A Ticket to Success

James W Cooper

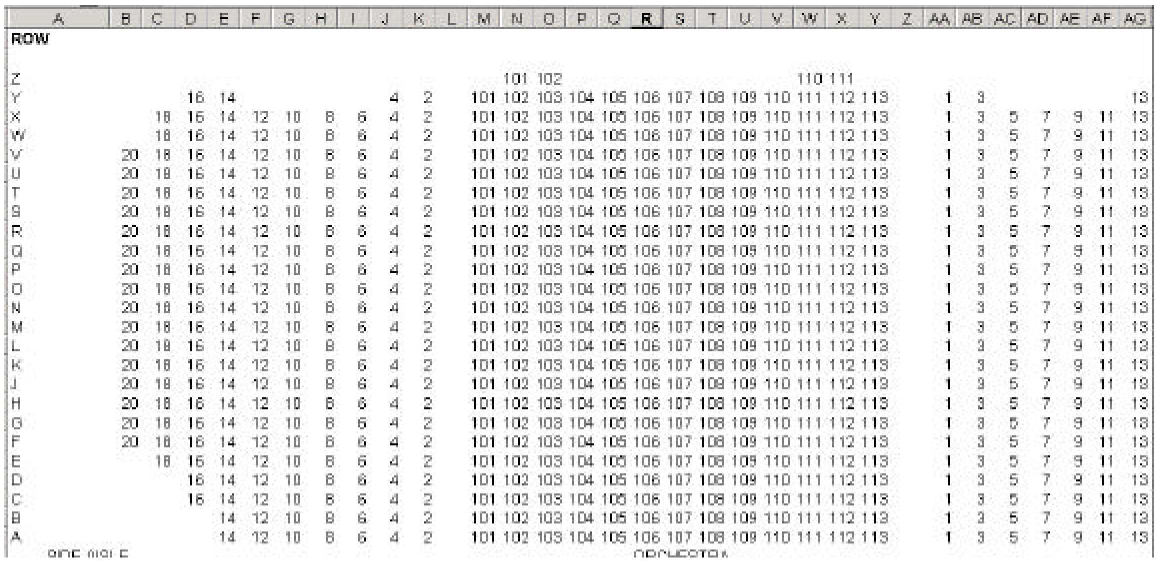
Robin: *I recognize you. You are the picture that hangs at the end of the gallery.*

Roderic: *In a bad light I am.*

Sometimes a problem presents itself in a way you don't expect so that it is not obvious that a simple Java program is the quickest solution. For example, I belong to a community theater group that puts on one comic opera a year for 5 performances. There are not enough performances, to justify complicated ticketing software, but the problem of selling reserved seats and producing tickets at the lowest cost continues to bedevil us.

Now it is perfectly possible to buy laser printer blank ticket forms. I bought ours from *perforatedpaper.com*. The problem is converting a list of seats in a theater into tickets and printing them out. One of the problems that we have to solve is that most theaters are not square: rows have different numbers of seats, and often the seat number system is not exactly linear. For example, many theaters number their prime center block of seats with 3 digits, and the less desirable side blocks with single digits, with the odd seats on one side and the even seats on the other. So it is not productive to spend time on a analytic algorithm for computing seat numbers. Instead, we need to start with a list of seats.

Figure 1 shows a section of the seat layout for the auditorium our company is using.



A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC	AD	AE	AF	AG
ROW																																
Z													101	102								110	111									
Y			16	14					4	2		101	102	103	104	105	106	107	108	109	110	111	112	113		1	3					13
X		18	16	14	12	10	8	6	4	2		101	102	103	104	105	106	107	108	109	110	111	112	113		1	3	5	7	9	11	13
W		18	16	14	12	10	8	6	4	2		101	102	103	104	105	106	107	108	109	110	111	112	113		1	3	5	7	9	11	13
V	20	18	16	14	12	10	8	6	4	2		101	102	103	104	105	106	107	108	109	110	111	112	113		1	3	5	7	9	11	13
U	20	18	16	14	12	10	8	6	4	2		101	102	103	104	105	106	107	108	109	110	111	112	113		1	3	5	7	9	11	13
T	20	18	16	14	12	10	8	6	4	2		101	102	103	104	105	106	107	108	109	110	111	112	113		1	3	5	7	9	11	13
S	20	18	16	14	12	10	8	6	4	2		101	102	103	104	105	106	107	108	109	110	111	112	113		1	3	5	7	9	11	13
R	20	18	16	14	12	10	8	6	4	2		101	102	103	104	105	106	107	108	109	110	111	112	113		1	3	5	7	9	11	13
Q	20	18	16	14	12	10	8	6	4	2		101	102	103	104	105	106	107	108	109	110	111	112	113		1	3	5	7	9	11	13
P	20	18	16	14	12	10	8	6	4	2		101	102	103	104	105	106	107	108	109	110	111	112	113		1	3	5	7	9	11	13
O	20	18	16	14	12	10	8	6	4	2		101	102	103	104	105	106	107	108	109	110	111	112	113		1	3	5	7	9	11	13
N	20	18	16	14	12	10	8	6	4	2		101	102	103	104	105	106	107	108	109	110	111	112	113		1	3	5	7	9	11	13
M	20	18	16	14	12	10	8	6	4	2		101	102	103	104	105	106	107	108	109	110	111	112	113		1	3	5	7	9	11	13
L	20	18	16	14	12	10	8	6	4	2		101	102	103	104	105	106	107	108	109	110	111	112	113		1	3	5	7	9	11	13
K	20	18	16	14	12	10	8	6	4	2		101	102	103	104	105	106	107	108	109	110	111	112	113		1	3	5	7	9	11	13
J	20	18	16	14	12	10	8	6	4	2		101	102	103	104	105	106	107	108	109	110	111	112	113		1	3	5	7	9	11	13
H	20	18	16	14	12	10	8	6	4	2		101	102	103	104	105	106	107	108	109	110	111	112	113		1	3	5	7	9	11	13
G	20	18	16	14	12	10	8	6	4	2		101	102	103	104	105	106	107	108	109	110	111	112	113		1	3	5	7	9	11	13
F	20	18	16	14	12	10	8	6	4	2		101	102	103	104	105	106	107	108	109	110	111	112	113		1	3	5	7	9	11	13
E		18	16	14	12	10	8	6	4	2		101	102	103	104	105	106	107	108	109	110	111	112	113		1	3	5	7	9	11	13
D			16	14	12	10	8	6	4	2		101	102	103	104	105	106	107	108	109	110	111	112	113		1	3	5	7	9	11	13
C			16	14	12	10	8	6	4	2		101	102	103	104	105	106	107	108	109	110	111	112	113		1	3	5	7	9	11	13
B				14	12	10	8	6	4	2		101	102	103	104	105	106	107	108	109	110	111	112	113		1	3	5	7	9	11	13
A				14	12	10	8	6	4	2		101	102	103	104	105	106	107	108	109	110	111	112	113		1	3	5	7	9	11	13
SIDE AISLE																	ORCHESTRA															

Figure 1 – A portion of the seat layout.

As you may deduce from the borders, this layout was typed into an Excel spreadsheet. It is probably possible to write a program to read from the Excel file in Java, perhaps using JDBC, but it is far easier to simpler export the data into a comma-separated text file and write a program to read it in.

A portion of that file looks like this:

```
M,20,18,16,14,12,10,8,6,4,2, ,101,102,103,104,105,106,107,
L,20,18,16,14,12,10,8,6,4,2, ,101,102,103,104,105,106,107,
K,20,18,16,14,12,10,8,6,4,2, ,101,102,103,104,105,106,107,
```

where we represent aisles and missing seats with a blank instead of a seat number.

We can design a Java class to represent a seat (and whether it is sold) simply enough:

```
/**Represents a single seat*/
public class Seat {
      //the seat location
      private String row, number;

      //number of customer ordering seat
      private int customerKey;

      public Seat(String row, String number) {
            this.number = number.trim();
            this.row = row;
      }
      /**returns true is this is a blank seat
       * Occurs when rows are different lengths
       * and for aisles.*/
      public boolean isBlank() {
            return number.length() < 1;
      }

      /**Returns the seat number.
       * @return String
       */
      public String getNumber() {
            if (isBlank())
                  return ".";
            else{
                  while(number.length() <3) {
                        number += " ";
                  }
                  return number;
            }
      }
```

Similarly, we can design a SeatRow class to present rows of seats as a Vector of seat objects.

```
/**represents a row of seats*/
public class SeatRow {
      private Vector seats;   //list of seats in row
      private String rowName;

      public SeatRow(String rowName) {
            seats = new Vector();
            this.rowName =rowName;
      }
      //add a seat to the row
      public void addSeat(Seat st) {
            seats.addElement( st);
      }
      public String getRowName() {
            return rowName;
      }
      //get an iterator for the seats in the row
      public Iterator getIterator() {
            return seats.iterator() ;
      }
}
```

Reading the Seating Plan

It is of course quite simple to read in the seat rows from this file using a StringTokenizer:

```
      rowList = new Vector();
      seatList = new Vector();
      try {
            InputFile inf = new InputFile("seating.txt");
            String srow = inf.readLine();
            while (srow != null) {
                  StringTokenizer tok = new StringTokenizer(srow, ",");
                  String rowName = tok.nextToken();

                  sRow = new SeatRow(rowName);
                  while (tok.hasMoreTokens()) {
                        String snum = tok.nextToken();
                        String tnum = snum;
                        if (snum.trim().length() < 1) {
                              tnum = ".";
                        }
                        Seat seat = new Seat(rowName, snum);
                        sRow.addSeat(seat);
                  }
                  rowList.add(sRow);
                  srow = inf.readLine();
            }
      } catch (IOException e) {
            System.out.println(e.getMessage());
      }
```

But, how do we print the tickets? This is the hard part in Java, because printing has the somewhat deserved reputation for innate complexity.

Printing in Java

Since Java 1.2, printing has become a good deal more friendly, however, and it is this newer set of classes we'll use to print out our tickets. In Java 1.2, you can print a set of pages all having the same format, as we'll be doing here, or you can establish a Book class where different pages have different formats.

In experimenting with the printing classes I found a couple of little quirks that took a little time to straighten out, and the code I show is the result of these experiments.

You need to provide a class for printing a page of tickets which implements the Printable interface. This interface, has the one method:

```
int print(Graphics graphics, PageFormat pageFormat, int pageIndex);
```

If you look at this interface, you see that the graphics surface you will be drawing to is supplied, along with an object describing the page format. But the most important thing that is passed to your printing class is the page number. You have to be prepared to print any page requested, and in any order. You must return PAGE_EXISTS if that page can be printed and NO_SUCH_PAGE if it cannot be printed. Without going into the formatting details of a single ticket, here is the loop that prints out 20 tickets on a page:

```
//called by printing system
      //note that this simplistic example only prints page 0
      public int print(Graphics gr, PageFormat pageFormat, int page)
            throws PrinterException {
            if (page == 0) {
                  this.fmt = pageFormat;
                  this.g = (Graphics2D) gr;

                  printPage();
                  return PAGE_EXISTS;
            } else
                  return NO_SUCH_PAGE;
      }
      //print all the seats on one page
      private void printPage() {
            Iterator iter = seatList.iterator();
            int count = 0;

            while (iter.hasNext() && count < 20) {
                  String seat = (String) iter.next();
                  printSeat(seat);        //print one seat
                  count++;
            }
      }
```

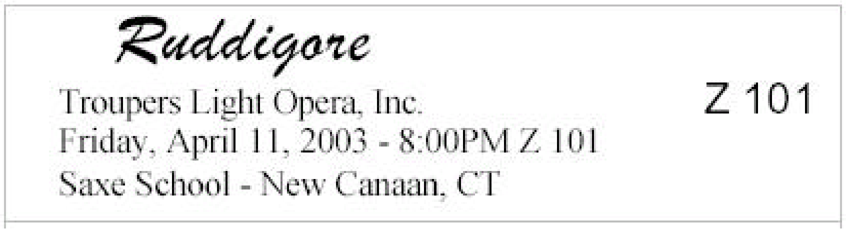
The details of printing an actually ticket will include actual positioning and font information and can be as messy as any other printing or graphics program. Here is a portion of the code for printing a ticket:

```
private void printSeat(String seat) {
      //draw ticket borders in light gray
      g.setColor(Color.LIGHT_GRAY);
      g.drawRect(xoff, yoff, (int) (3.9 * INCH), INCH);
      //print seat info in black
      g.setColor(Color.BLACK);

      //print seat number in Arial (sans serif)
      Font arial = new Font("Arial", Font.PLAIN , 14);
      g.setFont( arial);
      g.drawString(seat, xoff +3 * INCH+INCH/4, yoff + INCH / 2);

      //print title of show in script
      Font script = new Font("Brush script", Font.BOLD, 24);
      g.setFont( script);
      g.drawString( "Ruddigore", xoff+INCH/2, yoff+ INCH/4);
```

The final ticket is shown in Figure 2.



Ruddigore
Troupers Light Opera, Inc.
Friday, April 11, 2003 - 8:00PM Z 101
Saxe School - New Canaan, CT
Z 101

Figure 2 – A single ticket printed by our program.

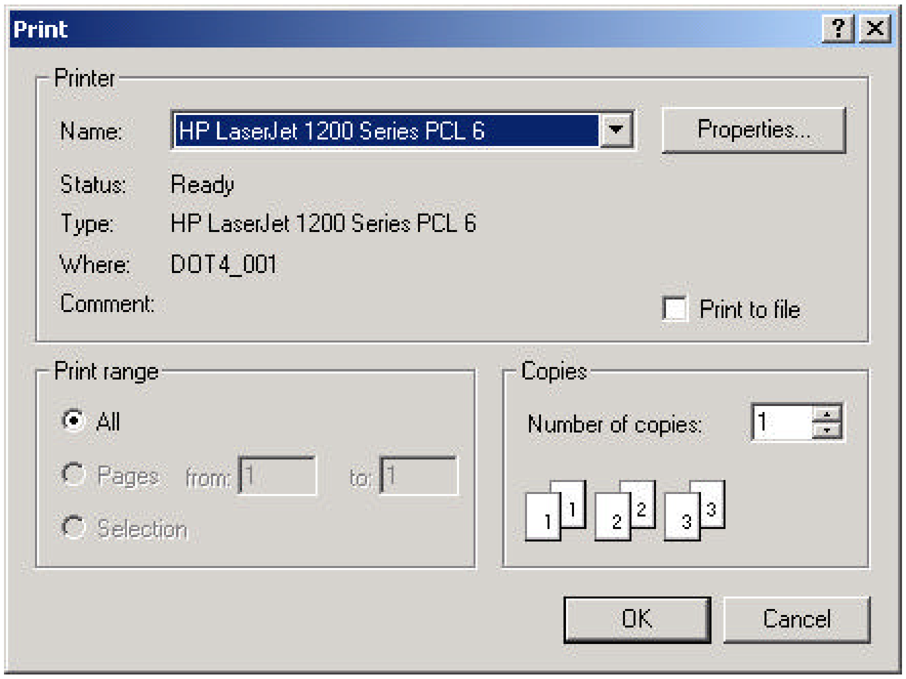
Setting up Printing

Now that we've shown you how to print a single ticket, we need to explain how you set up the whole system. You have to define a PageFormat and a Paper type. In addition, it seems empirically that printing works most reliably if you do not assume any defaults for any of these values.

You start by getting the current printer job from the static PrinterJob method:

```
//get the current print job
      pJob = PrinterJob.getPrinterJob();
```

Then, you can display the printer dialog box, if you want to allow the user to change from the default printer, as shown in Figure 3.



Print

Printer

Name: HP LaserJet 1200 Series PCL 6

Status: Ready

Type: HP LaserJet 1200 Series PCL 6

Where: DOT4_001

Comment:

Print to file

Print range

[✓ All]

Pages from: 1 to: 1

Selection

Copies

Number of copies: 1

Figure 3 – The printer dialog box.

If the user selects Cancel from that dialog box, the printDIalog method returns false and the printing does not take place.

```
if (pJob.printDialog()) {
      print(v);
      try {
            pJob.print();
```

```
        } catch (PrinterException e) {
              System.out.println(e.getMessage());
        }
}
```

The “pixel resolution” on a page is 1/72 of an inch, so we define the constant:

```
private final int INCH = 72;
```

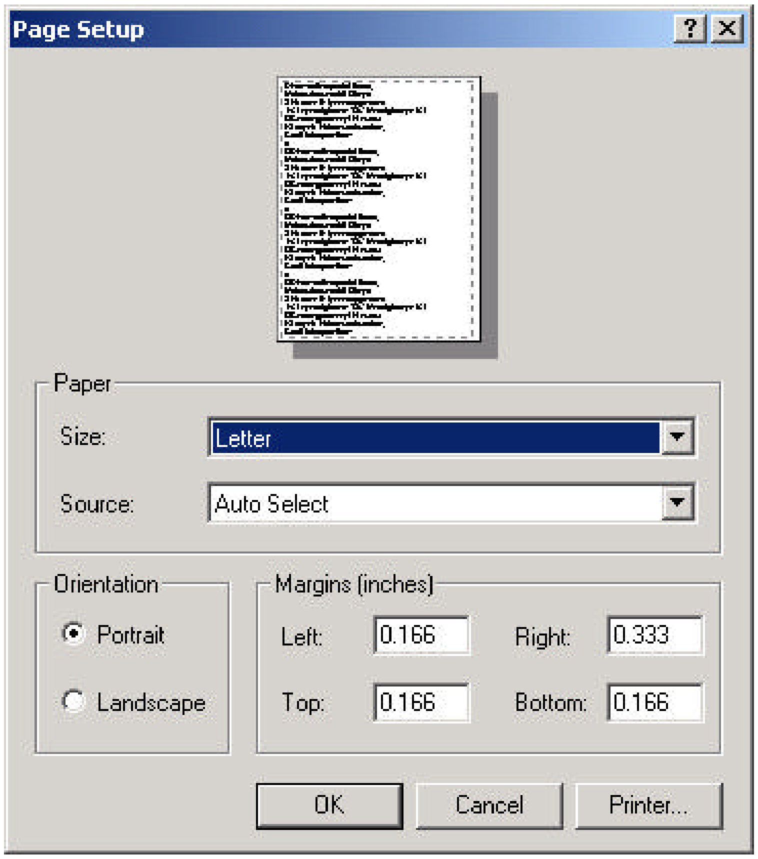
To set up the printer page, we set the page and paper format as we show below:

```
//set up a format
PageFormat fmt = pJob.defaultPage();
//define the orientation
fmt.setOrientation(PageFormat.PORTRAIT);
//set the paper size
Paper paper = fmt.getPaper();
paper.setSize(8.5 * INCH, 11 * INCH);
//define the full area as imageable
paper.setImageableArea(0 * INCH, 0 * INCH, 8.5 * INCH, 11 * INCH);
//set this paper as the current format
fmt.setPaper(paper);
```

Then, if you like, you can display the page format dialog:

```
      //show the page format dialog for the heck of it
      pJob.pageDialog( fmt);
```

which looks like that shown in Figure 4.



Size: Letter
Source: Auto Select
Orientation: [✓ Portrait] Landscape
Margins (inches)
Left: 0.166 Right: 0.333
Top: 0.166 Bottom: 0.166

Figure 4 – The page format dialog box.

Then, the final thing you must do is to create an instance of the object implementing the Printable interface and pass it and the format information to the printer job:

```
      //create a printable page
      pJob.setPrintable(new TicketPrinter(v), fmt);
```

This begins the printing process and the pubic *print* method of the TicketPrinter object us called as needed.

Displaying Our Results

Now, suppose we want to display this seating chart in Java. We can use the JTable component to display the matrix of seats. However, since we have a matrix of seats and not-seats (aisles and missing seats) we need to display them differently. We do this by telling the Jtable about 2 renderers, one for real seats and one for labels for the columns that are not actual seats:

```
JScrollPane sp = new JScrollPane(table);
table.setPreferredScrollableViewportSize(new Dimension(820,500));
//set renderer for non seats
table.setDefaultRenderer(String.class, new LabelRenderer());
//set renderer for seats
table.setDefaultRenderer(Seat.class, new SeatRenderer());
```

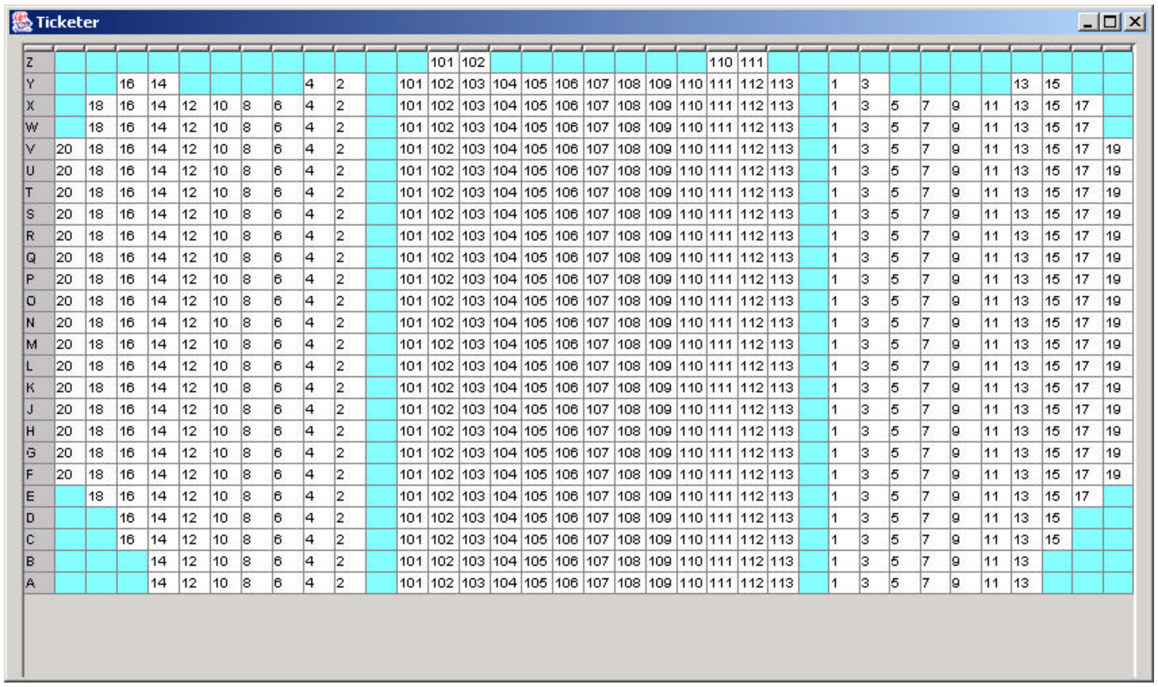
The SeatRenderer displays the seat number if there is one, and if there is not, displays a blue background for aisles and missing seats:

```
public class SeatRenderer extends DefaultTableCellRenderer {
      protected Font bold, plain;

      public SeatRenderer() {
            super();
            setOpaque(true);
            setBackground(Color.white);
            setForeground(Color.black);
            bold = new Font("SansSerif", Font.BOLD, 10);
            plain = new Font("SansSerif", Font.PLAIN, 10);
            setFont(plain);
      }
//--------------------
      public Component getTableCellRendererComponent(
            JTable jt,
            Object value,
            boolean isSelected,
            boolean hasFocus,
            int row,
            int column) {
            Seat seat = (Seat) value;
            String number = seat.getNumber();
            setBackground(Color.WHITE);
            //for missing seats and aisles
            //set the background to light blue
            if (seat.getNumber().trim().equals(".")) {
                setBackground(new Color(0x80, 0xff, 0xff));
                number = "";
            }
            setText(number);
            return this;
        }

}
```
The LabelRenderer just sets the background to gray and displays no text. It is used for the left hand column of row letters.

```
public class LabelRenderer extends SeatRenderer {
      public LabelRenderer() {
            super();
      }
      //----------
      public Component getTableCellRendererComponent(
            JTable jt,
            Object value,
            boolean isSelected,
            boolean hasFocus,
            int row,
            int column) {
            setFont(bold);
            setText((String) value);
            setBackground(Color.lightGray);
            return this;
      }
}
```
The final display is shown in Figure 6.


Ticketer

Z													101	102								110	111												
Y			16	14					4	2		101	102	103	104	105	106	107	108	109	110	111	112	113		1	3					13	15		
X		18	16	14	12	10	8	6	4	2		101	102	103	104	105	106	107	108	109	110	111	112	113		1	3	5	7	9	11	13	15	17	
W		18	16	14	12	10	8	6	4	2		101	102	103	104	105	106	107	108	109	110	111	112	113		1	3	5	7	9	11	13	15	17	
V	20	18	16	14	12	10	8	6	4	2		101	102	103	104	105	106	107	108	109	110	111	112	113		1	3	5	7	9	11	13	15	17	19
U	20	18	16	14	12	10	8	6	4	2		101	102	103	104	105	106	107	108	109	110	111	112	113		1	3	5	7	9	11	13	15	17	19
T	20	18	16	14	12	10	8	6	4	2		101	102	103	104	105	106	107	108	109	110	111	112	113		1	3	5	7	9	11	13	15	17	19
S	20	18	16	14	12	10	8	6	4	2		101	102	103	104	105	106	107	108	109	110	111	112	113		1	3	5	7	9	11	13	15	17	19
R	20	18	16	14	12	10	8	6	4	2		101	102	103	104	105	106	107	108	109	110	111	112	113		1	3	5	7	9	11	13	15	17	19
Q	20	18	16	14	12	10	8	6	4	2		101	102	103	104	105	106	107	108	109	110	111	112	113		1	3	5	7	9	11	13	15	17	19
P	20	18	16	14	12	10	8	6	4	2		101	102	103	104	105	106	107	108	109	110	111	112	113		1	3	5	7	9	11	13	15	17	19
O	20	18	16	14	12	10	8	6	4	2		101	102	103	104	105	106	107	108	109	110	111	112	113		1	3	5	7	9	11	13	15	17	19
N	20	18	16	14	12	10	8	6	4	2		101	102	103	104	105	106	107	108	109	110	111	112	113		1	3	5	7	9	11	13	15	17	19
M	20	18	16	14	12	10	8	6	4	2		101	102	103	104	105	106	107	108	109	110	111	112	113		1	3	5	7	9	11	13	15	17	19
L	20	18	16	14	12	10	8	6	4	2		101	102	103	104	105	106	107	108	109	110	111	112	113		1	3	5	7	9	11	13	15	17	19
K	20	18	16	14	12	10	8	6	4	2		101	102	103	104	105	106	107	108	109	110	111	112	113		1	3	5	7	9	11	13	15	17	19
J	20	18	16	14	12	10	8	6	4	2		101	102	103	104	105	106	107	108	109	110	111	112	113		1	3	5	7	9	11	13	15	17	19
H	20	18	16	14	12	10	8	6	4	2		101	102	103	104	105	106	107	108	109	110	111	112	113		1	3	5	7	9	11	13	15	17	19
G	20	18	16	14	12	10	8	6	4	2		101	102	103	104	105	106	107	108	109	110	111	112	113		1	3	5	7	9	11	13	15	17	19
F	20	18	16	14	12	10	8	6	4	2		101	102	103	104	105	106	107	108	109	110	111	112	113		1	3	5	7	9	11	13	15	17	19
E		18	16	14	12	10	8	6	4	2		101	102	103	104	105	106	107	108	109	110	111	112	113		1	3	5	7	9	11	13	15	17	
D			16	14	12	10	8	6	4	2		101	102	103	104	105	106	107	108	109	110	111	112	113		1	3	5	7	9	11	13	15		
C			16	14	12	10	8	6	4	2		101	102	103	104	105	106	107	108	109	110	111	112	113		1	3	5	7	9	11	13	15		
B				14	12	10	8	6	4	2		101	102	103	104	105	106	107	108	109	110	111	112	113		1	3	5	7	9	11	13			
A				14	12	10	8	6	4	2		101	102	103	104	105	106	107	108	109	110	111	112	113		1	3	5	7	9	11	13			

Figure 5 – A display of the seat plan using a JTable.

On with the show!

We have seen how to read and print out theater tickets and how to display the results on the screen. Wouldn’t it be great if we could keep track of ticket sales as well? Well, that must be part of the second act, but after this Act I Finale, perhaps you return after the interval to see what additional plot elements we’ve stirred up.