电子科技大学信息与软件工程学院

**实 验 报 告**

学 号 2017221105003

姓 名 黄文杰

（实验） 课程名称 面向对象程序设计Java

理论教师 江春华

实验教师 郝晓青

**电 子 科 技 大 学**

**实 验 报 告**

**学生姓名：黄文杰 学号：2017221105003 指导教师：江春华**

**实验地点：信软学院楼西306 实验时间：2018.11.19**

1. **实验名称：Java高级编程技术实验**
2. **实验学时：2学时**
3. **实验目的：**

理解Java异常处理、多线程、I/O、网络以及GUI等高级编程技术。

1. **实验原理：**

1.Java异常处理

①Java常见异常

检查性异常：最具代表的检查性异常是用户错误或问题引起的异常，这是程序员无法预见的。例如要打开一个不存在文件时，一个异常就发生了，这些异常在编译时不能被简单地忽略。

运行时异常： 运行时异常是可能被程序员避免的异常。与检查性异常相反，运行时异常可以在编译时被忽略。

错误： 错误不是异常，而是脱离程序员控制的问题。错误在代码中通常被忽略。例如，当栈溢出时，一个错误就发生了，它们在编译也检查不到的。

②常见Java异常类

|  |  |
| --- | --- |
| ArithmeticException | 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。 |
| ArrayIndexOutOfBoundsException | 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。 |
| IndexOutOfBoundsException | 指示某排序索引（例如对数组、字符串或向量的排序）超出范围时抛出。 |
| NullPointerException | 当应用程序试图在需要对象的地方使用 null 时，抛出该异常 |
| ClassNotFoundException | 应用程序试图加载类时，找不到相应的类，抛出该异常。 |
| InterruptedException | 一个线程被另一个线程中断，抛出该异常。 |
| NoSuchFieldException | 请求的变量不存在 |
| NoSuchMethodException | 请求的方法不存在 |

③捕获异常

|  |
| --- |
| try  {  // 程序代码  }catch(ExceptionName e1)  {  //Catch 块  } |

④throws/throw 关键字：

|  |
| --- |
| import java.io.\*;  public class className  {  public void withdraw(double amount) throws RemoteException,  InsufficientFundsException  {  // Method implementation  }  //Remainder of class definition  } |

2.Java多线程

①一个线程的生命周期

新建状态:

使用 new 关键字和 Thread 类或其子类建立一个线程对象后，该线程对象就处于新建状态。它保持这个状态直到程序 start() 这个线程。

就绪状态:

当线程对象调用了start()方法之后，该线程就进入就绪状态。就绪状态的线程处于就绪队列中，要等待JVM里线程调度器的调度。

运行状态:

如果就绪状态的线程获取 CPU 资源，就可以执行 run()，此时线程便处于运行状态。处于运行状态的线程最为复杂，它可以变为阻塞状态、就绪状态和死亡状态。

阻塞状态:

如果一个线程执行了sleep（睡眠）、suspend（挂起）等方法，失去所占用资源之后，该线程就从运行状态进入阻塞状态。在睡眠时间已到或获得设备资源后可以重新进入就绪状态。可以分为三种：

等待阻塞：运行状态中的线程执行 wait() 方法，使线程进入到等待阻塞状态。

同步阻塞：线程在获取 synchronized 同步锁失败(因为同步锁被其他线程占用)。

其他阻塞：通过调用线程的 sleep() 或 join() 发出了 I/O 请求时，线程就会进入到阻塞状态。当sleep() 状态超时，join() 等待线程终止或超时，或者 I/O 处理完毕，线程重新转入就绪状态。

死亡状态:

一个运行状态的线程完成任务或者其他终止条件发生时，该线程就切换到终止状态。

②Thread 方法

public void start()

使该线程开始执行；Java 虚拟机调用该线程的 run 方法。

public void run()

如果该线程是使用独立的 Runnable 运行对象构造的，则调用该 Runnable 对象的 run 方法；否则，该方法不执行任何操作并返回。

public final void setName(String name)

改变线程名称，使之与参数 name 相同。

public final void setPriority(int priority)

更改线程的优先级。

public final void setDaemon(boolean on)

将该线程标记为守护线程或用户线程。

public final void join(long millisec)

等待该线程终止的时间最长为 millis 毫秒。

public void interrupt()

中断线程。

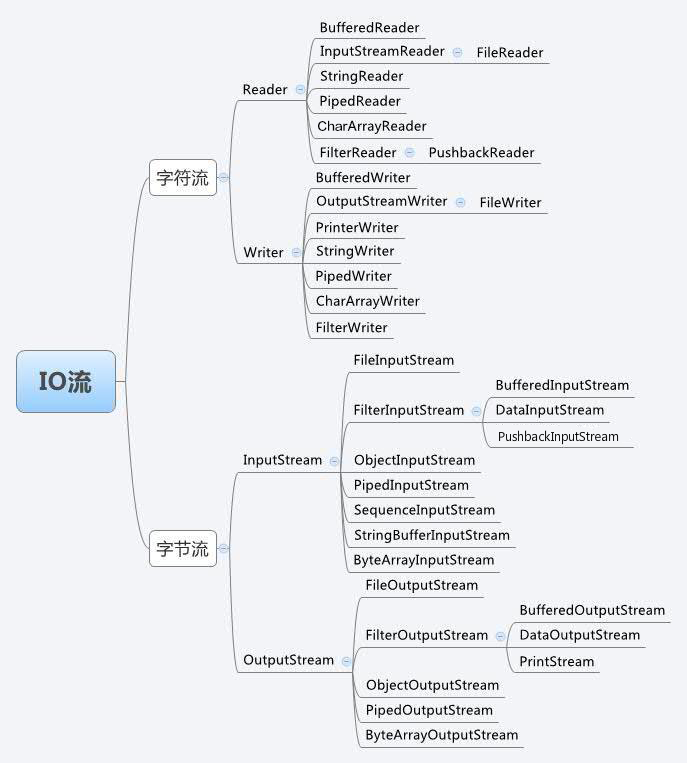
public final boolean isAlive()

测试线程是否处于活动状态。

3.Java I/O

①Java 流(Stream)、文件(File)和IO

Java.io 包几乎包含了所有操作输入、输出需要的类。所有这些流类代表了输入源和输出目标。Java.io 包中的流支持很多种格式，比如：基本类型、对象、本地化字符集等等。一个流可以理解为一个数据的序列。输入流表示从一个源读取数据，输出流表示向一个目标写数据。Java 为 I/O 提供了强大的而灵活的支持，使其更广泛地应用到文件传输和网络编程中。

②读写文件

4.Java网络编程

①常见网络协议

TCP：TCP 是传输控制协议的缩写，它保障了两个应用程序之间的可靠通信。通常用于互联网协议，被称 TCP / IP。

UDP：UDP 是用户数据报协议的缩写，一个无连接的协议。提供了应用程序之间要发送的数据的数据包。

②Socket 编程

套接字使用TCP提供了两台计算机之间的通信机制。 客户端程序创建一个套接字，并尝试连接服务器的套接字。当连接建立时，服务器会创建一个 Socket 对象。客户端和服务器现在可以通过对 Socket 对象的写入和读取来进行通信。java.net.Socket 类代表一个套接字，并且 java.net.ServerSocket 类为服务器程序提供了一种来监听客户端，并与他们建立连接的机制。

以下步骤在两台计算机之间使用套接字建立TCP连接时会出现：

服务器实例化一个 ServerSocket 对象，表示通过服务器上的端口通信。服务器调用 ServerSocket 类的 accept() 方法，该方法将一直等待，直到客户端连接到服务器上给定的端口。服务器正在等待时，一个客户端实例化一个 Socket 对象，指定服务器名称和端口号来请求连接。

Socket 类的构造函数试图将客户端连接到指定的服务器和端口号。如果通信被建立，则在客户端创建一个 Socket 对象能够与服务器进行通信。在服务器端，accept() 方法返回服务器上一个新的 socket 引用，该 socket 连接到客户端的 socket。

连接建立后，通过使用 I/O 流在进行通信，每一个socket都有一个输出流和一个输入流，客户端的输出流连接到服务器端的输入流，而客户端的输入流连接到服务器端的输出流。TCP 是一个双向的通信协议，因此数据可以通过两个数据流在同一时间发送.以下是一些类提供的一套完整的有用的方法来实现 socket。

③ServerSocket 类的方法

|  |
| --- |
| public ServerSocket(int port) throws IOException  创建绑定到特定端口的服务器套接字。 |
| public ServerSocket(int port, int backlog) throws IOException  利用指定的 backlog 创建服务器套接字并将其绑定到指定的本地端口号。 |
| public ServerSocket(int port, int backlog, InetAddress address) throws IOException  使用指定的端口、侦听 backlog 和要绑定到的本地 IP 地址创建服务器。 |
| public ServerSocket() throws IOException  创建非绑定服务器套接字。 |

④Socket 类的方法

|  |
| --- |
| public Socket(String host, int port) throws UnknownHostException, IOException.  创建一个流套接字并将其连接到指定主机上的指定端口号。 |
| public Socket(InetAddress host, int port) throws IOException  创建一个流套接字并将其连接到指定 IP 地址的指定端口号。 |
| public Socket(String host, int port, InetAddress localAddress, int localPort) throws IOException.  创建一个套接字并将其连接到指定远程主机上的指定远程端口。 |
| public Socket(InetAddress host, int port, InetAddress localAddress, int localPort) throws IOException.  创建一个套接字并将其连接到指定远程地址上的指定远程端口。 |
| public Socket()  通过系统默认类型的 SocketImpl 创建未连接套接字 |

5.Java GUI

①Applet

Applet 是一种 Java 程序。它一般运行在支持 Java 的 Web 浏览器内。因为它有完整的 Java API支持,所以Applet 是一个全功能的 Java 应用程序。

如下所示是独立的 Java 应用程序和 applet 程序之间重要的不同：

Java 中 Applet 类继承了 java.applet.Applet 类。Applet 类没有定义 main()，所以一个 Applet 程序不会调用 main() 方法。Applet 被设计为嵌入在一个 HTML 页面。当用户浏览包含 Applet 的 HTML 页面，Applet 的代码就被下载到用户的机器上。要查看一个 Applet 需要 JVM。 JVM 可以是 Web 浏览器的一个插件，或一个独立的运行时环境。用户机器上的 JVM 创建一个 Applet 类的实例，并调用 Applet 生命周期过程中的各种方法。Applet 有 Web 浏览器强制执行的严格的安全规则，Applet 的安全机制被称为沙箱安全。Applet 需要的其他类可以用 Java 归档（JAR）文件的形式下载下来。

②Java AWT

图形用户界面(Graphics User Interface, GUI) 是用户与程序交互的窗口，它比基于命令行的界面更直观并且更友好。

GUI的基本类库位于java.awt包中， 这个包也被称为抽象窗口工具箱(Abstract Window Toolkit, AWT)。AWT按照面向对象的思想来创建GUI，它提供了容器类、众多的组件类和布局管理器类。

AWT构建图形用户界面的机制包括：

提供了一些容器组件(如Frame和Panel), 用来容纳其他的组件(如按钮Button、复选框Checkbox和文本框TextField)。

用布局管理器来管理组件在容器上的布局；

利用监听器来响应各种事件，实现用户与程序的交互。一个组件如果注册了某种事件的监听器，由这个组件触发的特定事件就会被监听器接收和响应；

③Java Swing

Swing 是一个为Java设计的GUI工具包。

Swing是JAVA基础类的一部分。

Swing包括了图形用户界面（GUI）器件如：文本框，按钮，分隔窗格和表。

Swing提供许多比AWT更好的屏幕显示元素。它们用纯Java写成，所以同Java本身一样可以跨平台运行，这一点不像AWT。它们是JFC的一部分。它们支持可更换的面板和主题（各种操作系统默认的特有主题），然而不是真的使用原生平台提供的设备，而是仅仅在表面上模仿它们。这意味着你可以在任意平台上使用JAVA支持的任意面板。轻量级组件的缺点则是执行速度较慢，优点就是可以在所有平台上采用统一的行为。

1. **实验内容：**

1. 完成第七章习题6编程。

2. 完成第八章习题6编程。

3. 完成第九章习题10编程。

4. 完成第十章习题6编程。

5. 完成第十一章习题7编程。

6. 完成第十一章习题10编程。

1. **实验器材（设备、元器件）：**

硬件要求：普通计算机。

软件要求：Windows操作系统，Java开发环境JDK1.8。

1. **实验步骤：**

1.明确项目要求

2.编写代码

3.编译代码

4.测试程序

5.根据测试结果对程序进行调试改进

1. **实验结果与分析（含重要数据结果分析或核心代码流程分析）**

1. 第七章习题6编程

|  |
| --- |
| 1)源程序：  class sample {  private String name;  sample(){}  String getName(){  return name;  }  }  public class S7E6 {  public static void main(String[] args) {  int[] array = new int[3];  int num=2;  for(int i=1; i<=3; i++) {  try {  System.out.println("Case" + i + ": ");  switch(i) {  case 1:  num /= 0;  break;  case 2:  array[3] = num;  break;  case 3:  new sample().getName().length();  break;  }  }  catch(ArithmeticException e) {  System.out.println(e);  }  catch(IndexOutOfBoundsException e) {  System.out.println(e);  }  catch(NullPointerException e) {  System.out.println(e);  }  finally {  System.out.println("Finish Case" + i + ".");  }  }  }  } |
| 2)运行结果：   |  | | --- | | $ java S7E6  Case1:  java.lang.ArithmeticException: / by zero  Finish Case1.  Case2:  java.lang.ArrayIndexOutOfBoundsException: 3  Finish Case2.  Case3:  null  Finish Case3. | |
| 3)实验结论：测试正确 |

2. 完成第八章习题6编程。

|  |
| --- |
| 1)源程序：  import java.util.\*;  import java.text.\*;  public class S8E6s {  public static String timeString1;  public static String timeString2;  public static String timeString3;  public static void main(String[] args) {  Thread t1 = new Thread(new PekingTime());  Thread t2 = new Thread(new TokyoTime());  Thread t3 = new Thread(new LondonTime());  t1.start();  t2.start();  t3.start();  try {  Thread.sleep(1000);  }  catch(InterruptedException e) {}  while(true) {  try {  Thread.sleep(1000);  }  catch(InterruptedException e) {}  System.out.print("Peking: "+timeString1+"\t"+"Tokyo: "+timeString2+"\t"+"London: "+timeString3 + "\r");  }  }  }  class PekingTime implements Runnable {  public void run() {  while(true) {  try {  Thread.sleep(1000);  }  catch(InterruptedException e) {}  Date date = new Date();  SimpleDateFormat bjSdf = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");  bjSdf.setTimeZone(TimeZone.getTimeZone("Asia/Shanghai"));  //System.out.println("Peking:\t" + bjSdf.format(date) + "\r");  S8E6s.timeString1 = bjSdf.format(date).toString();  }  }  }  class TokyoTime implements Runnable {  public void run() {  while(true) {  try {  Thread.sleep(1000);  }  catch(InterruptedException e) {}  Date date = new Date();  SimpleDateFormat tokyoSdf = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");  tokyoSdf.setTimeZone(TimeZone.getTimeZone("Asia/Tokyo"));  //System.out.println("Tokyo:\t" + tokyoSdf.format(date) + "\r");  S8E6s.timeString2 = tokyoSdf.format(date).toString();  }  }  }  class LondonTime implements Runnable {  public void run() {  while(true) {  try {  Thread.sleep(1000);  }  catch(InterruptedException e) {}  Date date = new Date();  SimpleDateFormat londonSdf = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");  londonSdf.setTimeZone(TimeZone.getTimeZone("Europe/London"));  //System.out.println("London:\t" + londonSdf.format(date) + "\r");  S8E6s.timeString3 = londonSdf.format(date).toString();  }  }  } |
| 2)运行结果：   |  | | --- | | $ java S8E6s  Peking: 2018-12-23 11:29:14 Tokyo: 2018-12-23 12:29:14 London: 2018-12-23 03:29:14  Peking: 2018-12-23 11:29:16 Tokyo: 2018-12-23 12:29:16 London: 2018-12-23 03:29:16 | |
| 3)实验结论：测试正确 |

3. 完成第九章习题10编程。

|  |
| --- |
| 1)源程序：  import java.io.\*;  import java.util.\*;  public class S9E10 {  public static void main(String[] args) {  Scanner in = new Scanner(System.in);  try {  PrintWriter out = new PrintWriter("newfile.txt");  String str = in.nextLine();  out.println(str);  out.close();  } catch (FileNotFoundException e) {  e.printStackTrace();  }  }  } |
| 2)运行结果：   |  | | --- | | $ java S9E10  test input stream  $ cat newfile.txt  test input stream | |
| 3)实验结论：测试正确 |

4. 完成第十章习题6编程。

|  |  |  |
| --- | --- | --- |
| 1)源程序：   |  | | --- | | import java.io.BufferedReader;  import java.io.DataOutputStream;  import java.io.FileWriter;  import java.io.IOException;  import java.io.InputStream;  import java.io.InputStreamReader;  import java.io.OutputStream;  import java.io.PrintWriter;  import java.net.ServerSocket;  import java.net.Socket;  public class S10E6Server {  public static void main(String[] args) {  try {  ServerSocket server = new ServerSocket(10006);  Socket s = server.accept();  System.out.println(s.getInetAddress().getHostAddress()+" is uploading a file...");  //读取客户端上传过来的文本文件  //源 ---socket(字节流)---额外：需要转换成字符流 ，缓存流  InputStream in = s.getInputStream();  BufferedReader br = new BufferedReader(new InputStreamReader(in));  //目的 ---硬盘字符流 FileWriter---额外：打印流  PrintWriter pw = new PrintWriter(new FileWriter("server.txt"),true);  String line = null;  while((line=br.readLine())!=null){  pw.println(line);  }  pw.close();  OutputStream out = s.getOutputStream();  DataOutputStream dout = new DataOutputStream(out);  dout.writeUTF("File was uploaded successfully.");  s.close();  server.close();  dout.close();    } catch (IOException e) {  e.printStackTrace();  }  }  } |      |  | | --- | | import java.io.BufferedReader;  import java.io.DataInputStream;  import java.io.FileReader;  import java.io.IOException;  import java.io.InputStream;  import java.io.OutputStream;  import java.io.PrintWriter;  import java.net.Socket;  import java.net.UnknownHostException;  public class S10E6Client {  public static void main(String[] args) {  try {  Socket s = new Socket("127.0.0.1", 10006);  //思路：把本地文件的数据读取出来通过 s.getOutputStream()获得的out对象发送出去  BufferedReader bf = new BufferedReader(new FileReader("client.txt"));  OutputStream out = s.getOutputStream();//这里的输出流 对应的是服务器端的输入流  PrintWriter pw = new PrintWriter(out,true);  String str=null;  while((str=bf.readLine())!=null){  pw.println(str);  }  //给服务器发送结束标记---上传结束，要加结束标记，  s.shutdownOutput();  bf.close();  //接收服务器端反馈  InputStream in = s.getInputStream();  DataInputStream din = new DataInputStream(in);  System.out.println("server responsed: "+din.readUTF());  s.close();  din.close();  } catch (UnknownHostException e) {  e.printStackTrace();  } catch (IOException e) {  e.printStackTrace();  }  }  } | |
| 2)运行结果： |
| 3)实验结论：测试正确 |

5. 完成第十一章习题7编程。

|  |
| --- |
| 1)源程序：  import java.awt.\*;  import java.awt.event.ActionEvent;  import java.awt.event.ActionListener;  import javax.swing.\*;  import java.io.\*;  public class ChangeColor extends JFrame {  public JButton redBt = new JButton("set red");  public JButton greenBt = new JButton("set green");  public JButton blueBt = new JButton("set blue");  public ChangeColor() {  super("Change Color Program");  Container container = getContentPane();  container.setLayout(new GridLayout(1,3));  container.add(redBt);  container.add(greenBt);  container.add(blueBt);  redBt.addActionListener(new ActionListener() {  @Override  public void actionPerformed(ActionEvent e) {  redBt.setForeground(Color.RED);  greenBt.setForeground(Color.RED);  blueBt.setForeground(Color.RED);  }  });  greenBt.addActionListener(new ActionListener() {  @Override  public void actionPerformed(ActionEvent e) {  redBt.setForeground(Color.GREEN);  greenBt.setForeground(Color.GREEN);  blueBt.setForeground(Color.GREEN);  }  });  blueBt.addActionListener(new ActionListener() {  @Override  public void actionPerformed(ActionEvent e) {  redBt.setForeground(Color.BLUE);  greenBt.setForeground(Color.BLUE);  blueBt.setForeground(Color.BLUE);  }  });  setSize(330, 200);  setVisible(true);  setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);  }  public static void main(String[] args) {  new ChangeColor();  }  } |
| 2)运行结果： |
| 3)实验结论：测试正确 |

6. 完成第十一章习题10编程。

|  |
| --- |
| 1)源程序：  import java.awt.\*;  import java.awt.event.\*;  import java.io.\*;  import javax.swing.\*;  import javax.swing.text.\*;  public class TextEditor extends JFrame {  JTextPane textPane = new JTextPane();  JLabel statusBar = new JLabel(); //状态栏  JFileChooser fileChooser = new JFileChooser(); //文件选择器  public TextEditor() {  super("TextEditor"); //调用JFrame接收字符串的构造函数，能够改变窗口的标题  Action[] actions = {  new NewAction(), //"新建"操作  new OpenAction(), //"打开"操作  new SaveAction(), //"保存"操作  new ExitAction(), //"退出"操作  };  setJMenuBar(createJMenuBar(actions)); //菜单栏  Container container = getContentPane();  //container.add(createJToolBar(actions), BorderLayout.NORTH);  container.add(textPane, BorderLayout.CENTER);  container.add(statusBar, BorderLayout.SOUTH);  setSize(330, 200);  setVisible(true);  setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);  }  private JMenuBar createJMenuBar(Action[] actions) {  JMenuBar menuBar = new JMenuBar();  JMenu menuFile = new JMenu("File");  menuFile.add(new JMenuItem(actions[0]));  menuFile.add(new JMenuItem(actions[1]));  menuFile.add(new JMenuItem(actions[2]));  menuFile.add(new JMenuItem(actions[3]));  menuBar.add(menuFile);  return menuBar;  }  class NewAction extends AbstractAction {  public NewAction() {  super("New");  }  public void actionPerformed(ActionEvent e) {  textPane.setDocument(new DefaultStyledDocument());  }  }  class OpenAction extends AbstractAction {  public OpenAction() {  super("Open");  }  public void actionPerformed(ActionEvent e) {  int i = fileChooser.showOpenDialog(TextEditor.this);  if(i == JFileChooser.APPROVE\_OPTION) {  File f = fileChooser.getSelectedFile();  try {  InputStream is = new FileInputStream(f);  textPane.read(is, "d");  }  catch(Exception ex) {  ex.printStackTrace();  }  }  }  }  class SaveAction extends AbstractAction {  public SaveAction() {  super("Save");  }  public void actionPerformed(ActionEvent e) {  int i = fileChooser.showSaveDialog(TextEditor.this);  if(i == JFileChooser.APPROVE\_OPTION) {  File f = fileChooser.getSelectedFile();  try {  FileOutputStream out = new FileOutputStream(f);  out.write(textPane.getText().getBytes());  }  catch(Exception ex) {  ex.printStackTrace();  }  }  }  }  class ExitAction extends AbstractAction {  public ExitAction() {  super("Exit");  }  public void actionPerformed(ActionEvent e) {  System.exit(0);  }  }  public static void main(String[] args) {  new TextEditor();  }  } |
| 2)运行结果： |
| 3)实验结论：测试正确 |

1. **总结及心得体会：**

①本次实验涉及的知识量很大，需要对Java异常处理、多线程、输入输出流、网络编程以及GUI都有较为深刻的理解，不仅要会单独应用这些知识，还必须将其结合在一起，进行编程使用。

②个人觉得主要难点集中于多线程以及输入输出流，多线程难在对于对象锁的理解，理解对象锁的概念，是掌握线程同步的必要条件。输入输出流难在大量的方法需要区别，不同的方法对应不同的字符流、字节流处理方式，流的概念也较为抽象，需要更深入的理解。

③Java网络编程基础较为容易理解，但很容易编程出错。因为网络编程需要同时考虑客户端和服务端的数据传送流，在信息流的传送上要兼顾流的方向、流的类型等多方面因素的影响，所以网络编程需要更严格的程序设计。

④通过本次实验，个人掌握了许多Java的重要编程知识，提高了编程能力；同时也提高了自学能力和编程解决问题的能力，在本次实验中受益匪浅。

1. **对本实验过程及方法、手段的改进建议：**

①Java GUI开发部分中，Applet是较为落后，且被许多企业放弃的技术，建议将Applet的介绍篇幅减少，增加对Swing包的讲解。示例代码也应该减少使用Applet，而是使用更为流行的JFrame。

②在多线程部分，由于受到如今普遍的多核多线程处理器影响，某些示例代码的运行结果与单核处理器上的运行结果截然不同，简易对这类示例进行提示或修改。

③GUI部分的题目要求不够清晰，如十一章第7题的变色要求，没有阐述明白需要对哪几个按钮进行颜色的变化；另外十一章第10题的文本编辑器程序，没有说明除了界面外是否需要实现对应的功能。这些将影响到编程的难度。

**报告评分：**

**指导教师签字：**