* 学习目标

* 能够掌握Properties的使用

* 底层HashTable

* 属性：key-value（String，String）

* 一组持久化的属性，它可以被保存到流或者从流中获取

* setProperty，getProperty,load(InputStream,Reader),store(OutputStream,String)

* stringPropertyNames(keySet())

* 路径的：InputStream is=Test4.class.getClassLoader().getResourceAsStream(String)

* System.getProperties();System.getProperty

* java.io.tmpdir:缓冲存放路径

* 清除缓存

* 能够掌握装饰者模式

* 开闭原则：对修改进行关闭，进行扩展进行开放

* 继承

* 装饰者模式

* BufferedReader br =new BufferedReader(new FileReader("abc.txt"));

* 能够使用缓冲流读写数据到程序

* BufferedInputStream BufferedOutputStream

* BufferedReader BufferedWriter

* 默认缓存数组的大小：8k

* BufferedInputStream bis=new BufferedInputStream (new FileInputStream("xx.avi"));

BufferedOutputStream bos=new BufferedOutputStream(new FileOutputStream("xx.avi"));

int len=0;

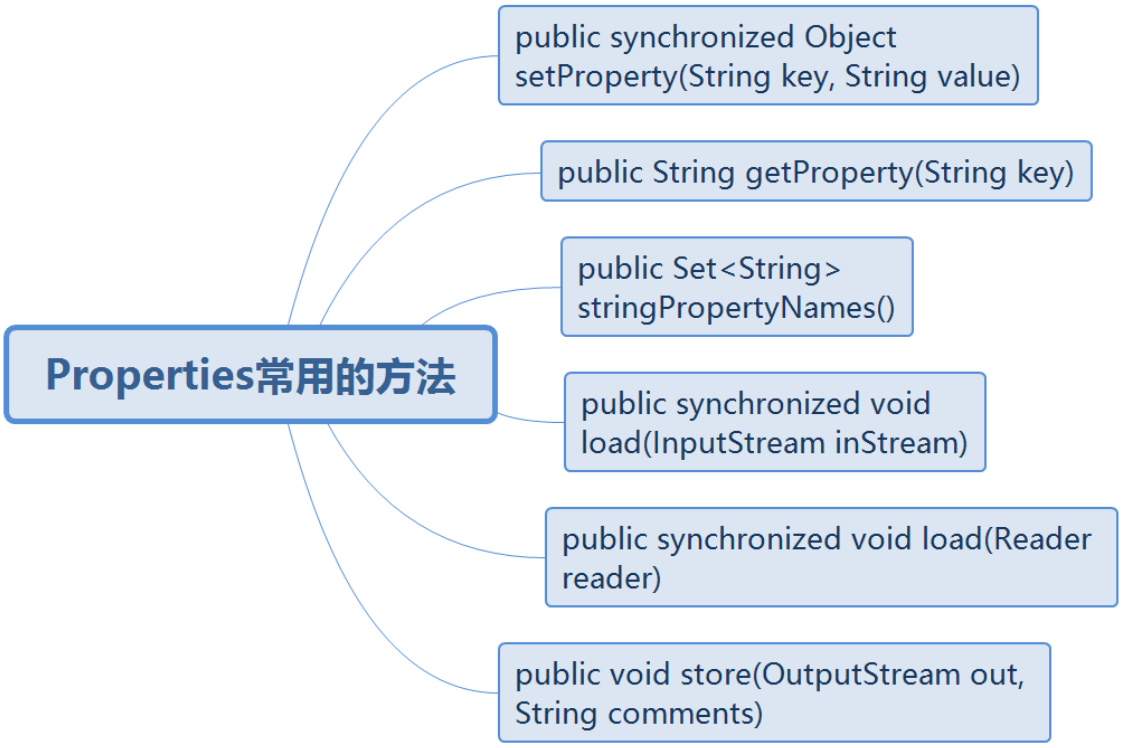
byte[] b=new byte[1024];

while((len=bis.read(b))!=-1){

```
            bos.write(b,0,len);
        }
```

---

* 回顾
  * IO流的继承架构
    * InputStream
      * FileInputStream，ByteArrayInputStream，ObjectInputStream，SequenceInputStream，BufferedInputStream
    * OutputStream
      * FileOutputStream，ByteArrayOutputStream，ObjectOutputStream
      PrintStream,BufferedOutputStream
    * Reader
      * BufferedReader.StringReader,InputStreamReader,FileReader
    * Writer
      * BufferedWriter,StringWriter,InputStreamWriter,FileWriter,PrintWriter
  * InputStream,OutputStream,Reader,Writer
    * File
  * IO的异常处理
    * JDK1.7之前
      try{}catcfh()finally{...}
    * JDK1.7
      * AutoClosable
      * try(...;...){...}catch()
* 能够掌握Properties的使用
  * Properties 表示一组持久性的属性，可以保存到流或从流加载。
  * Propperties继承HashTable
  * Properties常用的方法




```
* Properties
public static void main(String[] args) throws IOException {
        // 构建Properties 属性
        Properties prop=new Properties();
        // 1 Hashtable的用法
        prop.setProperty("name", "xiaohei");
        prop.setProperty("age", "18");
        String name=prop.getProperty("name");
        String age=prop.getProperty("age");
        System.out.println(name+":"+age);
        System.out.println("----------------------");
        Set<String> keys = prop.stringPropertyNames();
        for(String key:keys) {
            System.out.println(key+":"+prop.get(key));
        }
        System.out.println("----------------------");
        // 2 从Properties文件获取
        Properties prop1=new Properties();
        // 演示导出jar之后，他们的不同
        // 导出到桌面：由于桌面没有src目录
```

```
        // java.io.FileNotFoundException: .\src\application.properties (系统找不
//      prop1.load(new FileInputStream("./src/application.properties"));
//      prop1.load(new FileReader("./src/application.properties"));
        prop1.load(Test1.class.getClassLoader().getResourceAsStream("applicatio
        keys = prop1.stringPropertyNames();
        for(String key:keys) {
            System.out.println(key+":"+prop1.get(key));
        }

        System.out.println("用户名: "+prop1.getProperty("username"));

    }
 * 结果:
 xiaohei:18
-----------------------
age:18
name:xiaohei
-----------------------
password:root
username:root
用户名: root
 * Properties 保存到流中
 public static void main(String[] args) throws IOException {
        Properties prop=new Properties();
        prop.setProperty("username", "root");
        prop.setProperty("password", "root");
        System.out.println(prop);
        // 属性保存到流中
        prop.store(System.out, "lg");
    }
{password=root, username=root}
#lg
#Thu Sep 05 16:18:03 CST 2019
password=root
username=root

 * 获取系统参数
 public static void main(String[] args) {
        // 获取系统的变量
        Properties properties = System.getProperties();
```

```
        Set<Object> keys = properties.keySet();
        for(Object key:keys) {
            System.out.println(key+"---------"+properties.getProperty((String)
        }

        System.out.println("---------------------------------");
        System.out.println(properties.get("java.io.tmpdir"));
    }
结果：
java.runtime.name---------Java(TM) SE Runtime Environment
sun.boot.library.path---------C:\Program Files\Java\jre1.8.0_181\bin
java.vm.version---------25.181-b13
java.vm.vendor---------Oracle Corporation
java.vendor.url---------http://java.oracle.com/
path.separator---------;
java.vm.name---------Java HotSpot(TM) 64-Bit Server VM
file.encoding.pkg---------sun.io
user.country---------CN
user.script---------
sun.java.launcher---------SUN_STANDARD
sun.os.patch.level---------
java.vm.specification.name---------Java Virtual Machine Specification
user.dir---------E:\ws0722\e26
java.runtime.version---------1.8.0_181-b13
java.awt.graphicsenv---------sun.awt.Win32GraphicsEnvironment
java.endorsed.dirs---------C:\Program Files\Java\jre1.8.0_181\lib\endorsed
os.arch---------amd64
java.io.tmpdir---------C:\Users\xiaozhao\AppData\Local\Temp\
line.separator---------

java.vm.specification.vendor---------Oracle Corporation
user.variant---------
os.name---------Windows 10
sun.jnu.encoding---------GBK
java.library.path---------C:\Program Files\Java\jre1.8.0_181\bin;C:\windows\Sur
java.specification.name---------Java Platform API Specification
java.class.version---------52.0
sun.management.compiler---------HotSpot 64-Bit Tiered Compilers
os.version---------10.0
user.home---------C:\Users\xiaozhao
```

```
user.timezone---------
java.awt.printerjob---------sun.awt.windows.WPrinterJob
file.encoding---------GBK
java.specification.version---------1.8
java.class.path---------C:\Program Files\Java\jre1.8.0_181\lib\resources.jar;C:
user.name---------xiaozhao
java.vm.specification.version---------1.8
sun.java.command---------com.lg.test6.Test3
java.home---------C:\Program Files\Java\jre1.8.0_181
sun.arch.data.model---------64
user.language---------zh
java.specification.vendor---------Oracle Corporation
awt.toolkit---------sun.awt.windows.WToolkit
java.vm.info---------mixed mode
java.version---------1.8.0_181
java.ext.dirs---------C:\Program Files\Java\jre1.8.0_181\lib\ext;C:\windows\Sur
sun.boot.class.path---------C:\Program Files\Java\jre1.8.0_181\lib\resources.ja
java.vendor---------Oracle Corporation
file.separator---------\
java.vendor.url.bug---------http://bugreport.sun.com/bugreport/
sun.io.unicode.encoding---------UnicodeLittle
sun.cpu.endian---------little
sun.desktop---------windows
sun.cpu.isalist---------amd64
---------------------------------
C:\Users\xiaozhao\AppData\Local\Temp\
```

* 能够掌握装饰者模式

* 设计原则(开闭原则)

```
* 设计原则：开闭原则(OCP)：应该对什么开放，对什么进行关闭？？？
    * 项目上线（银行）：架构师...
      * 添加一个广告的功能
    * 对修改进行关闭
    * 对扩展进行开放
      * 新建一个类
    * Collection--List--Set
      * ArrayList ---LinkedList  -- MLinkedList Implements List extends LinkedL
```

```
    * FileReader  ---> 增强缓冲的功能BufferedReader(FlieReader in)
```

* 装饰模式指的是在不必改变原类文件和使用继承的情况下，动态地扩展一个对象的功能。它是通过创建一个包装对象，也就是装饰来包裹真实的对象。

* 装饰设计模式的特点：装饰类与被装饰类都必须所属同一个接口或父类。

* 装饰模式比继承灵活

* 什么时候用继承，什么时候用装饰者模式？

    * 当要增强功能的时候，需要增强的功能类与原始类没有继承关系，这个时候可以用装饰者模式，例如：FileReader需要增强缓冲的功能，只需要加成装饰(Buffered)BufferedReader，FileReader与BufferedReader没有继承关系。

```
* 自定义BufferedReader
    * 加上缓冲数组
    * 添加读一行
 * 自定义BufferedReader
/**
 * @author xiaozhao 线程不安全的BufferedReader
 */
public class BufferedReader extends Reader {
    // 默认缓存区的大小
    private static final int DefaultCharBufferSize = 8192;
    // 源Reader
    private Reader in;
    // 缓冲区
    private char[] cb;
    // 缓冲区的元素进行访问下标
    private int index;
    // 计算器：缓冲区元素的计算器
    private int count;

    public BufferedReader(Reader in, int bufferSize) {
        if (bufferSize <= 0) {
            throw new IllegalArgumentException("Buffer Size <= 0");
        }
        this.in = in;
        cb = new char[bufferSize];
```

```
26      }
27
28      public BufferedReader(Reader in) {
29          this(in, DefaultCharBufferSize);
30      }
31
32      // 增强了buffer的功能
33      // 直接从buf中直接读取单个字符，而不是硬盘
34      public int read() throws IOException {
35          // 假如count等于0，从源Reader都数据到缓存区
36          if (count == 0) {
37              count = in.read(cb);
38              index = 0;
39          }
40          // 假如count 小于0，意味流到末尾了
41          if (count < 0) {
42              return -1;
43          }
44          char ch = cb[index++];
45          count--;
46          return ch;
47      }
48
49      // 增强一个一行的方法
50      public String readLine() throws IOException {
51          StringBuilder sb = new StringBuilder();
52          int c;
53          while ((c = read()) != -1) {
54              if (c == '\r') {// 在windows下回车符为\r \n 读取到\r 不处理跳出当前循环
55                  continue;
56              }
57              if (c == '\n') {
58                  return sb.toString();
59              }
60              sb.append((char) c);
61          }
62          // 最后一行
63          if (sb.toString().length() != 0) {
64              return sb.toString();
65          }
```

```
        return null;
    }

    @Override
    public int read(char[] cbuf, int off, int len) throws IOException {
        return in.read(cbuf, off, len);
    }

    @Override
    public void close() throws IOException {
        if (in == null) {
            return;
        }
        try {
            in.close();
        } finally {
            in = null;
            cb = null;
        }
    }

}


* 测试
public static void main(String[] args) throws IOException {
        FileReader fw=new FileReader("test3.txt");
        int c;
        while((c=fw.read())!=-1) {
            System.out.printf("%c",c);
        }
        fw.close();
    }
public static void main(String[] args) throws IOException {
        FileReader fw=new FileReader("test3.txt");
        BufferedReader br=new BufferedReader(fw);
        String line=null;
        while((line=br.readLine())!=null) {
            System.out.println(line);
        }
```

```
106         br.close();
107     }
108 
109  * 测试所花时间
110     public static void main(String[] args) throws IOException {
111         FileReader fw=new FileReader("ab.txt");
112         int c;
113         long start=System.currentTimeMillis();
114         while((c=fw.read())!=-1) {
115 //          System.out.printf("%c",c);
116         }
117         long end=System.currentTimeMillis();
118         System.out.println("没有缓存花的时间："+(end-start));
119         fw.close();
120 
121 
122         fw=new FileReader("ab.txt");
123         BufferedReader br=new BufferedReader(fw);
124         String line=null;
125         start=System.currentTimeMillis();
126         while((line=br.readLine())!=null) {
127 //          System.out.println(line);
128         }
129         end=System.currentTimeMillis();
130         System.out.println("有缓存花的时间："+(end-start));
131         br.close();
132 
133         fw=new FileReader("ab.txt");
134         java.io.BufferedReader br1=new java.io.BufferedReader(fw);
135         start=System.currentTimeMillis();
136         while((line=br1.readLine())!=null) {
137 //          System.out.println(line);
138         }
139         end=System.currentTimeMillis();
140         System.out.println("系统自带的有缓存花的时间："+(end-start));
141         br1.close();
142     }
143  * 结果：
144  没有缓存花的时间：431
145  有缓存花的时间：58
```

```
146  系统自带的有缓存花的时间：44
```

* 能够使用缓冲流读写数据到程序

```
1  * BufferedInputStream BufferedOutputStream
2    * 文件复制的测试有没有使用缓存流：时间进行对比
3      public static void main(String[] args)  {
4          long start=System.currentTimeMillis();
5          try(FileInputStream fis=new FileInputStream("E:\\avi\\01学习目标的介绍.w
6              FileOutputStream fos=new FileOutputStream("E:\\avi1\\01学习目标的介绍
7              int len=0;
8              while((len=fis.read())!=-1) {
9                  fos.write(len);
10             }
11         } catch (IOException e) {
12             e.printStackTrace();
13         }
14         long end=System.currentTimeMillis();
15         System.out.println("普通流复制时间:"+(end-start)+"毫秒");
16     }
17     public static void main(String[] args)  {
18         long start=System.currentTimeMillis();
19         try(BufferedInputStream bis=new BufferedInputStream(new FileInputStream
20             BufferedOutputStream bos=new BufferedOutputStream(new FileOutputStr
21             int len=0;
22             while((len=bis.read())!=-1) {
23                 bos.write(len);
24             }
25         } catch (IOException e) {
26             e.printStackTrace();
27         }
28         long end=System.currentTimeMillis();
29         System.out.println("缓存流流复制时间:"+(end-start)+"毫秒");
30     }
31   * 结果对比：
32   普通流复制时间:31266毫秒
33   缓存流流复制时间:302毫秒
```

```
  public static void main(String[] args)  {
        long start=System.currentTimeMillis();
        try(FileInputStream fis=new FileInputStream("E:\\avi\\01学习目标的介绍.w
            FileOutputStream fos=new FileOutputStream("E:\\avi1\\01学习目标的介绍
            int len=0;
            byte[] b=new byte[1024];
            while((len=fis.read(b))!=-1) {
                fos.write(b,0,len);
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
        long end=System.currentTimeMillis();
        System.out.println("普通流复制时间（一次读1kb）:"+(end-start)+"毫秒");
    }
    public static void main(String[] args)  {
        long start=System.currentTimeMillis();
        try(BufferedInputStream bis=new BufferedInputStream(new FileInputStream
            BufferedOutputStream bos=new BufferedOutputStream(new FileOutputStr
            int len=0;
            byte[] b=new byte[1024];
            while((len=bis.read(b))!=-1) {
                bos.write(b,0,len);
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
        long end=System.currentTimeMillis();
        System.out.println("缓存流复制时间(一次读1kb):"+(end-start)+"毫秒");
    }


 * 测试结果:
    缓存流复制时间(一次读1kb):15毫秒
    普通流复制时间（一次读1kb）:46毫秒

* BufferedReader BufferedWriter
  * BufferedReader
public static void main(String[] args) {
        try(BufferedReader br=new BufferedReader(new FileReader("test3.txt"))){
```

```
            String line=null;
            while((line=br.readLine())!=null) {
                System.out.println(line);
            }
        }  catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }

    }
结果:
亮哥教育
做教育，我们是认真的。
亮哥教育
做教育，我们是认真的。
* BufferedWriter
public static void main(String[] args) {
        try(BufferedWriter bw=new BufferedWriter(new FileWriter("test5.txt"))){
            bw.write("亮哥教育");
            bw.newLine();
            bw.write("做教育我们是认真的。");
        } catch (IOException e) {
            e.printStackTrace();
        }
    }


```