
      Weapon a = new Weapon() {...};

2 字符串

- 封装 char[] 数组
- 字符串常量池
    - 第一次使用一个字面值，在常量池新建对象
    - 再次使用相同字面值，直接访问常量池中存在的对象
- 字符串不可变

- s1 = "aaa";
  s2 = "bbb"
  s3 = "ccc"
  s4 = s1 + s2 + s3;

- s4 = "aaa" + "bbb" + "ccc";
  编译器编译时，优化为：
  s5 = "aaabbbccc";

- 字符串的方法
  - length()
    字符串长度

  - charAt(i)
    获取指定位置的字符

  - indexOf(子串)
    查找子串，得到子串的起始位置下标
    s = "abcabcabc";
    i = s.indexOf("bc")

    如果找不到，返回 -1

  - indexOf(子串，起始位置)
    从起始位置向后找
    i = s.indexOf("bc", 2)

  - lastIndexOf(子串)
    从后向前找

  - substring(start)
    截取子串，截取start到末尾

  - substring(start, end)
    截取 [start, end)

  - trim()
    去除两端的空白字符

字符串

项目：day1001_字符串
类： day1001.Test1

Test1

```
package day1001;

import java.util.Scanner;

public class Test1 {
```

```
    public static void main(String[] args) {
        /*
         * email:
         * abc@def.com
         * 提取名字部分 abc
         */
        System.out.print("email:");
        String e = new Scanner(System.in).nextLine();

        String name = getName(e);
        System.out.println(name);
    }

    private static String getName(String e) {
        /*
         * *) 在参数e中，查找"@"的下标位置，存到变量i
         * *) 如果没有"@"符号，返回"格式错误"
         *
         * *) 从 e 截取 [0, i) 部分子串，返回
         */

        e = e.trim();

        int i = e.indexOf("@");
        if(i == -1) {
            return "格式错误";
        }

        return e.substring(0, i);
    }
}
```

Test2

```
package day1001;

import java.util.Scanner;

public class Test2 {
    public static void main(String[] args) {
        /*
         * - 判断一个字符串是否是“回文”
         * - abcdedcba
         */
        System.out.println("输入回文：");
        String s = new Scanner(System.in).nextLine();

        if(huiWen(s)) {
            System.out.println("是回文");
        } else {
            System.out.println("不是回文");
        }
```

```
    }

    private static boolean huiWen(String s) {
        /*
         *   abcdedcba
         *   i       j
         *
         *   *) 循环
         *      i=0,j=s.length()-1; i<j; i++,j--
         *
         *      *) 如果i位置字符与j位置字符不相等
         *          *) 返回 false
         *
         *   *) 循环结束后，返回true
         */
        for(int i=0,j=s.length()-1; i<j; i++,j--) {
            if(s.charAt(i) != s.charAt(j)) {
                return false;
            }
        }

        return true;
    }
}
```

3 StringBuilder / StringBuffer

- 可变的字符序列
- 封装 char[] 数组的对象
- 提供一组对内部字符修改的方法
- 常用来代替字符串，做高效率的字符串连接运算
  - append(...) 追加
  - 内部数组的初始容量 16
  - 放满后，创建翻倍容量的新数组
- StringBuiler 和 StringBuffer
  - 功能、作用完全相同
  - StringBuffer 是一个旧版本的类
  - StringBuilder 线程不安全，效率高
  - StringBuffer 线程安全

StringBuilder

项目：day1002_StringBuilder

类: day1002.Test1

```
package day1002;

public class Test1 {
   public static void main(String[] args) {
      String s0 = "abcdefghijklmnopqrstuvwxyz";
      StringBuilder sb = new StringBuilder();
      long t = System.currentTimeMillis();
      for(int i=0;i<100000;i++) {
         sb.append(s0);
      }
      t = System.currentTimeMillis()-t;
      System.out.println(t);
   }
}
```

猜字母

day0703_猜游戏
复制
day1003_猜游戏

```
package day1003;

import java.util.Random;

public class GuessLetterGame extends GuessGame {
   @Override
   public String suiJi() {
      /*                    j
       * CWKSEFGHIJDLMNOPQRATUVBXYZ
       *    i
       *
       *    t = A
       */
                                    StringBuilder      sb      =      new
StringBuilder("ABCDEFGHIJKLMNOPQRSTUVWXYZ");
      for(int i=0;i<5;i++) {
         int j = new Random().nextInt(26);
         char t = sb.charAt(i);
         sb.setCharAt(i, sb.charAt(j));
         sb.setCharAt(j, t);
      }

      sb.delete(5, 26);
      System.out.println(sb);
      return sb.toString();
   }
```

```
    @Override
    public void tiShi() {
        System.out.println("已经产生了5个不重复的大写字母");
        System.out.println("请猜这5个字母");
    }

    @Override
    public String biJiao(String c, String r) {
        /*
         *              a=0, b=0
         * r  KQCTH
         *      j
         *
         * c  QPCTN
         *      i
         */
        //两个计数变量
        int a = 0;
        int b = 0;
        for(int i=0;i<5;i++) { //遍历c字符串
            for(int j=0;j<5;j++) {//遍历r字符串
                //比较i位置字符和j位置字符
                if(c.charAt(i) == r.charAt(j)) {
                    if(i==j) {//比较i和j下标
                        a++;
                    } else {
                        b++;
                    }
                    break;
                }
            }
        }
        return a+"A"+b+"B";

    }



    @Override
    public boolean caiDui(String result) {
        return "5A0B".equals(result);
    }
}
```

4 正则表达式 Regex

- Regular Expression

规则表达式

- 正确的字符串格式规则
- 一般用来判断用户输入内容，是否符合格式要求
- 百度“正则表达式大全”

正则表达式	匹配的字符串
k	k
abc	abc
[abc]	a,b,c
[abc][123]	a1,a2,a3,b1,b2, b3,c1,c2,c3
[a-z]	a,z,t,h,o
[a-zA-Z_0-9]	g,K,9,3,_
[^a-zA-Z]	9,&,*
[\u4e00-\u9fa5]	中文
\d	数字[0-9]
\D	排除数字[^0-9]
\w	单词字符[a-zA-Z_0-9]
\W	排除单词字符
\s	空白字符
\S	排除空白字符
.	任意字符
[abc]?	? 0个或1个 a,b,,c
[abc]?[123]	a1,b2,c3,2,3,1
[abc]*	* 0到多个 a,bb,abcc,,ccbbcaaccbca
[abc]+	+ 1到多个，至少1个 a,ab,abc,abcccbbaacb
[abc]{3}	3个 abc,cba,ccc,bac,cca,bbb
[abc]{3,5}	3到5个 abc,abca,abcab
[abc]{3,}	3到多个，至少3个 abc,abca,abccbccaacbba
^	字符串头部
^abc.*	abc,abcrw,abc*^%&,abc834
$	字符串尾部
.*x$	wewx,%*%*%x,6858x
|	或

4.1 String 的正则表达式匹配运算方法

- matches(正则)
  判断字符串能否匹配指定的正则表达式

- replaceAll(正则, 子串)

用新的子串，替换所有匹配的子串

- split(正则)
  用匹配的分隔字符，拆分字符串
  "aaa,bbb,ccc"

  ["aaa", "bbb", "ccc"]

正则表达式

项目：day1004_正则表达式
类：day1004.Test1

```
package day1004;

import java.util.Scanner;

public class Test1 {
   public static void main(String[] args) {
      /*
       * "\n" 换行
       * "\t" 制表符
       * "\\" 一个斜杠字符 \
       * \ 是java的转移运算符
       */
      //System.out.println("\\\\\\\\\\\\\\\\\\\\");
      //System.out.println("\\d");

      System.out.println("输入身份证号：");
      String id = new Scanner(System.in).nextLine();
      System.out.println("输入固定电话：");
      String tel = new Scanner(System.in).nextLine();
      if(f1(id)) {
         System.out.println("身份证号格式正确");
      } else {
         System.out.println("身份证号格式错误");
      }
      if(f2(tel)) {
         System.out.println("电话号码格式正确");
      } else {
         System.out.println("电话号码格式错误");
      }
   }

   private static boolean f1(String id) {
      /*
       * 123456789012345
       * 123456789012345678
       * 12345678901234567x
       * 12345678901234567X
       *
```

```
         * |
         * \d{15}|
         * \d{15}|\d{17}
         * \d{15}|\d{17}[\dxX]
         *
         * \\d{15}|\\d{17}[\\dxX]
         */
        String regex = "\\d{15}|\\d{17}[\\dxX]";
        //参数id，能否匹配正则表达式 regex
        return id.matches(regex);
    }

    private static boolean f2(String tel) {
        /*
         * 1234567
         * 12345678
         * 010-1234567
         * 0102-1234567
         * (010)12345678
         * (0102)12345678
         *
         * \d{7,8}
         * (区号格式)?\d{7,8}
         * (|)?\d{7,8}
         * (\d{3,4}-|)?\d{7,8}
         * (\d{3,4}-|())?\d{7,8}
         * (\d{3,4}-|\(\))?\d{7,8}
         * (\d{3,4}-|\(\d{3,4}\))?\d{7,8}
         *
         * (\\d{3,4}-|\\(\\d{3,4}\\))?\\d{7,8}
         */
        String regex = "(\\d{3,4}-|\\(\\d{3,4}\\))?\\d{7,8}";
        return tel.matches(regex);
    }
}
```

5 基本类型的包装类

- 有时需要把基本类型，当做引用类型来使用

```
void f(Object obj) {

}

f(new Integer(5))
```

byte	Byte

short	Short
int	Integer
long	Long
float	Float
double	Double
char	Character
boolean	Boolean

5.1 Number 类

- 数字类型的抽象父类
- 子类:
  - Byte,Short,Integer,Long
  - Float,Double
  - BigDecimal,BigInteger
- 取值方法:
  - byteValue()
  - shortValue()
  - intValue()
  - longValue()
  - floatValue()
  - doubleValue()

5.2 Integer

- 创建对象
  - new Integer(5)
  - Integer.valueOf(5)
    Integer类中，存在256个Integer的缓存对象，范围[-128,127]
    如果指定范围内的值，访问缓存的对象，而不新建
    如果指定范围外的值，直接新建对象

- 方法
  - 字符串解析成 int
    ```
    Integer.parseInt("255")  255
    Integer.parseInt("11111111", 2) 255
    Integer.parseInt("377", 8)  255
    Integer.parseInt("ff", 16)  255
    ```
  - 进制转换
    ```
    Integer.toBinaryString(255) "11111111"
    Integer.toOctalString(255)  "377"
    Integer.toHexString(255) "ff"
    ```

包装类

项目：day1005_包装类
类：day1005.Test1

```
package day1005;

public class Test1 {
   public static void main(String[] args) {
      Integer a = new Integer(5);//新分配内存
      Integer b = Integer.valueOf(5);//访问缓存对象
      Integer c = Integer.valueOf(5);//访问缓存对象

      System.out.println(a==b);
      System.out.println(b==c);
      System.out.println(a.equals(b));

      System.out.println(a.byteValue());
      System.out.println(a.shortValue());
      System.out.println(a.intValue());
      System.out.println(a.longValue());
      System.out.println(a.floatValue());
      System.out.println(a.doubleValue());

      System.out.println(Integer.parseInt("255"));
      System.out.println(Integer.parseInt("11111111", 2));
      System.out.println(Integer.parseInt("377", 8));
      System.out.println(Integer.parseInt("ff", 16));
      System.out.println(Integer.toBinaryString(255));
      System.out.println(Integer.toOctalString(255));
      System.out.println(Integer.toHexString(255));
   }
}
```

5.3 Double

- 创建对象
  - new Double(3.14)
  - Double.valueOf(3.14)
    与 new 相同

- 方法
  - 字符串解析成 double
    Double.parseDouble("3.14")

```
Byte.parseByte()
Short.parseShort()
Integer.parseInt()
Long.parseLong()
Float.parseFloat()
Double.parseDouble()
Boolean.parseBoolean()
```

- 对浮点数特殊值进行判断
  - Infinity 3.14/0

    Double.isInfinite(double)

  - NaN Math.sqrt(-2)

    Double.isNaN(double)

5.4 自动装箱和自动拆箱

- 自动装箱
  基本类型值，自动封装成对象

  ```
  Integer a = 5;
  编译器编译成:
  Integer a = Integer.valueOf(5);
  ```

- 自动拆箱
  从对象中，自动取出封装的基本类型值

  ```
  int i = a;
  编译器编译成:
  int i = a.intValue();

  a = a + 1;
  编译器编译成:
  a = Integer.valueOf(a.intValue() + 1);
  ```

  - 自动拆箱，要当心 null 值

6 BigDecimal/BigInteger

- BigDecimal 精确地浮点数运算
- BigInteger 超大的整数运算

- 创建对象
  - BigDecimal.valueOf(2)
- 方法
  - add(BigDecimal bd)
  - subtract(BigDecimal bd)
  - multiply(BigDecimal bd)
  - divide(BigDecimal bd)
  - divide(BigDecimal bd,保留位数,舍入方式)
  - setScale(保留位数,舍入方式) 舍入运算

BigDecimal

项目：day1006_BigDecimal
类：day1006.Test1

```
package day1006;

import java.math.BigDecimal;
import java.util.Scanner;

public class Test1 {
   public static void main(String[] args) {
      System.out.println("输入两个浮点数：");
      double a = new Scanner(System.in).nextDouble();
      double b = new Scanner(System.in).nextDouble();
      System.out.println(a+b);
      System.out.println(a-b);
      System.out.println(a*b);
      System.out.println(a/b);
      /*
       * 2      4.35
       * 1.9    100
       */
      System.out.println("----------------");
      BigDecimal bd1 = BigDecimal.valueOf(a);
      BigDecimal bd2 = BigDecimal.valueOf(b);
      BigDecimal bd3;//用来保存计算结果

      bd3 = bd1.add(bd2);
      System.out.println(bd3.doubleValue());

      bd3 = bd1.subtract(bd2);
      System.out.println(bd3.doubleValue());

      bd3 = bd1.multiply(bd2);
      System.out.println(bd3.doubleValue());

      bd3 = bd1.divide(bd2, 30, BigDecimal.ROUND_HALF_UP);
      System.out.println(bd3.doubleValue());
      System.out.println(bd3.toString());

      bd3 = bd3.setScale(2, BigDecimal.ROUND_HALF_UP);
      System.out.println(bd3.doubleValue());


   }
}
```

自由落体距离

day0203_自由落体距离
复制
day1007_自由落体距离

```
package day0203;

import java.math.BigDecimal;
import java.util.Scanner;

public class Test1 {
   public static void main(String[] args) {
      System.out.print("输入降落时间（秒）： ");
      //先获得输入的 double 值
      //再存到变量t
      double t = new Scanner(System.in).nextDouble();

      //求降落距离,
      //存到变量 d
      //double d = 0.5 * 9.8 * t * t;

      BigDecimal bd1 = BigDecimal.valueOf(4.9);
      BigDecimal bd2 = BigDecimal.valueOf(t);

      double d =
         bd1.multiply(bd2.pow(2))
         .setScale(3, BigDecimal.ROUND_HALF_UP)
         .doubleValue();


      System.out.println(t+"秒降落了"+d+"米");
   }
}
```

阶乘

day0303_阶乘
复制
day1008_阶乘

```
package day0303;

import java.math.BigInteger;
import java.util.Scanner;

public class Test1 {
   public static void main(String[] args) {
      System.out.println("输入整数求阶乘: ");
```

```
        int n = new Scanner(System.in).nextInt();

        //把n的值，传递到f()方法求它的阶乘
        f(n);
    }

    static void f(int n) {
        /*
         * r = 5
         * i
         * 4, r=r*i
         * 3, r=r*i
         * 2, r=r*i
         * 1, r=r*i
         */
        //long r = n;
        //for(int i=n-1; i>=1; i--) {
        // r *= i;
        //}

        BigInteger r = BigInteger.valueOf(n);
        for(int i=n-1; i>=1; i--) {
            r = r.multiply(BigInteger.valueOf(i));
        }

        System.out.println(r.toString());
    }
}
```

- String
  - charAt(), indexOf(), substring(), trim()
- StringBuilder,StringBuffer
  - append()
- 正则
  - s.matches(正则)
- 包装类
  - Integer.valueOf()
    缓存对象 -128到127
- BigDecimal,BigInteger

7 作业

- 复习基础API
- 查找子串出现的所有位置

  输入字符串：abcabcabc

输入查找的子串：bc

```
1
4
7

aaaaaa
aaaa
0
1
2


aaaaaa
a
0
1
2
3
4
5

abcabcabc
|
index

s.indexOf(子串, index)
```