University of NingBo Technology

《面向对象设计实验》

学　　院　宁波工程学院
专　　业　信息与计算科学
任课教师　陆星家

2018 年 3 月 10 日

目录

1 Java 的多线程机制

1、主要教学目标

(1) 理解并区分进程和线程的概念;
(2) 理解 Java 的多线程机制;
(3) 掌握使用 Thread 类和 Runnable 接口创建线程;
(4) 了解线程的生命周期及其控制方法;

2、重点内容

通过继承 Thread、实现接口 Runnable 来创建线程，实现 run() 方法，通过调用 start() 方法运行 run() 方法体。

3、难点分析

通过继承和实现接口，创建 Java 线程。

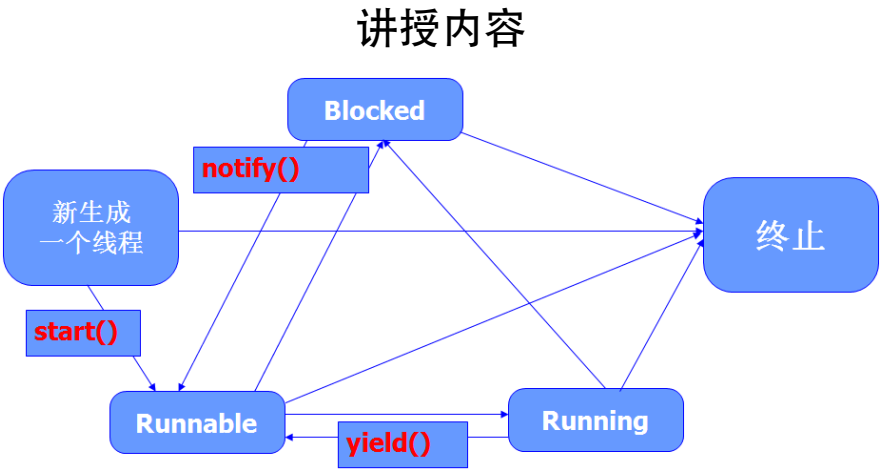


图 1: 线程在不同时刻的状态

1.1 多线程机制

- 如果一个程序被称为是多线程的, 则该程序包括两条以上并发的独立执行部分, 每一个这样的并发执行部分被称为一个线程 (thread)。
- 与基于进程 (process-based) 的执行环境相比较, 基于线程 (thread-based) 的执行环境是多任务处理的一种特殊形式。
- 多进程环境中每一个进程既包括其所要执行的指令, 也包括了执行指令所需的任何系统资源, 如 CPU、内存空间、I/O 端口等, 不同进程所占用的系统资源相对独立。
- 多线程环境中每一个线程都隶属于某一进程, 由进程触发执行, 在系统资源的使用上, 属于同一进程的所有线程共享该进程的系统资源。

- 正在运行 (running); 在等待运行 (runnable); 被唤醒 (notify); 被阻塞 (blocked); 被终止 (terminate);

1.2 线程的创建

通常有两种办法让我们来创建一个新的线程:

- 创建一个 Thread 类, 或者一个 Thread 子类的对象。
- 创建一个实现 Runnable 接口的类的对象。
- 每一个线程都有自己的优先级, 如果不显式地说明, 则被创建的线程具有与创建它的线程相同的优先级。

§1.1 继承 Thread 类

```
class MyThread extends Thread
{
       int begin,end;
       MyThread(int b, int e)
       {
      begin = b;
      end = e;
       }
       public void run()
       {
              int sum = 0;
              int i= begin;
              while(i<=end)
              {
          sum +=i;
          i++;
              }
              System.out.println("sum " + sum);
       }
}

public class Lesson11
{
       public static void main(String[] args)
       {
              int a,b,c;
              a = 1;
              b = 200;
              c = b/2;
              MyThread thread1 = new MyThread(a,c);
              MyThread thread2 = new MyThread(c+1,b);
```

```
32              thread1.start();
33              thread2.start();
34      }
35  }
```

§1.2 实现 Runnable 接口

```
1   class MyThread implements Runnable
2   {
3          int begin,end;
4          MyThread(int b, int e)
5          {
6         begin = b;
7         end = e;
8          }
9          public void run()
10         {
11                 int sum = 0;
12                 int i= begin;
13                 while(i<=end)
14                 {
15            sum +=i;
16            i++;
17                 }
18                 System.out.println("sum " + sum);
19         }
20  }
21
22  public class Lesson12
23  {
24         public static void main(String[] args)
25         {
26                 int a,b,c;
27                 a = 1;
28                 b = 200;
29                 c = b/2;
30                 //Thread thread1 = new Thread(new MyThread(a,c));
31                 //Thread thread2 = new Thread(new MyThread(c+1,b));
32                 MyThread thread1 = new MyThread(a,c);
33                 MyThread thread2 = new MyThread(c+1,b);
34                 thread1.run();
35                 thread2.run();
36                 //thread1.start();
37                 //thread2.start();
38         }
39  }
```

1.3 创建不同内容的线程

- 线程创建后仅仅是占有了内存资源，在 JVM 管理的线程中还没有这个线程，此线程必须调用 start() 方法（从父类继承的方法）通知 JVM，这样 JVM 就会知道又有一个新一个线程排队等候切换了。
- 线程调用 start() 将启动线程，使之从新建状态进入就绪队列排队，一旦轮到它来享用 CPU 资源时，就可以脱离创建它的线程独立开始自己的生命周期了。
- Thread 类的 run() 方法与 Runnable 接口中的 run() 方法的功能和作用相同，都用来定义线程对象被调度之后所执行的操作，都是系统自动调用而用户程序不得引用的方法。系统的 Thread 类中，run() 方法没有具体内容，所以用户程序需要创建自己的 Thread 类的子类，并重写 run() 方法来覆盖原来的 run() 方法。当 run 方法执行完毕，线程就变成死亡状态。
- 现程占有 CPU 期间，执行 sleep 方法来使自己放弃 CPU 资源，休眠一段时间。休眠时间的长短由 sleep 方法的参数决定，millsecond 是毫秒为单位的休眠时间。如果线程在休眠时被打断，JVM 就抛出 InterruptedException 异常。因此，必须在 try catch 语句块中调用 sleep 方法。

```
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.*;
class NumThread extends Thread {
    JLabel lab;
    NumThread(JLabel l)
    {
        lab = l;
    }
        public void run(){
               for(int i=1;i<=50;i++){
                      lab.setText("NumThread: "+i);
                      try{
                             sleep(500);
                      }
                      catch(InterruptedException e)
                      {
                      lab.setText(e.getMessage());
                      }
               }
        }
}
class LetterThread extends Thread{
        JLabel lab;
```

```
26        LetterThread(JLabel l)
27        {
28      lab = l;
29        }
30        public void run(){
31              for(char ch = 'a';ch<='z';ch++){
32                    lab.setText("LetterThread: "+ch);
33                    try
34                    {
35                          sleep(500);
36                    }
37                    catch(InterruptedException e)
38                    {
39                    lab.setText(e.getMessage());
40                    }
41              }
42        }
43  }
44  public class Lesson13 {
45        public static void main(String[] args) {
46              JLabel lab1,lab2;
47              JPanel pan;
48              JFrame frm = new JFrame("Two Threads");
49              lab1 = new JLabel();
50              lab2 = new JLabel();
51              pan = new JPanel();
52              frm.setSize(300,300);
53              frm.setLocationRelativeTo(null);
54        frm.setContentPane(pan);
55        pan.add(lab1);
56        pan.add(lab2);
57              NumThread thread1 = new NumThread(lab1);
58              LetterThread thread2 = new LetterThread(lab2);
59              thread1.start();
60              thread2.start();
61              frm.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
62              frm.setVisible(true);
63        }
64  }
```

```
1  import java.util.Date;
2  import javax.swing.*;
3  import java.awt.Color;
4
5  class TimeThread implements Runnable
6  {
```

图 2: 两个不同的线程

```
7         JLabel lab;
8         TimeThread(JLabel l)
9         {
10                lab = l;
11        }
12        public void run(){
13                String str;
14                Date date;
15                try{
16                        while(true)
17                        {
18                                date = new Date();
19                                str = date.toString().substring(11,19);
20                                Thread.sleep(100);
21                                lab.setText(str);
22                        }
23                }
24                catch(Exception e){
25                        System.out.println(e.toString());
26                }
27        }
28  }
29
30  class ColorThread implements Runnable
31  {
32        JPanel pane;
33        ColorThread(JPanel p)
34        {
35                pane = p;
36        }
37        public void run()
38        {
39                String str;
```

```
40              Date date;
41              try{
42                      while(true)
43                      {
44                              date = new Date();
45                              str = date.toString().substring(11,19);
46                              Thread.sleep(100);
47                              if((Integer.parseInt(str.substring(6,8))%3==0))
48                              {
49                                      pane.setBackground(Color.GREEN);
50                              }
51                              if((Integer.parseInt(str.substring(6,8))%7==0))
52                              {
53                                  pane.setBackground(Color.WHITE);
54                              }
55
56                      }
57              }
58              catch(Exception e){
59                      System.out.println(e.toString());
60              }
61      }
62
63  }
64
65  public class Lesson14{
66      public static void main(String[] args) {
67              JFrame frm = new JFrame("Show Time");
68              frm.setSize(300,100);
69              frm.setLocationRelativeTo(null);
70              JLabel lab = new JLabel();
71              JPanel pane = new JPanel();
72              Thread time = new Thread(new TimeThread(lab));
73              Thread color = new Thread(new ColorThread(pane));
74              time.start();
75              color.start();
76              frm.setContentPane(pane);
77              frm.getContentPane().add(lab);
78      frm.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
79      frm.setVisible(true);
80      }
81  }
```

```
1  import javax.swing.JFrame;
2  import javax.swing.JPanel;
3  import javax.swing.JLabel;
```

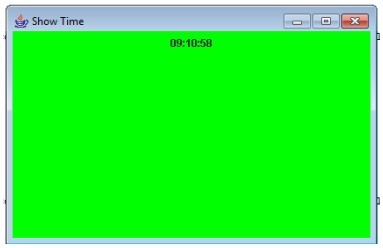


图 3: 两个不同的线程

```
4  import javax.swing.JButton;
5  import javax.swing.JTextField;
6  import java.awt.event.ActionListener;
7  import java.awt.event.ActionEvent;
8
9
10 class MyThread extends Thread
11 {
12        int begin,end;
13        JLabel label;
14        MyThread(int b, int e, JLabel l)
15        {
16       begin = b;
17       end = e;
18       label = l;
19        }
20        public void run()
21        {
22               int sum = 0;
23               int i= begin;
24               while(i<=end)
25               {
26           sum +=i;
27           i++;
28                }
29               label.setText("sum " + sum);
30        }
31 }
32
33 class MyFrame extends JFrame implements ActionListener
34 {
35        JPanel pane;
36        JTextField tf;
37        JLabel lab1,lab2;
38        JButton btn;
39        int middle;
```

```
40        MyFrame(String msg)
41        {
42               super(msg);
43           pane = new JPanel();
44           tf = new JTextField(15);
45           lab1 = new JLabel("Input Integer");
46           lab2 = new JLabel("Input Integer");
47           btn = new JButton("Click");
48           setSize(300,300);
49               setLocationRelativeTo(null);
50        setContentPane(pane);
51        pane.add(lab1);
52        pane.add(lab2);
53        pane.add(tf);
54        pane.add(btn);
55        btn.addActionListener(this);
56        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
57               setVisible(true);
58        }
59        public void actionPerformed(ActionEvent e)
60    {
61        if(e.getSource()==btn)
62        {
63        middle = (Integer.parseInt(tf.getText()))/2;
64        MyThread thread1 = new MyThread(1,middle,lab1);
65        MyThread thread2 = new MyThread(middle,middle*2,lab2);
66        thread1.start();
67        thread2.start();
68        }
69    }
70 }
71
72 public class Lesson15
73 {
74        public static void main(String[] args)
75        {
76               MyFrame frm = new MyFrame("Calculator");
77        }
78 }
```

1.4 作业

1、利用线程制作龟兔赛跑的动画。

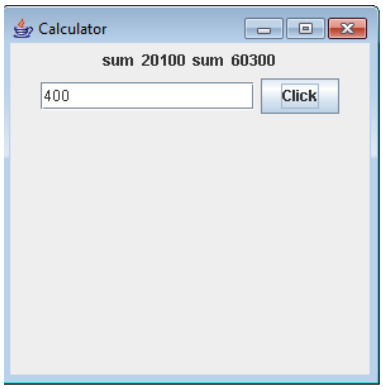


图 4: 两个不同的线程

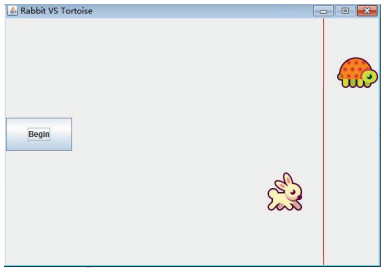


图 5: 龟兔赛跑

2 线程的同步机制和计时器 Timer 类

1、主要教学目标

(1) 理解多线程的同步机制;
(2) 理解线程通过互斥、加锁等协同操作互斥区;
(3) 理解 Java 的 Timer 类的使用。

2、重点内容

掌握 Java 中的同步机制，线程间的互斥、加锁，掌握 Java 中的 join、wait、notify、sychronized 等关键词的使用。

3、难点分析

Java 线程间的互斥、加锁，Java 的 join、wait、notify、sychronized 等关键词的使用。

讲授内容

2.1 线程的同步机制

§1.1 join 插队方法

在编写多线程程序时，如果希望一个线程能够优先于其他线程运行时，此时除了可以设置该线程的优先级高于其他线程外，更直接的方法是使用 Thread 类的 join 方法。使用 join() 方法可以让调用该方法的线程优先于其他线程发生，实现类似”插队“的效果，当”插队“的线程运行完毕后，其他线程会继续运行，不会因为”插队“而发生改变。

- join() 等待调用该方法的线程终止
- join(long millis) 等待调用该方法的线程终止的时间最长为 millis 毫秒
- join(long millis, int nanos) 等待调用该方法的线程终止的时间最长为 millis 毫秒加 nanos 纳秒

```
import java.lang.*;

class MyThread implements Runnable {
   public void run()
   {
   Thread t = Thread.currentThread();
   try
   {
   Thread.sleep(1000);
   }
   catch(InterruptedException e)
   {
      e.toString();
   }
   System.out.print(t.getName());
   System.out.println(", status = " + t.isAlive());
   }
}
public class Lesson21
{
   public static void main(String args[]) throws Exception
   {
   Thread t = new Thread(new MyThread());
   t.start();
   t.join();
   System.out.print("---"+t.getName());
   System.out.println(", status = " + t.isAlive());
   }
}
```

```
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.JButton;
import javax.swing.JLabel;

```

```
class Numthread extends Thread
{
	private JLabel lab;
	Numthread(JLabel l)
	{
		lab =l;
	}
	public void run()
	{
		int i=1;
		try
		{
		while(i<=10)
		{
			lab.setText("Number = " + i);
			sleep(1000);
			i++;
		}
	    }
	    catch(Exception e)
	    {
	        e.printStackTrace();
	    }
	    }
}
class Charthread extends Thread
{
	private JLabel lab;
	Charthread( JLabel l)
	{
		lab = l;
	}
	public void run()
	{
		char c ='a';
		try
		{
		while(c<='z')
		{
			lab.setText("Character = " + c);
			sleep(1000);
			c++;
	    }
	    }
	    catch(Exception e)
	    {
```

```
52                 e.printStackTrace();
53             }
54     }
55 }
56 
57 public class Lesson22
58 {
59         public static void main(String[] args)
60         {
61                 JPanel pane;
62                 JLabel lab1,lab2;
63                 pane = new JPanel();
64                 lab1 = new JLabel("Number");
65                 lab2 = new JLabel("Label2");
66         JFrame frm = new JFrame("Join Demo");
67                 frm.setSize(400,400);
68         frm.setLocationRelativeTo(null);
69         frm.setContentPane(pane);
70         pane.add(lab1);
71         pane.add(lab2);
72         frm.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
73         frm.setVisible(true);
74                 Numthread thread1 = new Numthread(lab2);
75         Charthread thread2 = new Charthread(lab2);
76         thread1.start();
77         thread2.start();
78             System.out.println("MultiThread Finish\n");
79         }
80 }
```

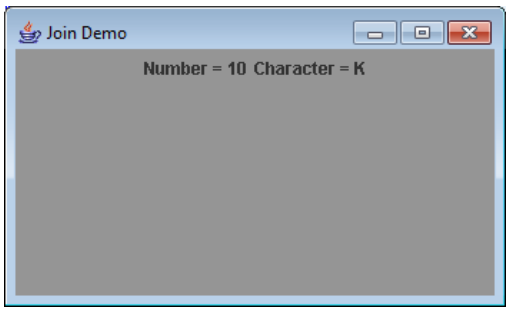


图 1: 优先执行 Numthread 线程

```
1 import javax.swing.JFrame;
2 import javax.swing.JPanel;
3 import javax.swing.JButton;
4 import javax.swing.JLabel;
```

```
5  import java.awt.Color;
6  import java.util.Random;
7
8  class Numthread extends Thread
9  {
10         private JLabel lab;
11         Numthread(JLabel l)
12         {
13                 lab =l;
14         }
15         public void run()
16         {
17                 int i=1;
18                 try
19                 {
20                 while(i<=10)
21                 {
22                         lab.setText("Number = " + i);
23                         sleep(1000);
24                         i++;
25                 }
26             }
27             catch(Exception e)
28             {
29                 e.printStackTrace();
30             }
31             }
32 }
33
34 class Charthread extends Thread
35 {
36         private Numthread th;
37         private JLabel lab;
38         Charthread(Numthread t, JLabel l)
39         {
40                 th = t;
41                 lab = l;
42         }
43         public void run()
44         {
45                 char c ='A';
46                 try
47                 {
48             th.join();
49                 while(c<='K')
50                 {
```

```
51                      lab.setText("Character = " + c);
52                      sleep(1000);
53                      c++;
54          }
55          }
56          catch(Exception e)
57          {
58              e.printStackTrace();
59          }
60      }
61  }
62
63  class Colorthread extends Thread
64  {
65          private Numthread th;
66          private JPanel pane;
67          Colorthread(Numthread t, JPanel p)
68          {
69                  th = t;
70                  pane = p;
71          }
72          public void run()
73          {
74                  try
75                  {
76              th.join();
77              char c='A';
78                  while(c<='z')
79                  {
80                          Random m = new Random();
81                          int random = m.nextInt(256);
82                          pane.setBackground(new Color(random,random,random));
83                          sleep(1000);
84                          c++;
85              }
86              }
87              catch(Exception e)
88              {
89                  e.printStackTrace();
90              }
91      }
92  }
93
94  public class Lesson23
95  {
96          public static void main(String[] args)
```

```
97          {
98                  JPanel pane;
99                  JLabel lab1,lab2;
100                 pane = new JPanel();
101                 lab1 = new JLabel("Label1");
102                 lab2 = new JLabel("Label2");
103         JFrame frm = new JFrame("Join Demo");
104                 frm.setSize(200,200);
105         frm.setLocationRelativeTo(null);
106         frm.setContentPane(pane);
107         pane.add(lab1);
108         pane.add(lab2);
109         frm.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
110         frm.setVisible(true);
111                 Numthread thread1 = new Numthread(lab1);
112         Charthread thread2 = new Charthread(thread1,lab2);
113         Colorthread thread3 = new Colorthread(thread1,pane);
114         thread1.start();
115         thread2.start();
116         thread3.start();
117         try
118         {
119                 thread1.join();
120             }
121             catch (InterruptedException e)
122             {
123                 e.printStackTrace();
124             }
125         }
126 }
```

§1.2 Sleep 休眠方法

在龟兔赛跑的故事里，原本领先的兔子因为骄傲自满在中途休息而痛失冠军。通过调用 sleep 方法来模拟龟兔赛跑。

```
1  import javax.swing.JFrame;
2  import javax.swing.JPanel;
3  import javax.swing.JButton;
4  import java.awt.BorderLayout;
5  import javax.swing.JLabel;
6  import javax.swing.JScrollPane;
7  import javax.swing.JTextArea;
8  import java.awt.event.ActionEvent;
9  import java.awt.event.ActionListener;
10
```

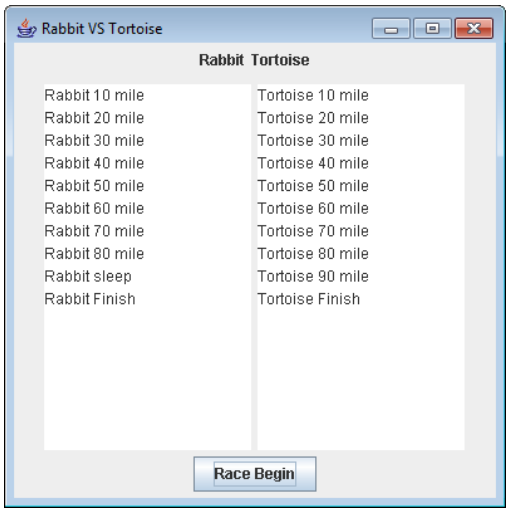


图 2: 龟兔赛跑

```
11 class RaceFrame extends JFrame
12 {
13         private JLabel lab1,lab2;
14         private JPanel pane,pane1,pane2,pane3;
15         private BorderLayout blay;
16         private JButton btn;
17         private JTextArea ta1,ta2;
18         RaceFrame(String msg)
19         {
20       super(msg);
21       setSize(400,400);
22       setLocationRelativeTo(null);
23       blay = new BorderLayout();
24       pane = new JPanel();
25       pane1 = new JPanel();
26       pane2 = new JPanel();
27       pane3 = new JPanel();
28       lab1 = new JLabel("Rabbit");
29       lab2 = new JLabel("Tortoise");
30       btn = new JButton("Race Begin");
31       ta1 = new JTextArea(20,15);
32       ta2 = new JTextArea(20,15);
33       pane.setLayout(blay);
34       setContentPane(pane);
35       pane.add(pane1,BorderLayout.NORTH);
36       pane1.add(lab1);
37       pane1.add(lab2);
38       pane.add(pane2,BorderLayout.CENTER);
39       pane2.add(ta1);
```

```
40      pane2.add(ta2);
41      pane.add(pane3,BorderLayout.SOUTH);
42      pane3.add(btn);
43      ActionListener run = new runAction();
44      btn.addActionListener(run);
45      setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
46      setVisible(true);
47        }
48
49        private class Rabbit implements Runnable
50        {
51               public void run()
52               {
53                      for(int i=1;i<11;i++)
54                      {
55                             String text = ta1.getText();
56                             try
57                             {
58                                    Thread.sleep(100);
59                             }
60                             catch(InterruptedException e)
61                             {
62                                    e.printStackTrace();
63                             }
64                             ta1.setText(text + "Rabbit " + i +"0 mile \n");
65                             if(i==9)
66                             {
67                                    ta1.setText(text + "Rabbit sleep\n");
68                                    try
69                                    {
70                                           Thread.sleep(10000);
71                                    }
72                                    catch(InterruptedException e)
73                                    {
74                                           e.printStackTrace();
75                                    }
76                             }
77                             if(i==10)
78                             {
79                                    try
80                                    {
81                                           Thread.sleep(1);
82                                    }
83                                    catch(InterruptedException e)
84                                    {
85                                           e.printStackTrace();
```

```
86                                      }
87                                      ta1.setText(text + "Rabbit Finish\n");
88                              }
89                      }
90              }
91
92      }
93
94      private class Tortoise implements Runnable
95      {
96
97              public void run()
98              {
99                      for(int i=1;i<11;i++)
100                     {
101                             String text = ta2.getText();
102                             try
103                             {
104                                     Thread.sleep(300);
105                             }
106                             catch(InterruptedException e)
107                             {
108                                     e.printStackTrace();
109                             }
110                             ta2.setText(text + "Tortoise " + i +"0 mile \n");
111                     if(i==10)
112                             ta2.setText(text + "Tortoise Finish\n");
113                     }
114             }
115     }
116
117     private class runAction implements ActionListener
118     {
119             public void actionPerformed(ActionEvent e)
120             {
121                     if(e.getSource()==btn)
122                     {
123             Thread rabbit1 = new Thread(new Rabbit());
124             Thread tortoise1 = new Thread(new Tortoise());
125             rabbit1.start();
126             tortoise1.start();
127                     }
128              }
129     }
130 }
131
```

```
132
133 public class Lesson24
134 {
135         public static void main(String[] args)
136         {
137                 RaceFrame frm = new RaceFrame("Rabbit VS Tortoise");
138         }
139 }
```

§1.3 synchronized 方法

- 线程同步问题线程同步是指几个线程都需要调用一个同步方法 (使用关键字 synchronized 修饰的方法)
- 同步机制　当一个线程 A 使用一个 synchronized 修饰的方法时，其他线程想使用这个方法时就必须等待，直到线程 A 使用完该方法（除非线程 A 使用 wait 主动让出 CUP 资源）.
- 等待与通知一个线程在使用的同步方法中时，可能根据问题的需要，必须使用 wait() 方法使本线程等待，暂时让出 CPU 的使用权，并允许其它线程使用这个同步方法。其它线程如果在使用这个同步方法时如果不需要等待，那么它用完这个同步方法的同时，应当执行 notifyAll() 方法通知所有的由于使用这个同步方法而处于等待的线程结束等待。

```
1  class RunnableDemo implements Runnable {
2    public Thread t;
3    private String threadName;
4    boolean suspended = false;
5    RunnableDemo( String name)
6    {
7       threadName = name;
8       System.out.println("Creating " + threadName );
9    }
10
11   public void run() {
12      System.out.println("Running " + threadName );
13      try {
14         for(int i = 10; i > 0; i--) {
15            System.out.println("Thread: " + threadName + ", " + i);
16            Thread.sleep(300);
17            synchronized(this) {
18               while(suspended) {
19                  wait();
20               }
21            }
22         }
23      }catch (InterruptedException e) {
```

```
24          System.out.println("Thread " + threadName + " interrupted.");
25       }
26       System.out.println("Thread " + threadName + " exiting.");
27    }
28
29    public void start () {
30       System.out.println("Starting " + threadName );
31       if (t == null) {
32          t = new Thread (this, threadName);
33          t.start ();
34       }
35    }
36
37    void suspend() {
38       suspended = true;
39    }
40
41    synchronized void resume() {
42       suspended = false;
43       notify();
44    }
45 }
46
47 public class Lesson25
48 {
49    public static void main(String args[]) {
50       RunnableDemo R1 = new RunnableDemo( "Thread-1");
51       R1.start();
52       RunnableDemo R2 = new RunnableDemo( "Thread-2");
53       R2.start();
54       try {
55          Thread.sleep(1000);
56          R1.suspend();
57          System.out.println("Suspending First Thread");
58          Thread.sleep(1000);
59          R1.resume();
60          System.out.println("Resuming First Thread");
61          R2.suspend();
62          System.out.println("Suspending thread Two");
63          Thread.sleep(1000);
64          R2.resume();
65          System.out.println("Resuming thread Two");
66       }catch (InterruptedException e) {
67          System.out.println("Main thread Interrupted");
68       }try {
69          System.out.println("Waiting for threads to finish.");
```

```
70          R1.t.join();
71          R2.t.join();
72        }catch (InterruptedException e) {
73          System.out.println("Main thread Interrupted");
74        }
75        System.out.println("Main thread exiting.");
76     }
77  }
```

2.2 计时器 Timer 类的使用

使用 Timer 类的构造方法：Timer(int a, Object b) 创建一个计时器参数 a 的单位是毫秒，确定计时器每隔 a 毫秒“震铃”一次，参数 b 是计时器的监视器。计时器发生震铃事件是 ActionEvent 事件，参数 b 应为实现了 ActionListener 接口的类对象。每隔 a 毫秒调用一次 actionPerformed() 方法。

- 使用 Timer 类的 setReapeats(boolean b) 方法可以设置事件是否只触发一次；
- 使用 Timer 类的 setInitialDelay(int delay) 方法可以设置首次触发的延时；
- 使用 Timer 类的 start() 方法启动计时器，即启动线程。使用 stop() 方法停止计时器，即挂起线程，使用 restart() 方法重新启动计时器，即恢复线程.

```
1  import java.awt.event.*;
2  import java.awt.*;
3  import java.util.Date;
4  import javax.swing.*;
5
6  class TimeFrame extends JFrame implements ActionListener
7  {
8         JPanel pan;
9         JTextField text;
10        JButton bStart,bStop,bContinue;
11        Timer time;
12        TimeFrame(String s){
13        super(s);
14        this.setSize(300,100);
15        this.setLocationRelativeTo(null);
16        pan = new JPanel();
17        text = new JTextField(10);
18        bStart = new JButton("Start");
19        bStop = new JButton("Stop");
20        bContinue = new JButton("Continue");
21        pan.add(bStart);
22        pan.add(bStop);
23        pan.add(bContinue);
```

```
24          pan.add(text);
25          this.setContentPane(pan);
26          bStart.addActionListener(this);
27          bStop.addActionListener(this);
28          bContinue.addActionListener(this);
29          time = new Timer(1000,this);
30          setVisible(true);
31          setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
32  }
33
34          public void actionPerformed(ActionEvent e){
35                  if(e.getSource() == time){
36                          Date date = new Date();
37                          String str = date.toString().substring(11,19);
38                          text.setText("Time:"+str);
39                  }
40                  else if(e.getSource() == bStart){
41                          time.start();
42                  }
43                  else if(e.getSource() == bStop){
44                          time.stop();
45                  }
46                  else if(e.getSource() == bContinue){
47                          time.restart();
48                  }
49          }
50  }
51
52  public class Lesson26 {
53          public static void main(String[] args) {
54                  new TimeFrame("Timer");
55          }
56  }
```

2.3 DeadLock

是指两个或两个以上的进程在执行过程中，因争夺资源而造成的一种互相等待的现象，若无外力作用，它们都将无法推进下去。此时称系统处于死锁状态或系统产生了死锁，这些永远在互相等待的进程称为死锁进程。

- 互斥条件：一个资源每次只能被一个进程使用；
- 请求与保持条件：一个进程因请求资源而阻塞时，对已获得的资源保持不放；
- 不剥夺条件: 进程已获得的资源，在未使用完之前，不能强行剥夺；

- 循环等待条件: 若干进程之间形成一种头尾相接的循环等待资源关系。

以上四个条件是死锁的必要条件，只要系统发生死锁，这些条件必然成立，而只要上述条件之一不满足，就不会发生死锁。

```
public class Lesson27
{
   public static Object Lock1 = new Object();
   public static Object Lock2 = new Object();
   public static void main(String args[])
   {
      ThreadDemo1 T1 = new ThreadDemo1();
      ThreadDemo2 T2 = new ThreadDemo2();
      T1.start();
      T2.start();
   }
   private static class ThreadDemo1 extends Thread {
      public void run()
      {
         synchronized (Lock1)
         {
            System.out.println("Thread 1: Holding lock 1...");
            try { Thread.sleep(10); }
            catch (InterruptedException e) {}
            System.out.println("Thread 1: Waiting for lock 2...");
            synchronized (Lock2)
            {
               System.out.println("Thread 1: Holding lock 1 & 2...");
            }
         }
      }
   }
   private static class ThreadDemo2 extends Thread {
      public void run()
      {
         synchronized (Lock2)
         {
            System.out.println("Thread 2: Holding lock 2...");
            try { Thread.sleep(10); }
            catch (InterruptedException e) {}
            System.out.println("Thread 2: Waiting for lock 1...");
            synchronized (Lock1)
            {
               System.out.println("Thread 2: Holding lock 1 & 2...");
            }
         }
      }
```

```
43      }
44  }
```

2.4 作业

1、售票员售票

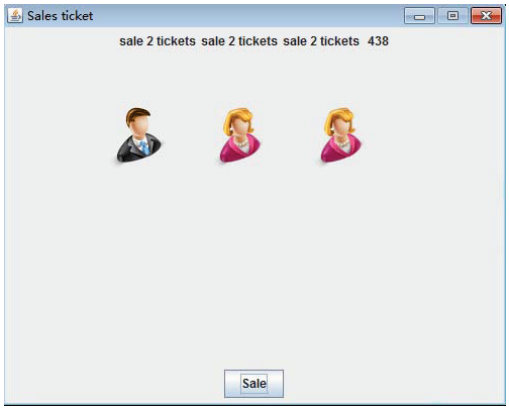


图 3: 售票员售票

3 平面文件处理

1、主要教学目标

(1) File 类的主要方法，通过 File 操纵文件、文件夹。
(2) InputStream、OutputStream 流。
(3) 通过流读写平面文件。
(4) TXT、CSV、JSON 文件的读写。

2、重点内容

利用 Java 的流, 访问文件与目录, 掌握平面文件流的输入和输出。

3、难点分析

在 Java 开发环境中, 利用 File 和 Stream 获取平面文件内容。

教学内容

3.1 File

§1 File

File 类提供了许多与平台无关的方法来对磁盘上的文件或目录进行操作，Java 为了能够做到与平台无关，克服不同系统对文件路径描述的差异。在 Java 语言中，使用抽象路径等概念，利用抽象路径，Java 可以自动进行不同系统平台的文件路径描述与抽象文件路径之间的转换。File 类的主要作用是操作目录和目录中文件的方法，文件类不涉及文件内容的操作。File 类的方法有：删除文件，创建目录，目录文件显示，查询文件的修改时间和尺寸。File 类提供方法来操纵文件和获得一个文件的信息。File 类是不对称的，意思是虽然存在允许验证一个简单文件对象属性的很多方法，但是没有响应的方法来修改这些属性，即有 get 无 set 方法。

- public String getName()　返回由此抽象路径名表示的文件或目录名。
- public String getParent()　返回此抽象路径名的父路径名的路径名字符串，如果此路径名没有指定父目录，则返回 null。
- public File getParentFile()　返回此抽象路径名的父路径名的抽象路径名，如果此路径名没有指定父目录，则返回 null。
- public String getPath()　将此抽象路径名转换为一个路径名字符串。
- public boolean isAbsolute()　测试此抽象路径名是否为绝对路径名。
- public String getAbsolutePath()　返回抽象路径名的绝对路径名字符串。
- public boolean canRead()　测试应用程序是否可以读取此抽象路径名表示的文件。

- public boolean canWrite() 测试应用程序是否可以修改此抽象路径名表示的文件。
- public boolean exists() 测试此抽象路径名表示的文件或目录是否存在。
- public boolean isDirectory() 测试此抽象路径名表示的文件是否是一个目录。
- public boolean isFile() 测试此抽象路径名表示的文件是否是一个标准文件。
- public long lastModified() 返回此抽象路径名表示的文件最后一次被修改的时间。
- public long length() 返回由此抽象路径名表示的文件的长度。
- public boolean createNewFile() throws IOException 当且仅当不存在具有此抽象路径名指定的名称的文件时，原子地创建由此抽象路径名指定的一个新的空文件。

```
import java.io.File;
import java.io.IOException;
import java.util.Scanner;
import java.io.BufferedReader;
import java.io.FileReader;

public class Lesson31
{
	public static void main(String[] args)
	{
		Scanner sc = new Scanner(System.in);
		String filename = sc.next();
		File f = new File(filename);
		if(f.isDirectory())
		{
			System.out.println("f is a directory");
		}
		if(f.isFile())
		{
	 try
	{
	BufferedReader in = new BufferedReader(new FileReader(f));

	   String line = null;
	while((line = in.readLine())!=null)
	  System.out.println("Line " + line + "\r\n");
	in.close();
	}
	catch(IOException e)
	{
		e.printStackTrace();
	}
		}
```

```
            if(!f.exists())
            {
         f.mkdirs();
       }

       }
}
```

```
import java.io.File;
import java.io.IOException;
import java.util.Scanner;
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.FilenameFilter;

public class Lesson32
{
       public static void main(String[] args)
       {
              Scanner sc = new Scanner(System.in);
              String file = sc.next();
              File f = new File(file);
              if(f.isDirectory())
              {
                     String[] files = f.list(new FilenameFilter()
                     {
                    public boolean accept(File f, String filename)
             {
        return filename.endsWith("txt");
        }});
                     for(int i=0;i<files.length;i++)
                     {
          try
          {
          BufferedReader in = new BufferedReader(new FileReader(new File(file,files[i])));
          String line = null;
          while((line = in.readLine())!=null)
            System.out.println("Content " + line + "\r\n");
          in.close();
          }
          catch(IOException e)
          {
              e.printStackTrace();
          }
                 }
                     System.out.println("Directory is closed");
```

```
39              }
40          }
41  }
```

- public boolean delete()　删除此抽象路径名表示的文件或目录。
- public void deleteOnExit()　在虚拟机终止时，请求删除此抽象路径名表示的文件或目录。
- public boolean mkdir()　创建此抽象路径名指定的目录。
- public boolean mkdirs()　创建此抽象路径名指定的目录，包括创建必需但不存在的父目录。
- public boolean renameTo(File dest)　重新命名此抽象路径名表示的文件。
- public boolean setLastModified(long time)　设置由此抽象路径名所指定的文件或目录的最后一次修改时间。
- public boolean setReadOnly()　标记此抽象路径名指定的文件或目录，以便只可对其进行读操作。
- public static File createTempFile(String prefix, String suffix, File directory) throws IOException 在指定目录中创建一个新的空文件，使用给定的前缀和后缀字符串生成其名称。
- public static File createTempFile(String prefix, String suffix) throws IOException　在默认临时文件目录中创建一个空文件，使用给定前缀和后缀生成其名称。
- public int compareTo(File pathname)　按字母顺序比较两个抽象路径名。
- public int compareTo(Object o)　按字母顺序比较抽象路径名与给定对象。
- public boolean equals(Object obj)　测试此抽象路径名与给定对象是否相等。
- public String toString()　返回此抽象路径名的路径名字符串。

§2 Directory

一个 File 类对象既可以表示目录也可以表示文件。mkdir 创建文件夹,mkdirs 根据抽象路径名创建目录。

```
1  import java.io.File;
2  import java.io.IOException;
3  import java.util.Scanner;
4  import java.io.InputStream;
5  import java.io.OutputStream;
6  import java.io.FileInputStream;
7  import java.io.FileOutputStream;
8
9  public class Lesson33
10 {
11         public static void main(String[] args)
```

```
12      {
13              Scanner sc = new Scanner(System.in);
14              String infile = sc.next();
15              File fin = new File(infile);
16              String outfile = fin.getAbsolutePath() + "copy" + fin.getName();
17              File fout = new File(outfile);
18              if(fin.exists())
19              {
20                      if(fin.isFile())
21                      {
22                              try
23              {
24                              InputStream in = new FileInputStream(fin);
25              OutputStream out = new FileOutputStream(fout);
26              int content;
27              while((content=in.read())!=-1)
28                        out.write(content);
29               in.close();
30               out.close();
31              }
32              catch(IOException e)
33              {
34                e.printStackTrace();
35              }
36
37                      }
38              }
39      }
40  }
```

§3 FilenameFilter

FilenameFilter 能够对文件扩展名进行过滤, 限制 list() 方法返回的文件, 使它仅返回那些与一定的文件名方式或者过滤器 (filter) 相匹配的文件。为达到这样的目的, 必须使用 list() 方法的带参数的重载形式:

- public String[] list()　返回由此抽象路径名所表示的目录中的文件和目录的名称所组成字符串数组。
- public String[] list(FilenameFilter filter)　返回由包含在目录中的文件和目录的名称所组成的字符串数组，这一目录是通过满足指定过滤器的抽象路径名来表示的。
- public File[] listFiles()　返回一个抽象路径名数组，这些路径名表示此抽象路径名所表示目录中的文件。
- public File[] listFiles(FileFilter filter)　返回表示此抽象路径名所表示目录中的文件和目录的抽象路径名数组，这些路径名满足特定过滤器。

public String[] list(FilenameFilter filter) 在该形式中,filter 是一个实现了 FilenameFilter 接口的类的对象, 该接口中有一个 accept() 方法, 只有该方法判断为 true 的文件名会被 list 方法返回。所以可以在 accept 方法中设置筛选规则。

```
import java.io.File;
import java.io.IOException;
import java.util.Scanner;
import java.io.InputStream;
import java.io.OutputStream;
import java.io.FileInputStream;
import java.io.FileOutputStream;

public class Lesson34
{
	public static void main(String[] args)
	{
		String infile = args[0];
		File fin = new File(infile);
		String outfile = args[1];
		File fout = new File(outfile);
		if(fin.exists())
		{
			if(fin.isFile())
			{
				try
		{
				InputStream in = new FileInputStream(fin);
		OutputStream out = new FileOutputStream(fout);
		int content;
		while((content=in.read())!=-1)
			out.write(content);
		 in.close();
		 out.close();
		}
		catch(IOException e)
		{
		  e.printStackTrace();
		}

			}
		}
	}
}
```

3.2 输入输出流 (I/O)

§1 字节流

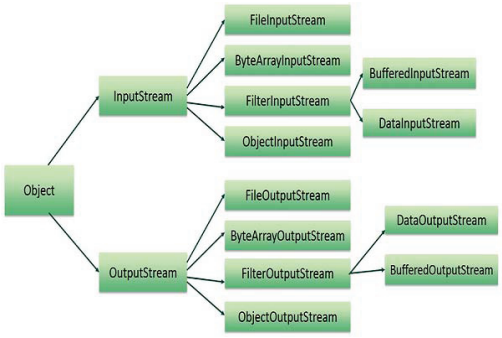


图 1: Java 的 IO

在 Java 中对文件内容进行读写的类成为输入、输出流, 这些流的来源与目的都可以是文件, 也可以是网络连接, 甚至是内存块。Java 的流主要包括字节流和字符流, 其中字节流以 8 位字节为一个输入、输出单位。抽象类 InputStream 和 OutputStream 构成有层次结构的输入或输出类的基础, 最常用的字节流类是 FileInputStream 和 FileOutputStream。

§2 字符流

字节流可以处理任何类型输入/输出操作, 但是不能直接操作 Unicode 字符,Unicode 使用两个字节来表示一个字符, 即一个字符占 16 位。字符流可以支持 Unicode 字符,Reader 和 Writer 抽象类是字符流流的基类。Reader 和 Writer 类也有较多的子类, 与字节流类似, 它们用来创建具体的字符流对象进行 I/O 操作, 字符流的读写等方法与字节流的相应方法类似, 但读写对象使用的是字符。

3.3 读写 TXT 文件和 CSV 文件

§1 读 TXT

FileInputStream 流可以读入 TXT 文件，并将读入的流转化为字节流，通过 JTextArea 显示 Txt 文件中的内容。

```
1  import java.io.File;
2  import javax.swing.JFrame;
3  import javax.swing.JPanel;
4  import javax.swing.JTextField;
5  import javax.swing.JTextArea;
6  import javax.swing.JLabel;
7  import java.awt.event.ActionListener;
8  import java.awt.event.ActionEvent;
9  import javax.swing.JButton;
10 import java.awt.BorderLayout;
11
```

```
12 class MyFrame extends JFrame
13 {
14     JPanel pan,pan1,pan2;
15     JButton btn,btn1;
16     JTextField tf;
17     JTextArea ta;
18     JLabel pathlab;
19     BorderLayout blay;
20     MyFrame(String title)
21         {
22                 super(title);
23                 setSize(500,500);
24                 setLocationRelativeTo(null);
25         blay = new BorderLayout();
26                 pan = new JPanel();
27         pan1 = new JPanel();
28         pan2 = new JPanel();
29         pathlab = new JLabel("Pls input Directory");
30         tf = new JTextField(15);
31         btn = new JButton("Click");
32         btn1 = new JButton("Clean");
33         ta = new JTextArea(20,30);
34         setContentPane(pan);
35         pan.setLayout(blay);
36         pan.add(pan1,BorderLayout.NORTH);
37         pan1.add(tf);
38         pan1.add(btn);
39         pan1.add(btn1);
40         pan.add(pan2,BorderLayout.CENTER);
41         pan2.add(ta);
42         ta.setLineWrap(true);
43         ActionListener action = new MyAction();
44         btn.addActionListener(action);
45         btn1.addActionListener(action);
46         setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
47         setVisible(true);
48         }
49 
50         private class MyAction implements ActionListener
51         {
52         public void actionPerformed(ActionEvent e)
53         {
54                 if(e.getSource()==btn)
55                 {
56            String dirstr = tf.getText();
57            File f = new File(dirstr);
```

```
58            if(f.isDirectory())
59            {
60            String[] str = f.list();
61            for(int i=0;i<str.length;i++)
62            ta.append(str[i]+"\r\n");
63            }
64            if(f.isFile())
65            {
66            ta.append("Input Directory Path");
67            }
68                }
69        if(e.getSource()==btn1)
70        {
71            tf.setText("");
72            ta.setText("");
73        }
74        }
75    }
76 }
77 public class Lesson35
78 {
79        public static void main(String[] args)
80        {
81      MyFrame frm = new MyFrame("list all file in directory");
82        }
83 }
```

§2 写 TXT 文件

FileInputStream 读入 TXT 文件内容，并将读入的流转化为字节流，通过 FileOutputStream 流写入文件。

```
1  import java.io.File;
2  import java.io.FilenameFilter;
3  import javax.swing.JFrame;
4  import javax.swing.JPanel;
5  import javax.swing.JTextField;
6  import javax.swing.JTextArea;
7  import javax.swing.JLabel;
8  import java.awt.event.ActionListener;
9  import java.awt.event.ActionEvent;
10 import javax.swing.JButton;
11 import java.awt.BorderLayout;
12 import java.io.IOException;
13 import java.util.Date;
14
```

```
15 class MyFrame extends JFrame
16 {
17     JPanel pan,pan1,pan2;
18     JButton btn,btn1,btn2;
19     JTextField tf;
20     JTextArea ta;
21     JLabel pathlab;
22     BorderLayout blay;
23     MyFrame(String title)
24         {
25                 super(title);
26                 setSize(500,500);
27                 setLocationRelativeTo(null);
28         blay = new BorderLayout();
29                 pan = new JPanel();
30         pan1 = new JPanel();
31         pan2 = new JPanel();
32         pathlab = new JLabel("Pls input Directory");
33         tf = new JTextField(15);
34         btn = new JButton("Click");
35         btn1 = new JButton("Create the file");
36         btn2 = new JButton("Create Directory");
37         ta = new JTextArea(20,30);
38         setContentPane(pan);
39         pan.setLayout(blay);
40         pan.add(pan1,BorderLayout.NORTH);
41         pan1.add(tf);
42         pan1.add(btn);
43         pan1.add(btn1);
44         pan1.add(btn2);
45         pan.add(pan2,BorderLayout.CENTER);
46         pan2.add(ta);
47         ta.setLineWrap(true);
48         ActionListener action = new MyAction();
49         btn.addActionListener(action);
50         btn1.addActionListener(action);
51         btn2.addActionListener(action);
52         setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
53         setVisible(true);
54         }
55 
56         private class MyAction implements ActionListener
57         {
58         public void actionPerformed(ActionEvent e)
59         {
60                 if(e.getSource()==btn)
```

```
61              {
62           String dirstr = tf.getText();
63           File f = new File(dirstr);
64           if(f.isFile())
65           {
66           ta.append(f.getName()+"\r\n");
67           ta.append(f.getPath()+"\r\n");
68           ta.append(f.getAbsolutePath()+"\r\n");
69           ta.append(f.getParent()+"\r\n");
70           java.text.SimpleDateFormat df = new java.text.SimpleDateFormat("yyyy-MM-dd
                 HH:mm:ss.SSS");
71           String dateTime=df.format(new Date(f.lastModified()));
72           ta.append(dateTime+"\r\n");
73           ta.append(Long.toString(f.length())+"\r\n");
74           }
75              }
76       if(e.getSource()==btn1)
77       {
78           String dirstr = tf.getText();
79           File f2 = new File(dirstr);
80           File f3 = new File(f2.getParent(),f2.getName());
81           try
82           {
83           f3.createNewFile();
84           }
85           catch(IOException error)
86           {error.printStackTrace();}
87       }
88       if(e.getSource()==btn2)
89       {
90           String dirstr = tf.getText();
91           File f1 = new File(dirstr);
92           f1.mkdir();
93       }
94       }
95   }
96 }
97 public class Lesson36
98 {
99       public static void main(String[] args)
100      {
101     MyFrame frm = new MyFrame("Create the File");
102      }
103 }
```

§3 读写 CSV 文件

CSV（逗号分隔值）是一种用来存储数据的纯文本文件，通常都是用于存放电子表格或数据的一种文件格式。CSV 文件中，通常以，分割不同的字符串，通常用 BufferedReader 流读入 CSV 文件，用 BufferedWriter 写入 CSV 文件。

```
import java.io.File;
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.JTextField;
import javax.swing.JLabel;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import java.awt.GridLayout;
import javax.swing.JButton;

class MyFrame extends JFrame
{
	private GridLayout glay;
	private JPanel pane,toppane,middlepane;
	private JLabel pathlab,filelab;
	private JTextField tf,tf1;
	private JButton btn;

	MyFrame(String title)
	{
		super(title);
		setSize(300,300);
		setLocationRelativeTo(null);
		glay = new GridLayout(3,2);
		pane = new JPanel();
	pathlab = new JLabel("Pls input Path");
	filelab = new JLabel("Pls input Directory");
	tf = new JTextField(15);
	tf1 = new JTextField(15);
	btn = new JButton("Click");
	setContentPane(pane);
	setLayout(glay);
	pane.add(pathlab);
	pane.add(tf);
	pane.add(filelab);
	pane.add(tf1);
	pane.add(btn);
	ActionListener action = new MyAction();
	btn.addActionListener(action);
	setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
	setVisible(true);
	}
```

```
43
44          private class MyAction implements ActionListener
45          {
46          public void actionPerformed(ActionEvent e)
47          {
48                  if(e.getSource()==btn)
49                  {
50              String dirname = tf.getText();
51              String filename = tf1.getText();
52              File f = new File(dirname,filename);
53              f.mkdir();
54          }
55          }
56      }
57  }
58  public class Lesson37
59  {
60          public static void main(String[] args)
61          {
62        MyFrame frm = new MyFrame("make directory");
63          }
64  }
```

3.4 课后作业

§1 添加学生数据

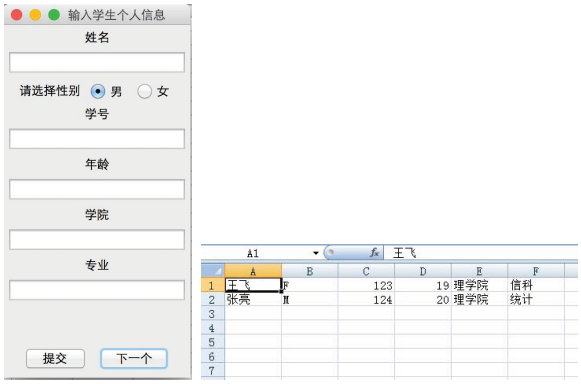


图 2: 添加学生记录

§2 文件复制器

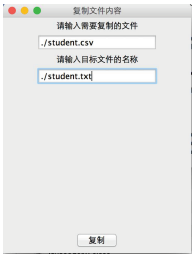


图 3: 文件内容复制工具

4 SQL 基本概念

1、教学内容

(1) MySQL 的安装和配置
(2) MySQL 中文字符支持
(3) SQL 语言简介

2、教学重点

MySQL 中文字符环境配置

3、教学难点

SQL 语言简介

教学内容

4.1 MySQL 安装和配置

§1.1 Windows

1、下载 MySQL 数据库
http://ftp.ntu.edu.tw/pub/MySQL/Downloads/MySQL-5.0/
安装 MySQL 的 no-install 版本，下载 msql-noinstall-5.1.60-winx64.zip，解压到指定文件夹下。
2、进入根目录下选择一个配置文件 my-medium.ini，将修改后的文件另存为 my.ini。

```
my-huge.ini
my-innodb-heavy-4G.ini
my-large.ini
my-medium.ini
my-small.ini
```

3、在 my.ini 文件中添加 basedir、datadir、中文字符和事务等信息。
在 [client]

```
#password = your_password
default-character-set=gb2312
port      = 3306
socket    = MySQL
```

在 [mysqld] 的 Server 端设置

```
basedir="d:\mysql_5.1.60"
datadir="d:\mysql_5.1.60\data"
default-character-set=gb2312
character-set-server=gb2312
init_connect='set names gb2312'
default-storage-engine=innodb
```

4、启动服务以管理员用户运行 cmd 命令，进入 MySQL 的安装路径新建 MySQL 服务，并启动 MySQL。

```
mysqld --install
net start mysql
```

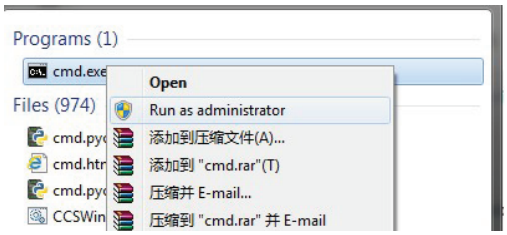


图 1: 管理员用户运行

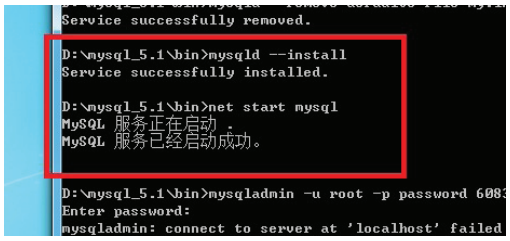


图 2: 在 Window 服务中安装 MySQL

5、停止 MySQL 服务

```
net stop mysql
```

6、从控制面板中删除 MySQL 服务

图 3: 控制面板启动 MySQL 服务

```
sc delete mysql
```

7、安装过程出现 1067 错误。

```
mysqld --remove
```

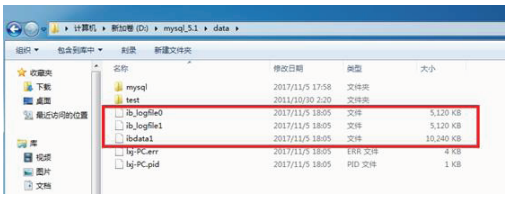


图 4: 删除 data 目录下的三个文件

§1.2 Ubuntu

1、下载 MySQL 安装包

http://ftp.ntu.edu.tw/pub/MySQL/Downloads/MySQL-5.0/

下载 mysql-5.5.27-linux-2.6-x86_64.tar.gz 文件。

2、解压，移动到/usr/local/mysql 文件夹中

```
tar -zxvf mysql-5.5.27-linux-2.6-x86_64.tar.gz
sudo mv mysql-5.5.27-linux-2.6-x86_64 /usr/local/mysql
```

3、添加用户组、用户，并对用户组、用户赋权限

```
adduser mysql
groupadd mysql
sudo chown -R mysql:mysql mysql
```

进入 mysql 目录，修改 mysql 目录的拥有者为 mysql 用户

```
cd /usr/local/mysql
sudo chown -R mysql .
sudo chgrp -R mysql .
```

4、安装数据库

```
/usr/local/mysql$ sudo scripts/mysql_install_db --user=mysql
```

```
lxj@nbut:/usr/local/mysql$ sudo scripts/mysql_install_db --user=mysql
Installing MySQL system tables...
OK
Filling help tables...
OK

To start mysqld at boot time you have to copy
support-files/mysql.server to the right place for your system

PLEASE REMEMBER TO SET A PASSWORD FOR THE MySQL root USER !
To do so, start the server, then issue the following commands:

./bin/mysqladmin -u root password 'new-password'
./bin/mysqladmin -u root -h nbut password 'new-password'

Alternatively you can run:
./bin/mysql_secure_installation

which will also give you the option of removing the test
databases and anonymous user created by default. This is
strongly recommended for production servers.

See the manual for more instructions.

You can start the MySQL daemon with:
cd . ; ./bin/mysqld_safe &

You can test the MySQL daemon with mysql-test-run.pl
cd ./mysql-test ; perl mysql-test-run.pl

Please report any problems with the ./bin/mysqlbug script!
```

如果出现以下错误信息

```
Installing MySQL system tables..../bin/mysqld: error while loading shared
   libraries: libaio.so.1: cannot open shared object file: No such file or
   directory
```

安装以下依赖包

```
sudo apt-get install libaio-dev
```

5、启动 MySQL 数据库服务

```
sudo ./support-files/mysql.server start
```

```
lxj@lxj-ThinkPad-X200:~$ sudo /usr/local/mysql/bin/mysqld_safe --user=root &
[1] 2525
lxj@lxj-ThinkPad-X200:~$ nohup: ignoring input and redirecting stderr to stdout
Starting mysqld daemon with databases from /usr/local/mysql/data
```

图 5: 启动 MySQL 数据库

6、修改 MySQL 初始密码

```
./bin/mysqladmin -u root password '6083836'
```

```
lxj@lxj-ThinkPad-X200:~$ sudo service mysql.server status
[sudo] password for lxj:
Sorry, try again.
[sudo] password for lxj:
Usage: /etc/init.d/mysql.server  {start|stop|restart|reload}  [ MySQL server opt
ions ]
lxj@lxj-ThinkPad-X200:~$ ps -A | grep mysql
 2526 pts/1    00:00:00 mysqld_safe
 2550 pts/1    00:00:00 mysqld
lxj@lxj-ThinkPad-X200:~$
```

图 6: 启动 MySQL 服务

7、登录数据库

```
/usr/local/mysql/bin# ./mysql -u root -p
```

8、配置环境变量

```
sudo vim /etc/profile
export MYSQL_HOME=/usr/local/mysql
export PATH=$MYSQL_HOME/bin:$PATH
source /etc/profile
```

9、添加 MySQL 自启动

将 mysql.server 文件复制到 init.d 文件夹中

```
sudo cp /usr/local/mysql/support-files/mysql.server /etc/init.d/mysql
lxj@nbut:/usr/local/mysql$
```

赋权限，设置自启动

```
sudo chmod +x /etc/init.d/mysql
cd /etc/init.d/
sudo update-rc.d mysql defaults
```

移除自启动

```
update-rc.d mysql remove
```

§1.3 Mac OSX

1、下载 MySQL 安装包

http://ftp.ntu.edu.tw/pub/MySQL/Downloads/MySQL-5.0/

下载 mysql-5.1.45-osx10.6-x86_64.dmg 文件。

2、Stop MySQL 服务，设置中文 utf-8 字符集

图 7: 关闭 MySQL 服务

复制/usr/local/mysql/support-files 下任意一个 cnf 配置文件到/etc/my.cnf。

```
cp /usr/local/mysql/support-files/my-medium.cnf /etc/my.cnf
```

3、编辑 my.cnf 文件

[client] 后面添加

```
default-character-set = utf8
```

[mysqld] 后面添加

```
character-set-server = utf8
collation-server = utf8_general_ci
init_connect='SET NAMES utf8'
default-storage-engine = INNODB
```

4.2 启动 MySQL 数据库

§2.1 启动 MySQL 数据库

1、切换到 root 用户下，并设置 root 用户密码，然后直接回车。

```
mysqladmin -u root -p password 1234567
Enter password:
```

§2.2 创建数据库

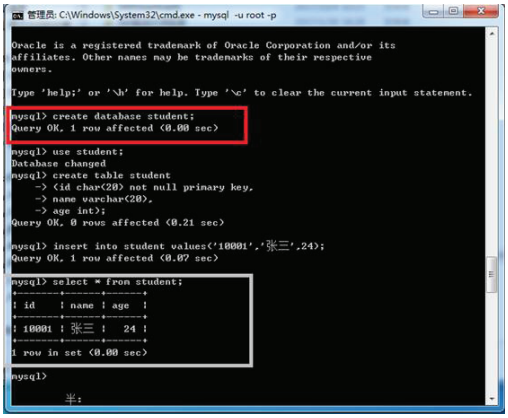


图 8: 新建数据库

1、创建书店数据库

```
create databse bookstore;
```

2、创建 book 数据表

```
create table books
(book_id INT,
title varchar(50),
author varchar(50)
);
```

3、查看表的结构

```
describe books
```

4、修改表的结构

```
alter tables books
change column book_id book_id int auto_crementprimary key,
change column author author_id int,
```

```
add column description Text,
add column genre ENUM('novel','poetry','drama'),
add column publisher_id int,
add column pub_year varchar(4),
add column isbn varchar(20);
```

5、导入数据

```
load data infile 'D:/Course/Java/Java_program/Markdown/csv4/books.csv' into
    table books
    -> fields terminated by ','
    -> optionally enclosed by '"'
    -> lines terminated by '\n';
```

替换导入数据

```
load data infile 'D:/Course/Java/Java_program/Markdown/csv4/books.csv' replace
    into table books
    -> fields terminated by ','
    -> optionally enclosed by '"'
    -> lines terminated by '\n';
```

如果出现中文字符问题，使用以下语句显示 MySQL 的字符。

```
SHOW VARIABLES LIKE 'character%'
```

6、导出数据

```
select * from books into outfile
    'D:/Course/Java/Java_program/Markdown/csv4/books1.csv'
    -> fields terminated by ','
    -> optionally enclosed by '"'
    -> lines terminated by '\n';
```

4.3 删除 MySQL 数据库

§3.1 Window

1、以管理员用户运行 cmd 命令，进入 MySQL 的 bin 路径。

```
net stop mysql
mysqld --remove
sc delete mysql
```

2、删除 MySQL 的安装文件夹。

§3.2 Ubuntu

1、卸载 mysql

```
sudo apt-get autoremove --purge mysql-server-5.0
sudo apt-get remove mysql-server
sudo apt-get autoremove mysql-server
sudo apt-get remove mysql-common
```

2、清理残留数据

```
dpkg -l |grep ^rc|awk '{print $2}' |sudo xargs dpkg -P
```

§3.3 MacOSX

关闭 System Preferences 的 MySQL 启动项，在 Terminal 中删除 MySQL 文件

```
sudo rm /usr/local/mysql
sudo rm -rf /usr/local/mysql*
sudo rm -rf /Library/StartupItems/MySQLCOM
sudo rm -rf /Library/PreferencePanes/My*
rm -rf ~/Library/PreferencePanes/My*
sudo rm -rf /Library/Receipts/mysql*
sudo rm -rf /Library/Receipts/MySQL*
```

编辑/etc/hostconfig 文件，删除 MYSQLCOM=-YES-行。
如果安装了其他版本的 MySQL，需要以下命令删除 MySQL 的残留文件。

```
sudo rm -rf /var/db/receipts/com.mysql.mysql*
```

4.4 课后作业

1、新建 Student 数据库，并将测试数据导入 Student 数据库中。
2、查询性别为男性同学的信息。
3、统计生源地是浙江的学生数。

5 MySQL 数据库

1、教学内容

(1) MySQL 服务器端、客户端程序
(2) MySQL 的常用命令
(3) 根据学生、教师、课程数据，建立教学管理系统。

2、教学重点

数据库的设计，ER 模型到 CLASS 模型的转化。

3、教学难点

MySQL 数据的导入、导出，ER 表的设计，ER-类图的转化。

教学内容

5.1 MySQL 服务器和客户程序

§1.1 mysql 服务器

1、mysqld

在 Linux 系统中，mysql 配置文件在/etc/my.cnf，在 windows 系统中，mysql 的配置文件在 my.ini 文件中。

启动 mysqld 可以更改服务器的各种行为。

§1.2 mysql 客户程序

2、mysql -u root -p

5.2 MySQL 的常用命令

§2.1 mysqladmin

mysqladmin 允许通过命令行执行 MySQL 服务器的管理任务，可以使用它检查服务器的状态和设置、刷新表、修改密码、关闭服务器等。

§2.2 mysqldump

mysqldump 可以导出 MySQL 中的数据和表结构。

§2.3 mysqlimport

5.3 数据的导入和导出

§3.1 Windows 数据的导入和导出

1、利用 load 命令导入 txt 数据

```
mysql> load data infile "D:/Course/Java/Java_program/code4/user.txt" into table
    user;
```

2、利用 load 命令导入 csv 文件

```
mysql> load data infile 'D:/Course/Java/Java_program/code4/user.csv' into table
    person
    -> FIELDS TERMINATED BY ','
    -> OPTIONALLY ENCLOSED BY '"'
    -> LINES TERMINATED BY '\n';
```

3、利用 select 命令将数据导出到 csv 文件中

```
mysql> SELECT * FROM person INTO OUTFILE
    'D:/Course/Java/Java_program/code4/user.csv'
    -> FIELDS TERMINATED BY ','
    -> OPTIONALLY ENCLOSED BY '"'
    -> LINES TERMINATED BY '\n';
```

4、利用 source 命令运行 sql 脚本文件

```
mysql> source D:/Course/Java/Java_program/code4/user.sql;
```

§3.2 Ubuntu 数据的导入和导出

§3.3 Mac oSX 数据的导入和导出

1、Mac OSX 下的导出文件夹进行递归权限赋值

```
sudo chmod -R 777 /Users/lili/Course/Java
```

2、Mac OSX 下导出数据，文件导出字符编码设置为 gbk

```
select * from stu into outfile
    '//Users//lili//Course//Java//Java_program//Lesson3//code3//xinke.csv'
    CHARACTER SET gbk fields terminated by ',' optionally enclosed by '"' lines
    terminated by '\r';
```

3、Excel for Macintosh 的行结尾符号为'\r' (0x0d), I used the clause LINES TERMINATED BY '\r' in my command.

```
load data infile
    '//Users//lili//Course//Java//Java_program//Lesson3//code3//xinke1.csv'
    replace into table stu CHARACTER SET gbk fields terminated by ',' optionally
    enclosed by '"' lines terminated by '\r';
```

4、Mac OSX 下导出数据，文件导出字符编码设置为 utf8

```
mysql> use nbut;
Database changed
mysql> select * from stu into outfile '/Users/lili/Course/out1.csv' fields
    terminated by ',' optionally enclosed by '"' lines terminated by '\n';
Query OK, 61 rows affected (0.00 sec)
```

5.4 MySQL 基础

§4.1 登录 MySQL 服务器

```
mysql -u root -p
```

§4.2 创建数据库和表

1、创建 nbut 数据库。

```
create database nbut;
```

2、创建 student 表。

```
use nbut;
create table student(
xuehao char(11) not null primary key,
xingming varchar(12) not null,
sex char(2) not null,
department varchar(20) not null,
zhuanye varchar(20) not null,
banji varchar(16) not null,
nianji varchar(8) not null,
birthday Date);
```

§4.3 插入数据

1、插入记录到 student 表中。

```
desc student;
insert into student
values('14480010103','李飞','男','理学院','信息与计算科学','信科14-1','2014','1996-08-17');
select * from student into outfile 'G:\\student.csv' fields terminated by ','
    optionally enclosed by '"' lines terminated by '\n';
```

2、从 student.csv 文件中导入到 student 表中。

```
load data infile 'G:\\student.csv' into table student fields terminated by ','
    optionally enclosed by '"' lines terminated by '\n';
```

§4.4 选择数据

1、select 选择数据

```
select * from student;
```

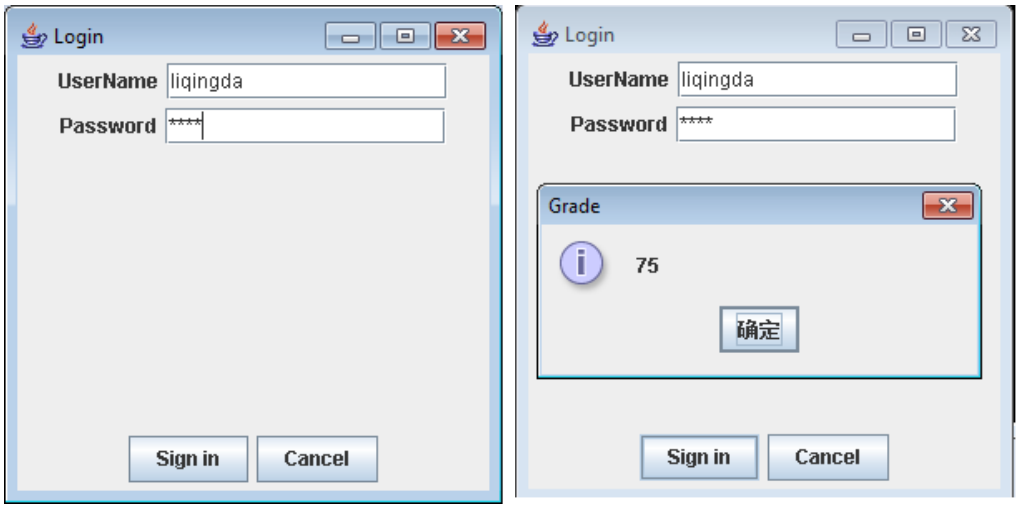
§4.5 排序、限制和分组

5.5 课后作业

1、新建 Student 数据库，并将测试数据导入 Student 数据库中。

```
+--------------+-------------+------+-----+---------+-------+
| Field        | Type        | Null | Key | Default | Extra |
+--------------+-------------+------+-----+---------+-------+
| stu_id       | varchar(20) | NO   | PRI |         |       |
| stu_name     | varchar(25) | YES  |     | NULL    |       |
| stu_password | varchar(25) | YES  |     | NULL    |       |
| stu_grade    | varchar(25) | YES  |     | NULL    |       |
+--------------+-------------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

2、查询性别为男性同学的信息。



6 JDBC

1、教学内容

(1) JDBC 数据库程序
(2) JDBC 的七个步骤
(3) 数据查询、插入、删除和更新

2、教学重点

JDBC 的七个步骤。

3、教学难点

JDBC 的七个步骤以及数据库查询结果的显示。

教学内容

6.1 JDBC 查询数据的七个步骤

§1.1 装载驱动程序

1、导入 MySQL 的 jar 包。

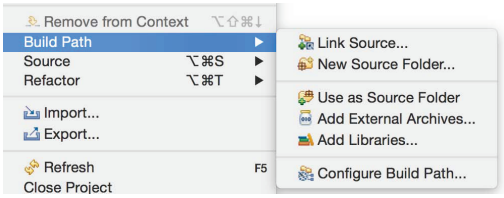


图 1: 导入 MySQL 驱动

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
```

§1.2 定义连接的 URL

```
String url = "jdbc:mysql://127.0.0.1:3306/nbut";
```

§1.3 建立连接

```
Class.forName("com.mysql.jdbc.Driver").newInstance();
Connection con = DriverManager.getConnection(url,"root","6083836");
```

§1.4 创建 Statement 对象

```
st = con.createStatement();
rs = st.executeQuery("show tables");
```

§1.5 执行查询

```
rs = st.executeQuery("show tables");
```

§1.6 处理结果集

```
while(rs.next())
{
rs.getString(1);
rs.getInt(2);
}
```

§1.7 关闭查询

```
rs.close();
st.close();
```

6.2 通过 GUI 访问 MySQL 数据库

§2.1 Statement 访问数据库

```
1  package lesson61;
2
3  import java.sql.Connection;
4  import java.sql.DriverManager;
5  import java.sql.ResultSet;
6  import java.sql.SQLException;
7  import java.sql.Statement;
8
9  public class Lesson61
10 {
11        public static void main(String[] args) throws InstantiationException,
               IllegalAccessException, ClassNotFoundException
12        {
13               Connection con = null;
14               Statement st = null;
15               ResultSet rs = null;
16               String url = "jdbc:mysql://localhost:3306/nbut";
17               String user = "root";
18               String password = "6083836";
19               try {
20 Class.forName("com.mysql.jdbc.Driver").newInstance();
21 con = DriverManager.getConnection(url, user, password);
22 //st = con.prepareStatement("select * from people where name = ?");
23 st = con.createStatement();
24 rs = st.executeQuery("select * from people");
25
26 while (rs.next())
27 {
28 System.out.println(rs.getString(1));
29 System.out.println(rs.getString(2));
30 System.out.println(rs.getString(3));
31 System.out.println(rs.getString(4));
32 System.out.println(rs.getString(5));
33 }
34 rs.close();
35 st.close();
36 con.close();
37 }
38               catch (SQLException ex) {
39                      ex.printStackTrace();
40  }
41  }
```

```
42 }
```

§2.2 PreparedStatement 访问数据库

```
1  package lesson62;
2
3  import java.sql.Connection;
4  import java.sql.DriverManager;
5  import java.sql.ResultSet;
6  import java.sql.SQLException;
7  import java.sql.PreparedStatement;
8
9  public class Lesson62
10 {
11        public static void main(String[] args) throws InstantiationException,
               IllegalAccessException, ClassNotFoundException
12        {
13              Connection con = null;
14              PreparedStatement st = null;
15              ResultSet rs = null;
16              String url = "jdbc:mysql://localhost:3306/nbut";
17              String user = "root";
18              String password = "6083836";
19              try {
20                    Class.forName("com.mysql.jdbc.Driver").newInstance();
21                    con = DriverManager.getConnection(url, user, password);
22                    st = con.prepareStatement("select * from people where name = ?");
23                    st.setString(1,"王风");
24                    rs = st.executeQuery();
25                    while (rs.next())
26                    {
27                          System.out.println(rs.getString(1));
28                          System.out.println(rs.getString(2));
29                          System.out.println(rs.getString(3));
30                          System.out.println(rs.getString(4));
31                          System.out.println(rs.getString(5));
32                    }
33                    rs.close();
34                    st.close();
35                    con.close();
36 }
37              catch (SQLException ex) {
38                    ex.printStackTrace();
39  }
40  }
41  }
```

§2.3 数据的添加

```
package lesson63;
import java.sql.SQLException;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.math.BigDecimal;

public class Lesson63
{
	public static void main(String[] args) throws InstantiationException,
	    IllegalAccessException
	{
		String url = "jdbc:mysql://localhost:3306/nbut";
		String username = "root";
		String password = "6083836";
		Connection con = null;
		PreparedStatement st;
		ResultSet rs =null;

		try {
			Class.forName("com.mysql.jdbc.Driver").newInstance();
			con = DriverManager.getConnection(url, username, password);
			st = con.prepareStatement("insert into people values(?,?,?,?,?)");
			st.setString(1,"1002");
			st.setString(2,"李杨");
			st.setInt(3,45);
			st.setString(4,"杭州");
			st.setBigDecimal(5,new BigDecimal(45.5));
			st.executeUpdate();
			st = con.prepareStatement("select * from people");
			rs = st.executeQuery();
			while (rs.next())
			{
				System.out.println(rs.getString(1));
				System.out.println(rs.getString(2));
				System.out.println(rs.getString(3));
				System.out.println(rs.getString(4));
				System.out.println(rs.getString(5));
			}
			st.close();
			con.close();
```

```
}
            catch(ClassNotFoundException ex)
            {
                  ex.printStackTrace();
            }
            catch (SQLException ex) {
                  ex.printStackTrace();
 }
 }
 }
```

§2.4 数据的删除

```
package lesson64;


import java.sql.SQLException;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.DriverManager;
import java.sql.PreparedStatement;

public class Lesson64
{
      public static void main(String[] args) throws InstantiationException,
          IllegalAccessException
      {
            String url = "jdbc:mysql://localhost:3306/nbut";
            String username = "root";
            String password = "6083836";
            Connection con = null;
            PreparedStatement st;
            ResultSet rs =null;

            try {
                  Class.forName("com.mysql.jdbc.Driver").newInstance();
                  con = DriverManager.getConnection(url, username, password);
                  st = con.prepareStatement("delete from people where stu_name = ?");
                  st.setString(1, "李杨");
                  st.executeUpdate();
                  st = con.prepareStatement("select * from people");
                  rs = st.executeQuery();
                  while (rs.next())
                  {
                        System.out.println(rs.getString(1));
                        System.out.println(rs.getString(2));
```

```
33                          System.out.println(rs.getString(3));
34                          System.out.println(rs.getString(4));
35                          System.out.println(rs.getString(5));
36                  }
37                  st.close();
38                  con.close();
39  }
40              catch(ClassNotFoundException ex)
41              {
42                      ex.printStackTrace();
43              }
44              catch (SQLException ex) {
45                      ex.printStackTrace();
46   }
47   }
48   }
```

§2.5 数据的更新

```
1   package lesson65;
2
3   import java.sql.SQLException;
4   import java.sql.Connection;
5   import java.sql.ResultSet;
6   import java.sql.DriverManager;
7   import java.sql.PreparedStatement;
8
9   public class Lesson65
10  {
11          public static void main(String[] args) throws InstantiationException,
                IllegalAccessException
12          {
13                  String url = "jdbc:mysql://localhost:3306/nbut";
14                  String username = "root";
15                  String password = "6083836";
16                  Connection con = null;
17                  PreparedStatement st;
18                  ResultSet rs =null;
19
20                  try {
21                          Class.forName("com.mysql.jdbc.Driver").newInstance();
22                          con = DriverManager.getConnection(url, username, password);
23                          st = con.prepareStatement("update people set stu_name = ? where stu_name =
                                ?");
24                          st.setString(1, "王理想");
25                          st.setString(2, "李杨");
```

```
                    st.executeUpdate();
                    st = con.prepareStatement("select * from people");
                    rs = st.executeQuery();
                    while (rs.next())
                    {
                           System.out.println(rs.getString(1));
                           System.out.println(rs.getString(2));
                           System.out.println(rs.getString(3));
                           System.out.println(rs.getString(4));
                           System.out.println(rs.getString(5));
                    }
                    st.close();
                    con.close();
}
              catch(ClassNotFoundException ex)
              {
                    ex.printStackTrace();
              }
              catch (SQLException ex) {
                    ex.printStackTrace();
 }
 }
 }
```

6.3 MetaData

§3.1 数据库表的元数据

```

import java.sql.ResultSetMetaData;
import java.sql.Statement;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;

public class Lesson66 {

       public static void main(String[] args) throws SQLException, InstantiationException,
           IllegalAccessException
       {
              String url = "jdbc:mysql://localhost:3306/nbut";
              String user = "root";
              String password = "1234567";
```

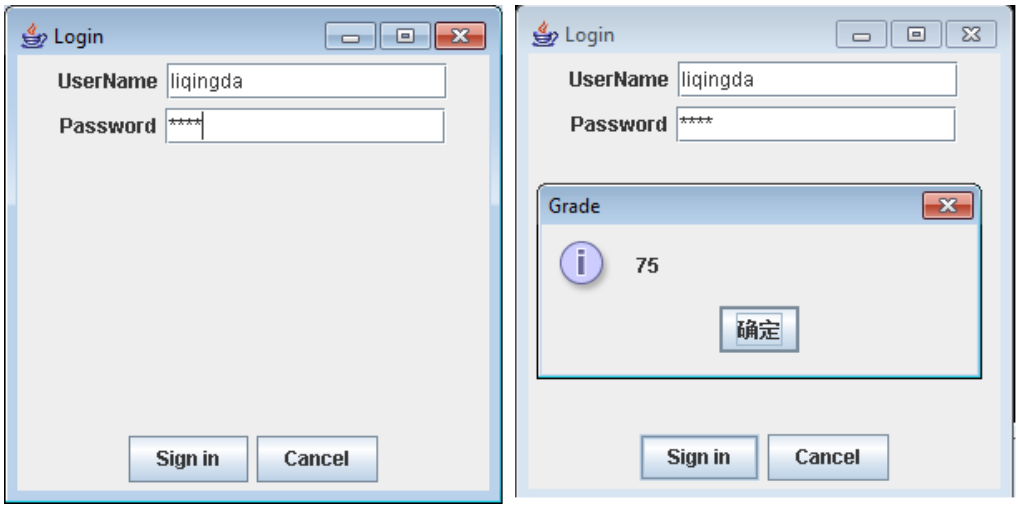
```
17              Connection con;
18              Statement st;
19              ResultSet rs;
20              ResultSetMetaData ma;
21              try
22              {
23                      Class.forName("com.mysql.jdbc.Driver").newInstance();
24                      con = DriverManager.getConnection(url, user, password);
25                      st = con.createStatement();
26                      rs = st.executeQuery("select * from student");
27                  ma = rs.getMetaData();
28                      for(int i=1;i<=ma.getColumnCount();i++)
29                      {
30                              System.out.print(ma.getColumnName(i)+" ");
31                      }
32
33              }
34
35              catch(ClassNotFoundException e)
36              {
37                      e.printStackTrace();
38              }
39      }
40  }
```

6.4 课后作业

1、访问 Teacher 数据库内容

```
+--------------+-------------+------+-----+---------+-------+
| Field        | Type        | Null | Key | Default | Extra |
+--------------+-------------+------+-----+---------+-------+
| stu_id       | varchar(20) | NO   | PRI |         |       |
| stu_name     | varchar(25) | YES  |     | NULL    |       |
| stu_password | varchar(25) | YES  |     | NULL    |       |
| stu_grade    | varchar(25) | YES  |     | NULL    |       |
+--------------+-------------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

2、数据库的增、删、查操作



7 MySQL Workbench 数据建模

1、教学内容

(1) Entity Relationship Model
(2) UML
(3) ÀíœâJava MailÊÕ · ¢ÓÊŒÞµÄ»úÖÆ¡£

2、教学难点

ER 模型、UML 模型、

3、教学重点

JavaµÄSocketÀà¡¢URLÀàÒÔŒ°MailÀàµÄÊ¹ÓÃ¡£

教学内容

7.1 Entity Relationship Model

网络编程是指编写运行在多个设备（计算机）的程序，这些设备都通过网络连接起来。java.net 包中 J2SE 的 API 包含有类和接口，它们提供低层次的通信细节。你可以直接使用这些类和接口，来专注于解决问题，而不用关注通信细节。java.net 包中提供了两种常见的网络协议的支持:

- 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。

7.2 Socket URL

- Socket 编程这是使用最广泛的网络概念。
- URL 处理

§1 Socket

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。

```
import java.net.*;
import java.io.*;

public class GreetingClient {
   public static void main(String [] args) {
      String serverName = args[0];
      int port = Integer.parseInt(args[1]);
      try {
         System.out.println("Connecting to " + serverName + " on port " + port);
         Socket client = new Socket(serverName, port);
         System.out.println("Just connected to " + client.getRemoteSocketAddress());
         OutputStream outToServer = client.getOutputStream();
         DataOutputStream out = new DataOutputStream(outToServer);
         out.writeUTF("Hello from " + client.getLocalSocketAddress());
         InputStream inFromServer = client.getInputStream();
         DataInputStream in = new DataInputStream(inFromServer);
         System.out.println("Server says " + in.readUTF());
```

```
18          client.close();
19       }catch(IOException e) {
20          e.printStackTrace();
21       }
22    }
23 }
```

```
1  import java.net.*;
2  import java.io.*;
3
4  public class GreetingServer extends Thread {
5     private ServerSocket serverSocket;
6     public GreetingServer(int port) throws IOException {
7        serverSocket = new ServerSocket(port);
8        serverSocket.setSoTimeout(10000);
9     }
10    public void run() {
11       while(true) {
12          try {
13             System.out.println("Waiting for client on port " +
14                serverSocket.getLocalPort() + "...");
15             Socket server = serverSocket.accept();
16
17             System.out.println("Just connected to " + server.getRemoteSocketAddress());
18             DataInputStream in = new DataInputStream(server.getInputStream());
19
20             System.out.println(in.readUTF());
21             DataOutputStream out = new DataOutputStream(server.getOutputStream());
22             out.writeUTF("Thank you for connecting to " + server.getLocalSocketAddress()
23                + "\nGoodbye!");
24             server.close();
25
26          }catch(SocketTimeoutException s) {
27             System.out.println("Socket timed out!");
28             break;
29          }catch(IOException e) {
30             e.printStackTrace();
31             break;
32          }
33       }
34    }
35    public static void main(String [] args) {
36       int port = Integer.parseInt(args[0]);
37       try {
38          Thread t = new GreetingServer(port);
39          t.start();
```

```
40      }catch(IOException e) {
41         e.printStackTrace();
42      }
43   }
44}
```

§2 URL

URL£šUniform Resource Locator£©ÖÐÎÄÃûÎªÍ³Ò»×ÊÔŽ¶šÎ»·û£¬ÓÐÊ±Ò²±»Ë×³ÆÎªÍøÒ³µØÖ·¡£±íÊŸÎª»¥ÁªÍøÉÏµÄ×ÊÔŽ£¬ÈçÍøÒ³»òÕßFTPµØÖ·¡£ protocols(ÐÒé)¿ÉÒÔÊÇ HTTP, HTTPS, FTP, ºÍFile¡£port Îª¶Ë¿ÚºÅ¡£pathÎªÎÄŒþÂ·Ÿ¶Œ°ÎÄŒþÃû¡£

7.3 UML

§1.1 PlaneUml

1、Windows

```
java plantUml.jar -verbose sequenceDiagram.txt
```

2、Ubuntu

3、Mac OSx

新建 sequenceDiagram.txt 文件，将代码填写到 sequenceDiagram.txt 文件中，然后利用以下命令生成 sequenceDiagram.png

```
java -Djava.awt.headless=true -jar plantUml.jar -verbose sequenceDiagram.txt
```

- API 的第一个部分是本课程的重点。基本上是如何发送和接收独立于提供程序/协议的消息。
- 第二个部分则使用特定的协议语言，如：SMTP、POP、IMAP 和 NNTP。如果要让 JavaMail API 与服务器通信，就需要为之提供协议。由于 Sun 公司对特定协议提供程序有充分的介绍，用户可以免费获取，所以本课程没有介绍创建特定协议提供程序的内容。

JavaMail API 有 4 种常用的协议：

§1.2 UML 关系

Relationship is another most important building block of UML. It shows how the elements are associated with each other and this association describes the functionality of an application.

There are four kinds of relationships available.

1、依赖（Dependency）

简单邮件传输协议（SMTP）是用于传送电子邮件的机制。在 JavaMail API 环境中，您的基于 JavaMail 的程序将与您公司或 Internet 服务提供商（ISP）的 SMTP 服务器通信。该 SMTP 服务器将会把消息转发给用作接收消息的 SMTP 服务器，最后用户可通过 POP 或 IMAP 协议获取该消息。由

于支持身份验证，所以不需要 SMTP 服务器是一种开放的转发器，但需要确保 SMTP 服务器配置正确。JavaMail API 中没有集成用于处理诸如配置服务器以转发消息或添加/删除电子邮件帐户这一类任务的功能。

2、Association

3、Generalization

4、Realization

§2 Violet

POP 的含义是邮局协议，当前的版本为 3，也称作 POP3，该协议是在 RFC 1939 中定义的。POP 是 Internet 上的大多数人用来接收邮件的机制。它为每个用户的每个邮箱定义支持，这是它所做的全部工作，也是大多数问题的根源。在使用 POP 协议时，人们熟悉的很多功能，如查看收到了多少新邮件消息的功能，POP 根本不支持。这些功能都内置到诸如 Eudora 或 Microsoft Outlook 之类的邮件程序中，能为您记住接收的上一封邮件，以及计算有多少新邮件这类信息。因此，使用 JavaMail API 时，如果想获取这类信息，将需要由自己进行计算。

§3 IMAP

IMAP 是用于接收消息的更加高级的协议，它是在 RFC 2060 中定义的。IMAP 的含义是“Internet 消息访问协议”，当前版本是第 4 版，也称作 IMAP4。使用 IMAP 时，您的邮件服务器必须支持该协议。您不能只是简单地把程序转变为支持 IMAP，而不是支持 POP，就指望能支持 IMAP 中的一切。假定您的邮件服务器支持 IMAP，那么基于 JavaMail 的程序就可利用在服务器上拥有多个文件夹的用户，并且这些文件夹可以被多个用户共享的功能。

由于 IMAP 协议具有更高级的功能，您也许会想 IMAP 应该被每一个人使用，但事实不是这样。因为 IMAP 会加重邮件服务器的负荷，它需要服务器接收新消息，发送消息给请求的用户，并在多个文件夹中为每个用户维护这些消息。而这要集中备份，因而长期下去用户的文件夹会变得越来越大，当磁盘空间用光了时，每个人都会遭受损失。而使用 POP 协议时，已保存消息可以解除服务器的重负。

§4 MIME

MIME 的含义是“多用途的网际邮件扩充协议”。它不是一种邮件传输协议，相反，它定义传输的内容：消息的格式、附件等。许多文档都定义了 MIME 协议，包含：RFC 822、RFC 2045、RFC 2046 和 RFC 2047。作为 JavaMail API 的用户，一般不需要担心这些格式。但是，这些格式确实存在，并为您的程序所用。

NNP 和其他协议由于 JavaMail API 分开了提供程序和其他部分，所以您可以轻松地为附加协议添加支持。Sun 公司提供第 3 方提供程序清单，这些提供程序要利用 Sun 公司不支持的少见的协议。在这份清单中，您将会看到对 NNTP（网络新闻传输协议）[新闻组]、S/MIME（安全多用途的网际邮件扩充协议）及其他协议的提供支持的第 3 方程序。

使用 Java 发送 E-Mail 十分简单，但是要在本地机器上安装 Java Mail API 和 Java Activation Framework(JAF)

从 Java 网站下载最新版本的 JavaMail，打开网页右侧有个 Downloads 链接，点击它下载。从 Java 网站下载最新版本的 JAF（版本 1.1.1）。使用本站提供的下载链接：JavaMail mail.jar 1.4.5 JAF（版

本 1.1.1）activation.jar 下载并解压缩这些文件，在新创建的顶层目录中，您会发现这两个应用程序的一些 jar 文件。您需要把 mail.jar 和 activation.jar 文件添加到您的 CLASSPATH 中。如果你使用第三方邮件服务器如 QQ 的 SMTP 服务器，可查看文章底部用户认证完整的实例。

在 WIndow 平台下编译命令为

```
D:\Java_program\code5\Mail>javac -classpath
    .;\%CLASSPATH\%;./activation.jar;./mail.jar -encoding utf-8 App2_8.java
D:\Java_program\code5\Mail>java -classpath
    .;\%CLASSPATH\%;./activation.jar;./mail.jar App2_8
```

在 Mac osX 平台下编译命令为

```
javac -classpath .:/System/Library/Java/:./activation.jar:./mail.jar -encoding
    utf8 SendMail.java
```

7.4 课后作业

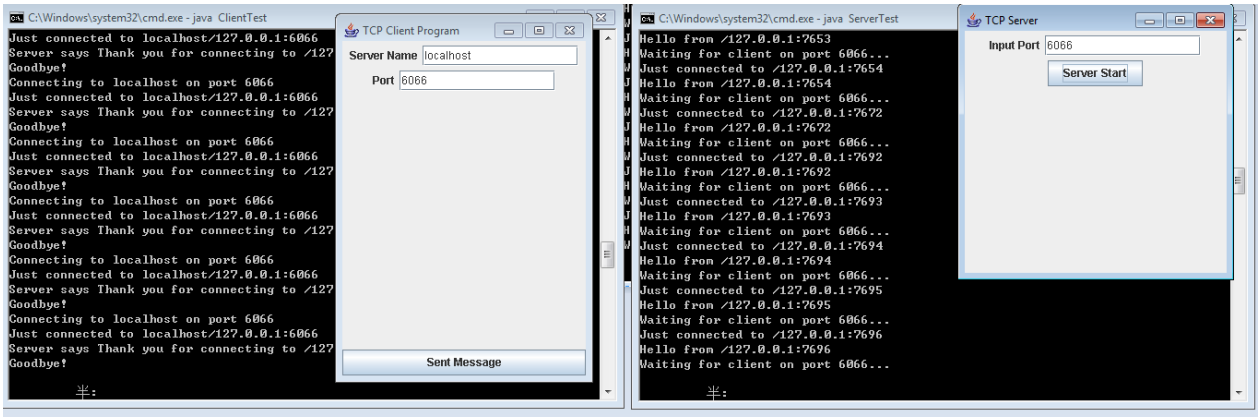


图 1: Client 和 Server 通信

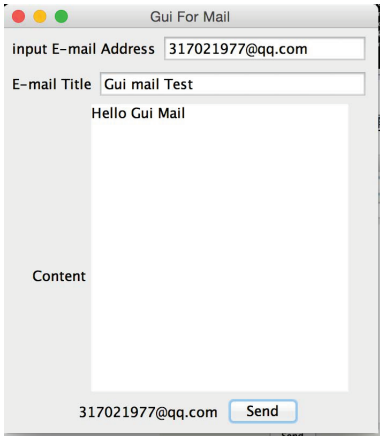


图 2: Gui For Mail

8 Network

1、教学内容

(1) 理解 TCP/IP、UDP 等网络传输协议；
(2) 理解 Java Net 的 Socket 类和 URL 处理；
(3) 理解 Java Mail 收发邮件的机制。

2、教学难点

理解 Java 网络编程，通过 Socket、URL 类处理。

3、教学重点

Java 的 Socket 类、URL 类以及 Mail 类的使用。

教学内容

8.1 Java Network

网络编程是指编写运行在多个设备（计算机）的程序，这些设备都通过网络连接起来。java.net 包中 J2SE 的 API 包含有类和接口，它们提供低层次的通信细节。你可以直接使用这些类和接口，来专注于解决问题，而不用关注通信细节。java.net 包中提供了两种常见的网络协议的支持：

- 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。

8.2 Socket URL

- Socket 编程这是使用最广泛的网络概念。
- URL 处理

§1 Socket

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。

```
1  import java.net.*;
2  import java.io.*;
3
4  public class GreetingClient {
5     public static void main(String [] args) {
6        String serverName = args[0];
7        int port = Integer.parseInt(args[1]);
8        try {
9           System.out.println("Connecting to " + serverName + " on port " + port);
10          Socket client = new Socket(serverName, port);
11          System.out.println("Just connected to " + client.getRemoteSocketAddress());
12          OutputStream outToServer = client.getOutputStream();
13          DataOutputStream out = new DataOutputStream(outToServer);
14          out.writeUTF("Hello from " + client.getLocalSocketAddress());
15          InputStream inFromServer = client.getInputStream();
16          DataInputStream in = new DataInputStream(inFromServer);
17          System.out.println("Server says " + in.readUTF());
```

```
18          client.close();
19       }catch(IOException e) {
20          e.printStackTrace();
21       }
22    }
23 }
```

```
1  import java.net.*;
2  import java.io.*;
3
4  public class GreetingServer extends Thread {
5     private ServerSocket serverSocket;
6     public GreetingServer(int port) throws IOException {
7        serverSocket = new ServerSocket(port);
8        serverSocket.setSoTimeout(10000);
9     }
10    public void run() {
11       while(true) {
12          try {
13             System.out.println("Waiting for client on port " +
14                serverSocket.getLocalPort() + "...");
15             Socket server = serverSocket.accept();
16
17             System.out.println("Just connected to " + server.getRemoteSocketAddress());
18             DataInputStream in = new DataInputStream(server.getInputStream());
19
20             System.out.println(in.readUTF());
21             DataOutputStream out = new DataOutputStream(server.getOutputStream());
22             out.writeUTF("Thank you for connecting to " + server.getLocalSocketAddress()
23                + "\nGoodbye!");
24             server.close();
25
26          }catch(SocketTimeoutException s) {
27             System.out.println("Socket timed out!");
28             break;
29          }catch(IOException e) {
30             e.printStackTrace();
31             break;
32          }
33       }
34    }
35    public static void main(String [] args) {
36       int port = Integer.parseInt(args[0]);
37       try {
38          Thread t = new GreetingServer(port);
39          t.start();
```

```
40      }catch(IOException e) {
41        e.printStackTrace();
42      }
43    }
44  }
```

§2 URL

URL（Uniform Resource Locator）中文名为统一资源定位符，有时也被俗称为网页地址。表示为互联网上的资源，如网页或者 FTP 地址。protocols(协议) 可以是 HTTP, HTTPS, FTP, 和 File。port 为端口号。path 为文件路径及文件名。

8.3 Mail

JavaMail API 是一种可选的、能用于读取、编写和发送电子消息的包（标准扩展）。您可使用这种包创建邮件用户代理（Mail User Agent ， MUA）类型的程序，它类似于 Eudora、Pine 及 Microsoft Outlook 这些邮件程序。其主要目的不是像发送邮件或其他邮件传输代理（Mail Transfer Agent，MTA）类型的程序那样用于传输、发送和转发消息。换句话说，用户可以与 MUA 类型的程序交互，以阅读和撰写电子邮件。MUA 依靠 MTA 处理实际的发送任务。

JavaMail API 的设计是，为收发信息提供与协议无关的访问。方式是把该 API 划分成两个部分：

- API 的第一个部分是本课程的重点。基本上是如何发送和接收独立于提供程序/协议的消息。
- 第二个部分则使用特定的协议语言，如：SMTP、POP、IMAP 和 NNTP。如果要让 JavaMail API 与服务器通信，就需要为之提供协议。由于 Sun 公司对特定协议提供程序有充分的介绍，用户可以免费获取，所以本课程没有介绍创建特定协议提供程序的内容。

JavaMail API 有 4 种常用的协议：

§1 SMTP

简单邮件传输协议（SMTP）是用于传送电子邮件的机制。在 JavaMail API 环境中，您的基于 JavaMail 的程序将与您公司或 Internet 服务提供商（ISP）的 SMTP 服务器通信。该 SMTP 服务器将会把消息转发给用作接收消息的 SMTP 服务器，最后用户可通过 POP 或 IMAP 协议获取该消息。由于支持身份验证，所以不需要 SMTP 服务器是一种开放的转发器，但需要确保 SMTP 服务器配置正确。JavaMail API 中没有集成用于处理诸如配置服务器以转发消息或添加/删除电子邮件帐户这一类任务的功能。

§2 POP

POP 的含义是邮局协议，当前的版本为 3，也称作 POP3，该协议是在 RFC 1939 中定义的。POP 是 Internet 上的大多数人用来接收邮件的机制。它为每个用户的每个邮箱定义支持，这是它所做的全部工作，也是大多数问题的根源。在使用 POP 协议时，人们熟悉的很多功能，如查看收到了多少新邮件消息的功能，POP 根本不支持。这些功能都内置到诸如 Eudora 或 Microsoft Outlook 之类的邮件程序中，能为您记住接收的上一封邮件，以及计算有多少新邮件这类信息。因此，使用 JavaMail API 时，如果想获取这类信息，将需要由自己进行计算。

§3 IMAP

IMAP 是用于接收消息的更加高级的协议，它是在 RFC 2060 中定义的。IMAP 的含义是“Internet 消息访问协议”，当前版本是第 4 版，也称作 IMAP4。使用 IMAP 时，您的邮件服务器必须支持该协议。您不能只是简单地把程序转变为支持 IMAP，而不是支持 POP，就指望能支持 IMAP 中的一切。假定您的邮件服务器支持 IMAP，那么基于 JavaMail 的程序就可利用在服务器上拥有多个文件夹的用户，并且这些文件夹可以被多个用户共享的功能。

由于 IMAP 协议具有更高级的功能，您也许会想 IMAP 应该被每一个人使用，但事实不是这样。因为 IMAP 会加重邮件服务器的负荷，它需要服务器接收新消息，发送消息给请求的用户，并在多个文件夹中为每个用户维护这些消息。而这要集中备份，因而长期下去用户的文件夹会变得越来越大，当磁盘空间用光了时，每个人都会遭受损失。而使用 POP 协议时，已保存消息可以解除服务器的重负。

§4 MIME

MIME 的含义是“多用途的网际邮件扩充协议”。它不是一种邮件传输协议，相反，它定义传输的内容：消息的格式、附件等。许多文档都定义了 MIME 协议，包含：RFC 822、RFC 2045、RFC 2046 和 RFC 2047。作为 JavaMail API 的用户，一般不需要担心这些格式。但是，这些格式确实存在，并为您的程序所用。

NNP 和其他协议由于 JavaMail API 分开了提供程序和其他部分，所以您可以轻松地为附加协议添加支持。Sun 公司提供第 3 方提供程序清单，这些提供程序要利用 Sun 公司不支持的少见的协议。在这份清单中，您将会看到对 NNTP（网络新闻传输协议）[新闻组]、S/MIME（安全多用途的网际邮件扩充协议）及其他协议的提供支持的第 3 方程序。

使用 Java 发送 E-Mail 十分简单，但是要在本地机器上安装 Java Mail API 和 Java Activation Framework(JAF)

从 Java 网站下载最新版本的 JavaMail，打开网页右侧有个 Downloads 链接，点击它下载。从 Java 网站下载最新版本的 JAF（版本 1.1.1）。使用本站提供的下载链接：JavaMail mail.jar 1.4.5 JAF（版本 1.1.1）activation.jar 下载并解压缩这些文件，在新创建的顶层目录中，您会发现这两个应用程序的一些 jar 文件。您需要把 mail.jar 和 activation.jar 文件添加到您的 CLASSPATH 中。如果你使用第三方邮件服务器如 QQ 的 SMTP 服务器，可查看文章底部用户认证完整的实例。

在 WIndow 平台下编译命令为

```
D:\Java_program\code5\Mail>javac -classpath
    .;\%CLASSPATH\%;./activation.jar;./mail.jar -encoding utf-8 App2_8.java
D:\Java_program\code5\Mail>java -classpath
    .;\%CLASSPATH\%;./activation.jar;./mail.jar App2_8
```

在 Mac osX 平台下编译命令为

```
javac -classpath .:/System/Library/Java/:./activation.jar:./mail.jar -encoding
    utf8 SendMail.java
```

8.4 课后作业

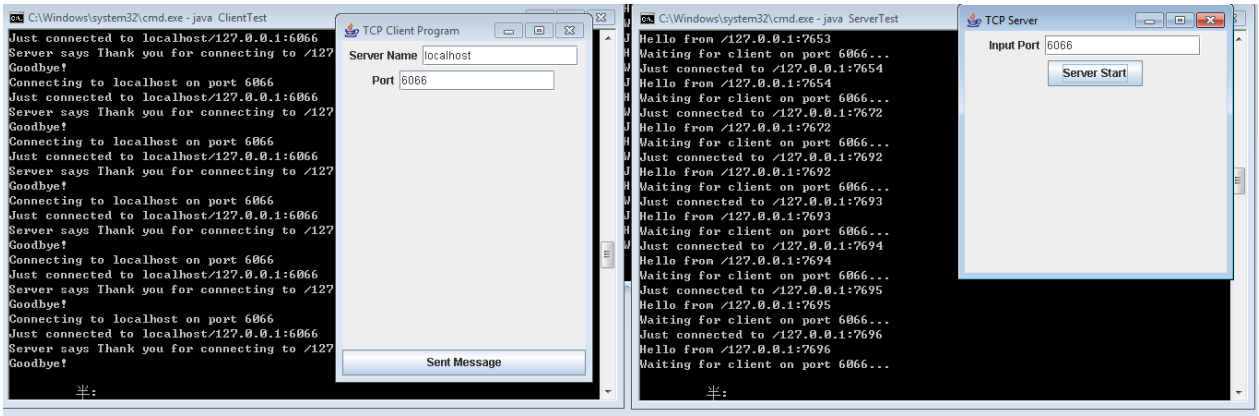


图 1: Client 和 Server 通信

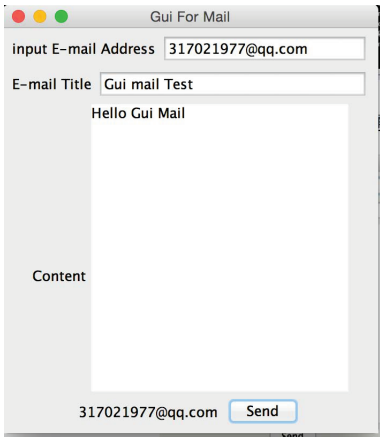


图 2: Gui For Mail

致谢 NBUT XeLatex 模板

参考文献

[1] Java 面向对象程序设计，董小园著，清华大学出版社。

[2] L. Baldassarre, L. Rosasco, A. Barla, and A. Verri. Vector ?eld learning via spectral ?ltering. In ECML, 2010.