**public** **class** PCB {

**int** id;// 进程号

**int** time;// 运行时间

**int** priority; // 优先级。 数字越小，优先级越高

**int** cometime; // 到达时间

**int** length; // 进程大小

**public** PCB(**int** id, **int** cometime, **int** time, **int** priority) {

**super**();

**this**.id = id;

**this**.time = time;

**this**.priority = priority;

**this**.cometime = cometime;

}

**public** PCB() {

**super**();

}

**public** PCB(**int** id, **int** time, **int** priority, **int** cometime, **int** length) {

**super**();

**this**.id = id;

**this**.time = time;

**this**.priority = priority;

**this**.cometime = cometime;

**this**.length = length;

}

@Override

**public** String toString() {

**return** id + "\_" + time + "\_" + priority + "\_" + cometime + "\_" + length;

}

}

**public** **class** Free {

**int** id; // 分区号

**int** start; // 起始地址

**int** capacity; // 分区大小

**int** status = 0; // 0 表示空闲

Free(**int** id, **int** start, **int** capacity) {

**this**.id = id;

**this**.start = start;

**this**.capacity = capacity;

}

@Override

**public** String toString() {

**if** (**this**.status == 0)

**return** id + "\_" + capacity + "\_" + start + "\_" + "空闲";

**return** id + "\_" + capacity + "\_" + start + "\_" + "占用";

}

}

**package** OS;

**import** java.awt.EventQueue;

**import** javax.swing.JFrame;

**import** javax.swing.JPanel;

**import** javax.swing.border.EmptyBorder;

**import** java.util.List;

**import** javax.swing.JLabel;

**import** java.awt.Font;

**import** javax.swing.JTextField;

**import** javax.swing.JButton;

**import** java.awt.Color;

**import** java.awt.event.MouseAdapter;

**import** java.awt.event.MouseEvent;

**import** java.util.ArrayList;

**import** java.util.Collections;

**import** java.util.Comparator;

**import** java.util.Iterator;

**import** java.util.LinkedList;

**import** javax.swing.JTextArea;

**import** javax.swing.JScrollPane;

**import** javax.swing.ScrollPaneConstants;

**public** **class** Demo **extends** JFrame {

/\* public void listIntegerSort() {

// 插入十个100以内的不重复随机整数

List<Integer> integerList = new ArrayList();

integerList.add(1);

System.out.println("-------------排序前--------------");

for (Integer integer : integerList) {

System.out.println("元素：" + integer);

}

Collections.sort(integerList);

System.out.println("----------------排序后-------------------");

for (Integer integer : integerList) {

System.out.println("元素：" + integer);

}

}\*/

**private** JPanel contentPane;

**private** JTextField textField;

**private** JTextField textField\_1;

**private** JTextField textField\_4;

**private** JTextField textField\_5;

**private** JTextField textField\_6;

**private** JTextField textField\_8;

**private** JTextField textField\_9;

// 空闲分区表

**static** LinkedList<Free> *LinkedFree* = **new** LinkedList<Free>();

// 后置队列

**static** ArrayList<PCB> *houzhi* = **new** ArrayList<PCB>();

// 就绪队列

**static** ArrayList<PCB> *jiuxu* = **new** ArrayList<PCB>(2);

// 挂起队列

**static** ArrayList<PCB> *guaqi* = **new** ArrayList<PCB>();

// 设置道数

**static** **int** *daoshu* = 3;// 总道数为3

**static** **int** *i* = 0; // 当前内存进程数

**static** **boolean** *cpu1* = **false**; // cpu状态

**static** PCB *cpuPcb1* = **new** PCB(); // cpu上运行的进程

**static** **boolean** *cpu2* = **false**; // cpu状态

**static** PCB *cpuPcb2* = **new** PCB(); // cpu上运行的进程

**static** **int** *t* = 1; // 总时间

**static** **int** *t1*=0;

/\*\* 将pcb从队列1转移到队列2 \*/

**public** **static** **void** Move(PCB pcb, ArrayList<PCB> al1, ArrayList<PCB> al2) {

**int** a = 0; // 记录pcb在队列1的位置

**int** b = al1.size();

**for** (**int** i = 0; i < al1.size(); i++) {

**if** (al1.get(i).id == pcb.id) {

a = i;

**break**;

}

}

al2.add(pcb);// al2队尾添加该进程

**for** (**int** j = a + 1; j < b; j++) {

al1.add(j - 1, al1.get(j));

}

**while** (al1.size() != b - 1) {

al1.remove(al1.size() - 1);

}

}

**public** **static** **void** Move1(PCB pcb, ArrayList<PCB> al2) {

al2.add(pcb);

}

/\*\* 初始化内存状态 \*/

**public** **static** **void** initMemory() {

**int** size = 950;

Free temp = **new** Free(0, 50, size);

*LinkedFree*.add(temp); // 空闲分区表添加元素

}

/\*\* 释放内存 \*/

**public** **static** **void** release(PCB pcb) {

Collections.*sort*(*LinkedFree*, **new** SortByStart());

**int** id = pcb.id;

Iterator<Free> iterator = *LinkedFree*.iterator();// 迭代器

**int** index = 0; // 要移除的分区号 下标

**int** index2 = -1; // 要合并的分区号

**while** (iterator.hasNext()) {

Free p = iterator.next();

index = *LinkedFree*.indexOf(p);

**if** (p.id == id) { // 找到此分区

Free free1 = **null**, free3 = **null**;

**if** (index - 1 >= 0) { // 前面有分区

**if** (*LinkedFree*.get(index - 1).status == 0) { // 前 是空闲分区

free1 = *LinkedFree*.get(index - 1);

}

}

**if** (index + 1 < *LinkedFree*.size()) { // 后面有分区

**if** (*LinkedFree*.get(index + 1).status == 0) { // 后 是空闲分区

free3 = *LinkedFree*.get(index + 1);

}

}

**if** (free1 != **null** && free3 != **null**) { // 合并前后分区

free1.capacity += p.capacity + free3.capacity;

index2 = index;

iterator.remove();

} **else** **if** (free1 != **null** && free3 == **null**) { // 合并前面分区

free1.capacity += p.capacity;

iterator.remove();

} **else** **if** (free1 == **null** && free3 != **null**) { // 合并后面分区

free3.capacity += p.capacity;

free3.start -= p.capacity;

iterator.remove();

} **else** { // 不需要合并

p.status = 0; // 忙碌 变为空闲

}

}

}

**if** (index2 != -1) {

*LinkedFree*.remove(index2); // 删除合并的内存块信息

}

}

/\*\* First Fit 算法 \*/

**public** **static** **void** allocation(PCB pcb) {

**int** applySize = pcb.length;

Free temp = **null**; // 要分配的分区

Collections.*sort*(*LinkedFree*, **new** SortByStart()); // 根据起始地址排序

**for** (**int** i = 0; i < *LinkedFree*.size(); i++) {

Free point = *LinkedFree*.get(i);

**if** (point.capacity >= applySize && point.status == 0) { // 内存充裕

temp = **new** Free(pcb.id, point.start, applySize);

temp.status = 1;

point.capacity -= applySize;

point.start += applySize;

*LinkedFree*.add(temp);

**if** (point.capacity == 0) { // 空间全部分配

*LinkedFree*.remove(point);

}

**break**;

}

}

Collections.*sort*(*LinkedFree*, **new** SortByStart()); // 根据起始地址排序

}

/\*\*

\* Launch the application.

\*/

**public** **static** **void** main(String[] args) {

EventQueue.*invokeLater*(**new** Runnable() {

**public** **void** run() {

**try**

{Demo frame = **new** Demo();

//frame. listIntegerSort();

*initMemory*();

frame.setVisible(**true**);

} **catch** (Exception e) {

e.printStackTrace();

}

}

});

}

/\*\*

\* Create the frame.

\*/

**public** Demo() {

setDefaultCloseOperation(JFrame.***EXIT\_ON\_CLOSE***);

setBounds(100, 100, 800, 593);

contentPane = **new** JPanel();

contentPane.setBorder(**new** EmptyBorder(5, 5, 5, 5));

setContentPane(contentPane);

contentPane.setLayout(**null**);

JLabel label = **new** JLabel("后备队列：");

label.setFont(**new** Font("宋体", Font.***PLAIN***, 15));

label.setBounds(10, 10, 80, 25);

contentPane.add(label);

JLabel lblNewLabel = **new** JLabel("进程名 运行时间 优先级 到达时间 大小");

lblNewLabel.setFont(**new** Font("宋体", Font.***PLAIN***, 12));

lblNewLabel.setBounds(10, 32, 280, 25);

contentPane.add(lblNewLabel);

JLabel label\_1 = **new** JLabel("就绪队列：");

label\_1.setFont(**new** Font("宋体", Font.***PLAIN***, 15));

label\_1.setBounds(10, 213, 80, 25);

contentPane.add(label\_1);

// 后备队列

JTextArea textArea = **new** JTextArea();

textArea.setFont(**new** Font("Monospaced", Font.***PLAIN***, 15));

textArea.setBounds(10, 67, 250, 135);

contentPane.add(textArea);

// 空闲分区表

JTextArea textArea\_1 = **new** JTextArea();

textArea\_1.setBounds(555, 53, 212, 150);

contentPane.add(textArea\_1);

// 就绪队列

JTextArea textArea\_2 = **new** JTextArea();

textArea\_2.setBounds(10, 237, 250, 100);

contentPane.add(textArea\_2);

// 挂起队列

JTextArea textArea\_3 = **new** JTextArea();

textArea\_3.setBounds(295, 237, 250, 100);

contentPane.add(textArea\_3);

// 状态栏（带进度条）

JTextArea textArea\_4 = **new** JTextArea();

JScrollPane scr = **new** JScrollPane(textArea\_4);

scr.setBackground(Color.***ORANGE***);

scr.setBounds(10, 420, 528, 128);

contentPane.add(scr);

JButton button = **new** JButton("挂起");

button.addMouseListener(**new** MouseAdapter() {

@Override

**public** **void** mouseClicked(MouseEvent e) {

PCB pcb = **new** PCB();

pcb.id = Integer.*parseInt*(textField.getText());

textField.setText("");

**for** (**int** i = 0; i < *jiuxu*.size(); i++) {

**if** (*jiuxu*.get(i).id == pcb.id)

pcb = *jiuxu*.get(i);

}

*release*(pcb);

*Move*(pcb, *jiuxu*, *guaqi*);

textArea\_4.append("进程" + pcb.id + "已被挂起。 t=" + *t* + "\n");

textArea\_3.append(pcb.toString() + "\n");

**if** (*i* < *daoshu* && *houzhi*.size() != 0) {

Collections.*sort*(*houzhi*, **new** SortByPriority()); // 根据优先级排序

*Move*(*houzhi*.get(0), *houzhi*, *jiuxu*);

*i*++;

}

textArea\_2.setText("");

textArea\_1.setText("");

textArea.setText("");

**for** (PCB pcb1 : *jiuxu*)// 打印就绪队列

textArea\_2.append(pcb1.toString() + "\n");

**for** (PCB pcb1 : *houzhi*)// 打印后置队列

textArea\_2.append(pcb1.toString() + "\n");

**for** (Free free : *LinkedFree*)// 打印空闲分区表

textArea\_1.append(free.toString() + "\n");

}

});

button.setBounds(170, 357, 90, 25);

contentPane.add(button);

JLabel lblNewLabel\_1 = **new** JLabel("状态：");

lblNewLabel\_1.setFont(**new** Font("宋体", Font.***PLAIN***, 15));

lblNewLabel\_1.setBounds(10, 392, 54, 18);

contentPane.add(lblNewLabel\_1);

JLabel label\_2 = **new** JLabel("挂起队列：");

label\_2.setFont(**new** Font("宋体", Font.***PLAIN***, 15));

label\_2.setBounds(295, 213, 80, 25);

contentPane.add(label\_2);

JButton btnNewButton = **new** JButton("解挂");

btnNewButton.addMouseListener(**new** MouseAdapter() {

@Override

**public** **void** mouseClicked(MouseEvent e) {

PCB pcb = **new** PCB();

pcb.id = Integer.*parseInt*(textField\_1.getText());

textField\_1.setText("");

**for** (**int** i = 0; i < *guaqi*.size(); i++) {

**if** (*guaqi*.get(i).id == pcb.id)

pcb = *guaqi*.get(i);

}

textArea\_3.setText("");

*allocation*(pcb);

*Move*(pcb, *guaqi*, *jiuxu*);

textArea\_4.append("进程" + pcb.id + "进入就绪队列。 t1=" + *t* + "\n");

textArea\_2.setText("");

textArea\_1.setText("");

textArea.setText("");

**for** (PCB pcb1 : *jiuxu*)// 打印就绪队列

textArea\_2.append(pcb1.toString() + "\n");

**for** (Free free : *LinkedFree*)// 打印空闲分区表

textArea\_1.append(free.toString() + "\n");

**for** (PCB pcb1 : *guaqi*)// 打印挂起队列

textArea\_3.append(pcb1.toString() + "\n");

**for** (PCB pcb1 : *houzhi*)// 打印后置队列

textArea.append(pcb1.toString() + "\n");

}

});

btnNewButton.setBounds(455, 357, 90, 25);

contentPane.add(btnNewButton);

JLabel lblPcb = **new** JLabel("PCB信息：");

lblPcb.setFont(**new** Font("宋体", Font.***PLAIN***, 15));

lblPcb.setBounds(570, 213, 80, 25);

contentPane.add(lblPcb);

JLabel label\_3 = **new** JLabel("进程号：");

label\_3.setBounds(570, 248, 80, 15);

contentPane.add(label\_3);

JLabel label\_4 = **new** JLabel("运行时间：");

label\_4.setBounds(570, 273, 80, 15);

contentPane.add(label\_4);

JLabel label\_5 = **new** JLabel("优先级：");

label\_5.setBounds(570, 297, 80, 15);

contentPane.add(label\_5);

JLabel label\_6 = **new** JLabel("到达时间：" + *t*);

label\_6.setBounds(570, 322, 75, 15);

contentPane.add(label\_6);

JLabel label\_7 = **new** JLabel("大小：");

label\_7.setBounds(570, 347, 80, 15);

contentPane.add(label\_7);

// 对应Pcb信息

textField\_4 = **new** JTextField();

textField\_4.setBounds(642, 245, 100, 18);

contentPane.add(textField\_4);

textField\_4.setColumns(10);

textField\_5 = **new** JTextField();

textField\_5.setBounds(642, 271, 100, 18);

contentPane.add(textField\_5);

textField\_5.setColumns(10);

textField\_6 = **new** JTextField();

textField\_6.setBounds(642, 294, 100, 18);

contentPane.add(textField\_6);

textField\_6.setColumns(10);

textField\_8 = **new** JTextField();

textField\_8.setBounds(642, 344, 100, 18);

contentPane.add(textField\_8);

textField\_8.setColumns(10);

JTextArea txtrkosk = **new** JTextArea();

txtrkosk.setFont(**new** Font("宋体", Font.***PLAIN***, 15));

txtrkosk.setBackground(Color.***WHITE***);

txtrkosk.setText("(1)内存大小为1000K，OS占前50K\r\n(2)道数取为"+*daoshu*+"\n(3)总时间设为t");

txtrkosk.setBounds(270, 11, 263, 58);

contentPane.add(txtrkosk);

JButton button\_1 = **new** JButton("添加");

button\_1.addMouseListener(**new** MouseAdapter() {

@Override

**public** **void** mouseClicked(MouseEvent e) {

**int** a = Integer.*parseInt*(textField\_4.getText());

**int** b = Integer.*parseInt*(textField\_5.getText());

**int** c = Integer.*parseInt*(textField\_6.getText());

**int** d = Integer.*parseInt*(textField\_8.getText());

PCB pcb = **new** PCB(a, b, c, *t1*, d);

*houzhi*.add(pcb);

**if** (*i* < *daoshu*)

// 内存中进程少于规定道数，选择优先级最高的进入就绪队列，分配内存

{

Collections.*sort*(*houzhi*, **new** SortByPriority()); // 后置队列按优先级排序

PCB pcb1 = *houzhi*.get(0);

*houzhi*.remove(pcb1);

*jiuxu*.add(pcb1);

textArea\_2.append(pcb1.toString() + "\n");

textArea\_4.append("进程" + pcb1.id + "进入就绪队列。 t1=" + *t1* + "\n");

*i*++;

*allocation*(pcb1); // 分配内存

textArea.setText("");

textArea\_1.setText("");

**for** (Free free : *LinkedFree*)

textArea\_1.append(free.toString() + "\n");

**for** (PCB pcb2 : *houzhi*) // 打印后置队列

textArea.append(pcb2.toString() + "\n");

} **else**

// 进入后置队列

{

textArea\_4.append("进程" + pcb.id + "进入后置队列。 t1=" + *t1* + "\n");

**for** (PCB pcb2 : *houzhi*) // 打印后置队列

textArea.append(pcb2.toString() + "\n");

}

}

});

button\_1.setBounds(570, 382, 90, 35);

contentPane.add(button\_1);

JButton button\_2 = **new** JButton("重置");

button\_2.addMouseListener(**new** MouseAdapter() {

@Override

**public** **void** mouseClicked(MouseEvent e) {

textField\_4.setText("");

textField\_5.setText("");

textField\_6.setText("");

textField\_8.setText("");

}

});

button\_2.setBounds(670, 382, 90, 35);

contentPane.add(button\_2);

JLabel lblCpu = **new** JLabel("CPU状态：");

lblCpu.setFont(**new** Font("宋体", Font.***PLAIN***, 15));

lblCpu.setBounds(295, 94, 80, 18);

contentPane.add(lblCpu);

textField\_9 = **new** JTextField();

textField\_9.setFont(**new** Font("宋体", Font.***PLAIN***, 15));

textField\_9.setBounds(295, 122, 243, 76);

contentPane.add(textField\_9);

textField\_9.setColumns(10);

JLabel lblosk = **new** JLabel("空闲分区表：");

lblosk.setFont(**new** Font("宋体", Font.***PLAIN***, 15));

lblosk.setBounds(548, 15, 226, 20);

contentPane.add(lblosk);

JLabel lblNewLabel\_3 = **new** JLabel("t=" + *t*);

lblNewLabel\_3.setFont(**new** Font("宋体", Font.***PLAIN***, 20));

lblNewLabel\_3.setBounds(570, 518, 100, 26);

contentPane.add(lblNewLabel\_3);

JButton btnNewButton\_1 = **new** JButton("Step");

btnNewButton\_1.addMouseListener(**new** MouseAdapter() {

**public** **void** mouseClicked(MouseEvent e) {

**try** {

lblNewLabel\_3.setText("t1=" + *t1*);

label\_6.setText("到达时间:" + *t*);

**if** (*i* < *daoshu* && *houzhi*.size() != 0) {

Collections.*sort*(*houzhi*, **new** SortByPriority()); // 后置队列按优先级排序

*Move*(*houzhi*.get(0), *houzhi*, *jiuxu*);

*i*++;

}

**if**(!*jiuxu*.isEmpty())

{Collections.*sort*(*jiuxu*, **new** SortByPriority());} // 就绪队列按优先级排序

**if** (!*jiuxu*.isEmpty()&&*cpu1* == **false**) {// cpu1上没有进程

PCB pcb1 = *jiuxu*.get(0);

*jiuxu*.remove(0);

*cpuPcb1* = pcb1;

**if** (*cpuPcb1*.time == 0) {

textArea\_4.append("cpu1上的进程" + *cpuPcb1*.id + "已完成。 t=" + *t* + "\n");

textField\_9.setText("");

*release*(*cpuPcb1*);

*i*--;

*cpu1* = **false**;

}

*cpuPcb1*.time -= 1;

*cpuPcb1*.priority+=1;

*cpu1* = **true**;

textArea\_4.append("进程" + *cpuPcb1*.id + "进入CPU1执行。 t=" + *t* + "\n");

textField\_9.setText("进程" + *cpuPcb1*.id + "正在执行");

} **else** **if** ( !*jiuxu*.isEmpty()&&*cpuPcb1*.priority > *jiuxu*.get(0).priority ) // cpu1上有进程，进行优先级比较

{

*Move1*(*cpuPcb1*, *guaqi*); //将进程置入挂起队列

textArea\_4.append("进程" + *cpuPcb1*.id + "被进程" + *jiuxu*.get(0).id + "抢占，进程" + *jiuxu*.get(0).id

+ "进入CPU1执行。 t=" + *t* + "\n");

*release*(*cpuPcb1*);

*cpuPcb1* = *jiuxu*.get(0);

*jiuxu*.remove(0);

textField\_9.setText("进程" + *cpuPcb1*.id + "正在执行");

*cpuPcb1*.time -= 1;

*cpuPcb1*.priority+=1;

} **else** **if**(!*jiuxu*.isEmpty()&&*cpuPcb1*.priority <= *jiuxu*.get(0).priority||*cpu1*==**true**) {

*cpuPcb1*.time -= 1;

*cpuPcb1*.priority+=1;

textArea\_4.append("进程" + *cpuPcb1*.id + "进入CPU1执行。 t=" + *t* + "\n");

textField\_9.setText("进程" + *cpuPcb1*.id + "正在执行");

}

**if**(*cpuPcb1*==**null**) {

textArea\_4.append("cpu1为空\n");

}

**else** **if** (*cpuPcb1*.time == 0) {

textArea\_4.append("cpu1上的进程" + *cpuPcb1*.id + "已完成。 t=" + *t* + "\n");

textField\_9.setText("");

*release*(*cpuPcb1*);

*cpuPcb1*=**null**;

*i*--;

*cpu1* = **false**;

}

//处理机2

**if** (!*jiuxu*.isEmpty()&&*cpu2* == **false**) {// cpu2上没有进程

PCB pcb1 = *jiuxu*.get(0);

*jiuxu*.remove(0);*cpuPcb2* = pcb1;

**if** (*cpuPcb2*.time == 0) {

textArea\_4.append("cpu2上的进程" + *cpuPcb2*.id + "已完成。 t=" + *t* + "\n");

textField\_9.setText("");

*release*(*cpuPcb2*);

*i*--;

*cpu2* = **false**;

}

*cpuPcb2*.time -= 1;

*cpuPcb2*.priority+=1;

*cpu2* = **true**;

textArea\_4.append("进程" + *cpuPcb2*.id + "进入CPU2执行。 t=" + *t* + "\n");

textField\_9.setText("进程" + *cpuPcb2*.id + "正在执行");

} **else** **if** (!*jiuxu*.isEmpty()&&*cpuPcb2*.priority > *jiuxu*.get(0).priority ) // cpu2上有进程，进行优先级比较

{

*Move1*(*cpuPcb2*, *guaqi*); //将进程由就绪队列置入挂起队列

textArea\_4.append("进程" + *cpuPcb2*.id + "被进程" + *jiuxu*.get(0).id + "抢占，进程" + *jiuxu*.get(0).id

+ "进入CPU2执行。 t=" + *t* + "\n");

*release*(*cpuPcb2*);

*cpuPcb2* = *jiuxu*.get(0);

*jiuxu*.remove(0);

textField\_9.setText("进程" + *cpuPcb2*.id + "正在执行");

*cpuPcb2*.time -= 1;

*cpuPcb2*.priority+=1;

} **else** **if**(!*jiuxu*.isEmpty()&&*cpuPcb2*.priority <= *jiuxu*.get(0).priority||*cpu2*==**true**){

*cpuPcb2*.time -= 1;

*cpuPcb2*.priority+=1;

textArea\_4.append("进程" + *cpuPcb2*.id + "进入CPU2执行。 t=" + *t* + "\n");

textField\_9.setText("进程" + *cpuPcb2*.id + "正在执行");

}

**if**(*cpuPcb2*==**null**) {

textArea\_4.append("cpu2为空\n");

}

**else** **if** (*cpuPcb2*.time == 0) {

textArea\_4.append("CPU2上的进程" + *cpuPcb2*.id + "已完成。 t=" + *t* + "\n");

textField\_9.setText("");

*release*(*cpuPcb2*);

*cpuPcb2*=**null**;

*i*--;

*cpu2* = **false**;

}

*t*++;

System.***out***.println(*jiuxu*.toString());

textArea\_3.setText("");

textArea\_2.setText("");

textArea\_1.setText("");

textArea.setText("");

**for** (PCB pcb : *jiuxu*)// 打印就绪队列

textArea\_2.append(pcb.toString() + "\n");

**for** (Free free : *LinkedFree*)// 打印空闲分区表

textArea\_1.append(free.toString() + "\n");

**for** (PCB pcb : *guaqi*)// 打印挂起队列

textArea\_3.append(pcb.toString() + "\n");

**for** (PCB pcb : *houzhi*)// 打印后置队列

textArea.append(pcb.toString() + "\n");

}

**catch** (Exception e2) {

lblNewLabel\_3.setText("总用时：" + (--*t*));

textField\_9.setText("");

}

}

});

btnNewButton\_1.setFont(**new** Font("宋体", Font.***PLAIN***, 15));

btnNewButton\_1.setBounds(570, 448, 190, 25);

contentPane.add(btnNewButton\_1);

JButton btnNewButton\_2 = **new** JButton("重置系统");

btnNewButton\_2.addMouseListener(**new** MouseAdapter() {

@Override

**public** **void** mouseClicked(MouseEvent e) {

*houzhi*.clear();

*jiuxu*.clear();

*guaqi*.clear();

*LinkedFree*.clear();

*i* = 0;

*cpu1* = **false**;

*cpu2*=**false**;

*t* = 0;

textArea\_4.setText("");

textArea\_3.setText("");

textArea\_2.setText("");

textArea\_1.setText("");

textArea.setText("");

textField\_4.setText("");

textField\_5.setText("");

textField\_6.setText("");

textField\_8.setText("");

textField\_9.setText("");

lblNewLabel\_3.setText("t=" + *t*);

*initMemory*();

}

});

btnNewButton\_2.setFont(**new** Font("宋体", Font.***PLAIN***, 15));

btnNewButton\_2.setBounds(570, 483, 190, 25);

contentPane.add(btnNewButton\_2);

JScrollPane scrollPane = **new** JScrollPane();

scrollPane.setBounds(434, 452, 2, 2);

contentPane.add(scrollPane);

JScrollPane scrollPane\_1 = **new** JScrollPane();

scrollPane\_1.setVerticalScrollBarPolicy(ScrollPaneConstants.***VERTICAL\_SCROLLBAR\_ALWAYS***);

scrollPane\_1.setHorizontalScrollBarPolicy(ScrollPaneConstants.***HORIZONTAL\_SCROLLBAR\_ALWAYS***);

scrollPane.setViewportView(scrollPane\_1);

JLabel lblNewLabel\_2 = **new** JLabel("分区号 分区大小 分区始址 状态");

lblNewLabel\_2.setFont(**new** Font("宋体", Font.***PLAIN***, 12));

lblNewLabel\_2.setBounds(555, 37, 212, 15);

contentPane.add(lblNewLabel\_2);

textField = **new** JTextField();

textField.setFont(**new** Font("宋体", Font.***PLAIN***, 15));

textField.setBounds(75, 357, 75, 25);

contentPane.add(textField);

textField.setColumns(10);

textField\_1 = **new** JTextField();

textField\_1.setBounds(370, 357, 75, 25);

contentPane.add(textField\_1);

textField\_1.setColumns(10);

}

}

**class** SortByPriority **implements** Comparator<PCB> { // 按优先级

@Override

**public** **int** compare(PCB p1, PCB p2) {

**if** (p1.priority > p2.priority)

**return** 1;

**else** **if** (p1.priority == p2.priority && p1.cometime > p2.cometime) // 若优先级相同，则先来先服务

**return** 1;

**return** -1;

}

}

**class** SortByStart **implements** Comparator<Free> { // 实现起点比较

@Override

**public** **int** compare(Free f1, Free f2) {

**if** (f1.start > f2.start)

**return** 1;

**return** -1;

}

}