第三部分
多媒体和多线程处理

20 实验 16：多线程处理

20.1 实验目的

当 Java 程序包含图形用户界面（GUI）时，Java 虚拟机在运行应用程序时会自动启动更多的线程，其中有两个重要的线程：AWT-EventQuecue 和 AWT-Windows。AWT-EventQuecue 线程负责处理 GUI 事件。AWT-Windows 线程负责将窗体或组件绘制到桌面。JVM 要保证各个线程都有使用 CPU 资源的机会，例如，程序中发生 GUI 界面事件时，JVM 就会将 CPU 资源切换给 AWT-EventQuecue 线程，AWT-EventQuecue 线程就会来处理这个事件，例如，你单击了程序中的按钮，触发 ActionEvent 事件，AWT-EventQuecue 线程就立刻排队等候执行处理事件的代码。本试验的目的是掌握在 GUI 中使用 Thread 的子类创建线程。

20.2 实验要求

编写一个 Java 应用程序，在主线程中再创建一个 Frame 类型的窗口，在该窗口中再创建一个线程 giveWord。线程 giveWord 每隔 2 秒钟给出一个汉字，用户使用一种汉字输人法将该汉字输人到文本框中。程序运行效果如图所示。

20.3 程序模板

请按模板要求，将【代码】替换为 Java 程序。

ThreadWordMainClass.java

```
public class ThreadWordMainClass {
  public static void main(String args[]) {
      new ThreadFrame().setTitle("汉字打字练习");
   }
```

```
}
```

WordThread.java

```
import javax.swing.JTextField;
public class WordThread extends Thread {
    char word;
    int startPosition =19968; //Unicode表的 19968位置至 32320上的汉字
    int endPosition = 32320;
    JTextField showWord;
    int sleepLength = 6000;
    public void setJTextField(JTextField t) {
        showWord = t;
        showWord.setEditable(false);
    }
    public void setSleepLength(int n){
       sleepLength = n;
    }
    public void run() {
       int k=startPosition;
       while(true) {
          word=(char)k;
          showWord.setText(""+word);
          try{
               【代码 1】//调用 sleep方法使得线程中断 sleepLength豪秒
          }
          catch(InterruptedException e){}
          k++;
          if(k>=endPosition)
             k=startPosition;
        }
    }
}
```

ThreadFrame.java

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
public class ThreadFrame extends JFrame implements ActionListener {
    JTextField showWord;
    JButton button;
    JTextField inputText,showScore;
    【代码 2】//用 WordThread声明一个 giveWord线程象
    int score = 0;
    ThreadFrame() {
        showWord = new JTextField(6);
        showWord.setFont(new Font("",Font.BOLD,72));
        showWord.setHorizontalAlignment(JTextField.CENTER );
        【代码 3】//创建 giveWord线程
        giveWord.setJTextField(showWord);
        giveWord.setSleepLength(5000);
        button = new JButton("开始");
        inputText = new JTextField(10);
        showScore = new JTextField(5);
        showScore.setEditable(false);
        button.addActionListener(this);
        inputText.addActionListener(this);
        add(button,BorderLayout.NORTH);
        add(showWord,BorderLayout.CENTER);
        JPanel southP=new JPanel();
```

```
        southP.add(new JLabel("输入汉字（回车）:"));
        southP.add(inputText);
        southP.add(showScore);
        add(southP,BorderLayout.SOUTH);
        setBounds(100,100,350,180);
        setVisible(true);
        validate();
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
    public void actionPerformed(ActionEvent e) {
        if(e.getSource()==button) {
            if(!(giveWord.isAlive())){
                【代码 4】//创建 giveWord
                giveWord.setJTextField(showWord);
                giveWord.setSleepLength(5000);
            }
            try {
                【代码 5】//giveWord调用方法 start()
            }
            catch(Exception exe){}
        }
        else if(e.getSource()==inputText) {
            if(inputText.getText().equals(showWord.getText()))
                score++;
            showScore.setText("得分:"+score);
            inputText.setText(null);
        }
    }
}
```

20.4 实验指导

AWT-Windows 线程负责将窗体或组件绘制到桌面。发生 GUI 界面事件时，JVM 就会将 CPU 资源切换给 AWT-EventQuecue 线程。

20.5 扩展练习

20.5.1

编写一个英文单词打字游戏。要求实现编辑一个英文单词组成的文本文件（单词用空格分隔），Widthread 线程使用 Scanner 类的实例解析文本文件中的单词。

20.5.2

输入以下 Java 应用程序，运行程序，分析程序的运行结果。

TwoThreadsTest.java

```
class SimpleThread extends Thread {
  public SimpleThread(String str) {
    super(str); //调用其父类的构造方法
  }
  public void run() { //重写 run方法
    for (int i = 0; i < 10; i++) {
      System.out.println(i + " " + getName());
    //打印次数和线程的名字
    try {
    sleep((int)(Math.random() * 1000));
    //线程睡眠，把控制权交出去
```

```
      }
      catch (InterruptedException e) { }
    }
    System.out.println("DONE! " + getName());
    //线程执行结束
  }
}
public class TwoThreadsTest {
  public static void main (String args[]) {
    new SimpleThread("First").start();
    //第一个线程的名字为 First
    new SimpleThread("Second").start();
    //第二个线程的名字为 Second
  }
}
```

21 独立实验任务 3（需要提交实验报告）

21.1 实验目的

熟悉 Java 的多线程应用程序开发方法。

21.2 实验任务

编写一个 Java 多线程应用程序，完成三个售票窗口同时出售 20 张票，如图所示。

21.2.1 任务要求

- 票数要使用同一个静态值；
- 为保证不会出现卖出同一个票数，要 java 多线程同步锁。

21.2.2 设计思路

- 创建一个站台类 Station，继承 Thread，重写 run 方法，在 run 方法里面执行售票操作。售票要使用同步锁：即有一个站台卖这张票时，其他站台要等这张票卖完。
- 创建主方法调用类。