Sample Applications - Case Studies

Now, you are equipped with the simple concepts of GUI programming and are ready to develop some solutions using GUI Application for real life problems. On day to day basis we come across so many problems, which are required to be solved with the help of arithmetic and logical operations. This chapter will guide you through the process of developing applications with the help of some sample solutions for such problems.

Types of Applications

There are several types of applications, which can be developed to simplify our life. All of us must have used many such applications several times such as playing games online, buying movie tickets online, making train/air reservations etc. These applications are broadly categorized as e-Gaming, e-Business, e-Governance, e-Management, e-Learning etc. e-Gaming involves gaming applications on computers or internet. e-Business involves applications dealing with buying and selling of products and services. e-Governance involves applications which are used for administration and management of an organization. e-Management involves applications dealing with day to day management of organizations. e-Learning involves applications which are developed to help learning of any concept.

Case Study 1 - Cross N Knot Game

Problem Statement

You must have played this game several times using pen and paper, yes it is the same game. The aim is to develop a two-player digital version of the same game. Let us first analyze the problem and decide on the requirements before starting the coding.

Problem Analysis

Cross n Knot is a two player game. The aim of the application is to allow the players to decide on the position where they want to place the symbol (in our case cross for player

one and knot for player two by default) one by one in a 3 x 3 grid. So the first thing we should have on the form is 9 text fields with suitable labels for placing the symbols. Note that these text fields should be disabled so that they just display the position without being edited.

The game starts with Player 1 and he has to decide on the position where he wants to place the cross. He will enter this input in a text field and similarly player 2 needs to enter his position in a text field. So the next thing we need to have on the form is 2 text fields to accept the inputs of Player 1 and Player 2 respectively. Note that the Labels against the text fields are necessary as they help in identifying the position where the player is going to place the symbol (cross or knot).

After the player decides on the input he should click on a button to place the symbol in the appropriate position and transfer the turn to the other player. To achieve this we should have two buttons, one for each player.

Apart from these controls we need to have two more buttons - first for restarting the game and second for ending the application.

The game continues till all the text fields are filled .To avoid confusion, when Player 1 is playing, only his controls have to be enabled and the controls of Player 2 have to be disabled. As soon as Player 1 finishes his turn by clicking on the button, the controls of Player 1 should be disabled and the controls for the Player 2 should be enabled.

Problem Solution

The first step is to design the form according to the above analysis. Carefully observe the Form given in Figure 11.1 and design a similar looking form with the mentioned characteristics.

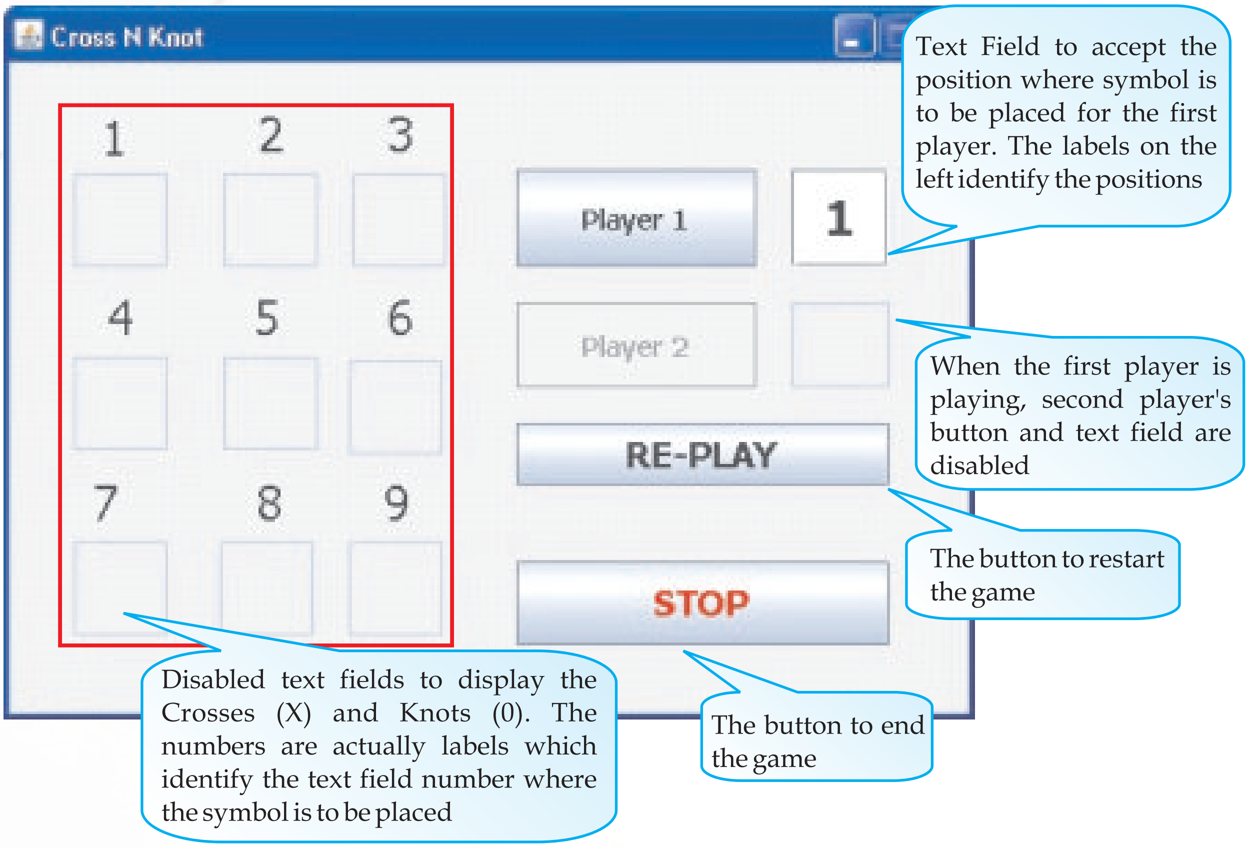

Cross N Knot
1
2
3
4
5
6
7
8
9
Player 1
1
Player 2
RE-PLAY
STOP
Text Field to accept the position where symbol is to be placed for the first player. The labels on the left identify the positions
When the first player is playing, second player's button and text field are disabled
The button to restart the game
The button to end the game
Disabled text fields to display the Crosses (X) and Knots (0). The numbers are actually labels which identify the text field number where the symbol is to be placed

Figure 11.1 Controls for the Cross N Knot Game Application

Let us first now summarize the functionalities required from each of the four buttons of the form - Player 1, Player 2, RE-PLAY and STOP and simultaneously look at the code for each of these buttons.

[Player 1] button should:

- Retrieve the position where "X" is to be placed from the jtextfield10 and display an error message if the input position is less than 1 or greater than 9.
- Check if the input position is empty then place the "X" in the relevant text field else display an error message that the position is occupied.
- Check if any of the three consecutive places are occupied by "X". If three consecutive positions are occupied by "X" then display a message that player 1 wins else check if all the text fields are occupied with no consecutive "X" then display a message that game is draw else set text field11 as editable, text field10 as non editable and also enable the Player 2 button and disable the Player 1 button.

Figure 11.2 CODE FOR [PLAYER1] BUTTON

```
 private void jButton1ActionPerformed
 (java.awt.event.ActionEvent evt) {
int Input1=Integer.parseInt(jTextField10.getText());
if (Input1<1 || Input1>9)
  JOptionPane.showMessageDialog
             (this,"Enter a Valid Position   (1..9)");
else
{
  if ((Input1==1 && jTextField1.getText().isEmpty()) ||
       (Input1==2 && jTextField2.getText().isEmpty()) ||
       (Input1==3 && jTextField3.getText().isEmpty()) ||
       (Input1==4 && jTextField4.getText().isEmpty()) ||
       (Input1==5 && jTextField5.getText().isEmpty()) ||
       (Input1==6 && jTextField6.getText().isEmpty()) ||
       (Input1==7 && jTextField7.getText().isEmpty()) ||
       (Input1==8 && jTextField8.getText().isEmpty()) ||
       (Input1==9 && jTextField9.getText().isEmpty())
     )
  {
    switch (Input1)
    {
      case 1:jTextField1.setText("X");break;
      case 2:jTextField2.setText("X");break;
      case 3:jTextField3.setText("X");break;
      case 4:jTextField4.setText("X");break;
      case 5:jTextField5.setText("X");break;
      case 6:jTextField6.setText("X");break;
```

```
        case 7:jTextField7.setText("X");break;
        case 8:jTextField8.setText("X");break;
        case 9:jTextField9.setText("X");break;
      }
       if ((jTextField1.getText().equals("X") &&
            jTextField2.getText().equals("X") &&
            jTextField3.getText().equals("X"))||
           (jTextField4.getText().equals("X") &&
            jTextField5.getText().equals("X") &&
            jTextField6.getText().equals("X"))||
           (jTextField7.getText().equals("X") &&
            jTextField8.getText().equals("X") &&
            jTextField9.getText().equals("X"))||
           (jTextField1.getText().equals("X") &&
            jTextField4.getText().equals("X") &&
            jTextField7.getText().equals("X"))||
           (jTextField2.getText().equals("X") &&
            jTextField5.getText().equals("X") &&
            jTextField8.getText().equals("X"))||
           (jTextField3.getText().equals("X") &&
            jTextField6.getText().equals("X") &&
            jTextField9.getText().equals("X"))||
           (jTextField1.getText().equals("X") &&
            jTextField5.getText().equals("X") &&
            jTextField9.getText().equals("X"))||
           (jTextField3.getText().equals("X") &&
            jTextField5.getText().equals("X") &&
```

```
           jTextField7.getText().equals("X"))
       )
        jOptionPane.showMessageDialog(this,"Player 1 Wins");
    else
      if (!jTextField1.getText().isEmpty() &&
          !jTextField2.getText().isEmpty() &&
          !jTextField3.getText().isEmpty() &&
          !jTextField4.getText().isEmpty() &&
          !jTextField5.getText().isEmpty() &&
          !jTextField6.getText().isEmpty() &&
          !jTextField7.getText().isEmpty() &&
          !jTextField8.getText().isEmpty() &&
          !jTextField9.getText().isEmpty())
        jOptionPane.showMessageDialog(this,"It is a DRAW...");
      else
        jTextField11.setEditable(true);

   jButton1.setEnabled(false);
   jButton2.setEnabled(true);
   jTextField10.setEditable(false);
  }
  else
    jOptionPane.showMessageDialog
              (this,"Already Occupied Option RE-ENTER (1..9)");
}
jTextField10.setText("");
}
```

[Player 2] button should:

- ❖ Retrieve the position where "O" is to be placed from the jtextfield11 and display an error message if the input position is less than 1 or greater than 9.
- ❖ Check if the input position is empty then place the "O" in the relevant text field else display an error message that the position is occupied.
- ❖ Check if any of the three consecutive places are occupied by "O". If three consecutive positions are occupied by "O" then display a message that player 2 wins else check if all the text fields are occupied with no consecutive "O" then display a message that game Is draw else set text field10 as editable, text field11 as non editable and also enable the Player 1 button and disable the Player 2 button.

Figure 11.3 CODE FOR [PLAYER2] BUTTON

```
 private void jButton2ActionPerformed
  (java.awt.event.ActionEvent evt) {
int Input2=Integer.parseInt(jTextField11.getText());
if (Input2<1 || Input2>9)
  JOptionPane.showMessageDialog
                (this,"Enter a Valid Position (1..9)");
else
{
  if ((Input2==1 && jTextField1.getText().isEmpty()) ||
       (Input2==2 && jTextField2.getText().isEmpty()) ||
       (Input2==3 && jTextField3.getText().isEmpty()) ||
       (Input2==4 && jTextField4.getText().isEmpty()) ||
       (Input2==5 && jTextField5.getText().isEmpty()) ||
       (Input2==6 && jTextField6.getText().isEmpty()) ||
       (Input2==7 && jTextField7.getText().isEmpty()) ||
       (Input2==8 && jTextField8.getText().isEmpty()) ||
        (Input2==9 && jTextField9.getText().isEmpty())
       )
    {
      switch (Input2)
      {
        case 1:jTextField1.setText("O");break;
        case 2:jTextField2.setText("O");break;
        case 3:jTextField3.setText("O");break;
        case 4:jTextField4.setText("O");break;
        case 5:jTextField5.setText("O");break;
        case 6:jTextField6.setText("O");break;
        case 7:jTextField7.setText("O");break;
        case 8:jTextField8.setText("O");break;
        case 9:jTextField9.setText("O");break;
      }
       if ((jTextField1.getText().equals("O") &&
            jTextField2.getText().equals("O") &&
            jTextField3.getText().equals("O"))||
           (jTextField4.getText().equals("O") &&
            jTextField5.getText().equals("O") &&
            jTextField6.getText().equals("O"))||
           (jTextField7.getText().equals("O") &&
            jTextField8.getText().equals("O") &&
            jTextField9.getText().equals("O"))||
           (jTextField1.getText().equals("O") &&
            jTextField4.getText().equals("O") &&
            jTextField7.getText().equals("O"))||
           (jTextField2.getText().equals("O") &&
            jTextField5.getText().equals("O") &&
            jTextField8.getText().equals("O"))||
           (jTextField3.getText().equals("O") &&
            jTextField6.getText().equals("O") &&
            jTextField9.getText().equals("O"))||
           (jTextField1.getText().equals("O") &&
            jTextField5.getText().equals("O") &&
            jTextField9.getText().equals("O"))||
           (jTextField3.getText().equals("O") &&
             jTextField5.getText().equals("O") &&
             jTextField7.getText().equals("O"))
            )
        jOptionPane.showMessageDialog(this,"Player 2 Wins");
      else
        if (!jTextField1.getText().isEmpty() &&
            !jTextField2.getText().isEmpty() &&
            !jTextField3.getText().isEmpty() &&
            !jTextField4.getText().isEmpty() &&
            !jTextField5.getText().isEmpty() &&
            !jTextField6.getText().isEmpty() &&
            !jTextField7.getText().isEmpty() &&
            !jTextField8.getText().isEmpty() &&
            !jTextField9.getText().isEmpty()
           )
           jOptionPane.showMessageDialog(this,"It is a DRAW...");
```

```
       else
          jTextField10.setEditable(true);
     jButton2.setEnabled(false);
     jButton1.setEnabled(true);
     jTextField11.setEditable(false);
   }
   else
 jOptionPane.showMessageDialog
              (this,"Already Occupied Option RE-ENTER (1..9)");
 }
 jTextField11.setText("");
 }
```

[RE-PLAY] button should:

- ❖ Clear all the 9 text fields
- ❖ Enable the Player 1 input text field & disable the Player 2 input text field
- ❖ Enable the Player 1 button and disable the Player 2 button

Figure 11.4 CODE FOR [RE-PLAY] BUTTON

```
private void jButton4ActionPerformed
(java.awt.event.ActionEvent evt) {
    jTextField1.setText("");
    jTextField2.setText("");
    jTextField3.setText("");
    jTextField4.setText("");
    jTextField5.setText("");
    jTextField6.setText("");
    jTextField7.setText("");
    jTextField8.setText("");
    jTextField9.setText("");
    jTextField10.setEditable(true);
    jTextField11.setEditable(false);
    jButton1.setEnabled(true);
    jButton2.setEnabled(false);
}
```

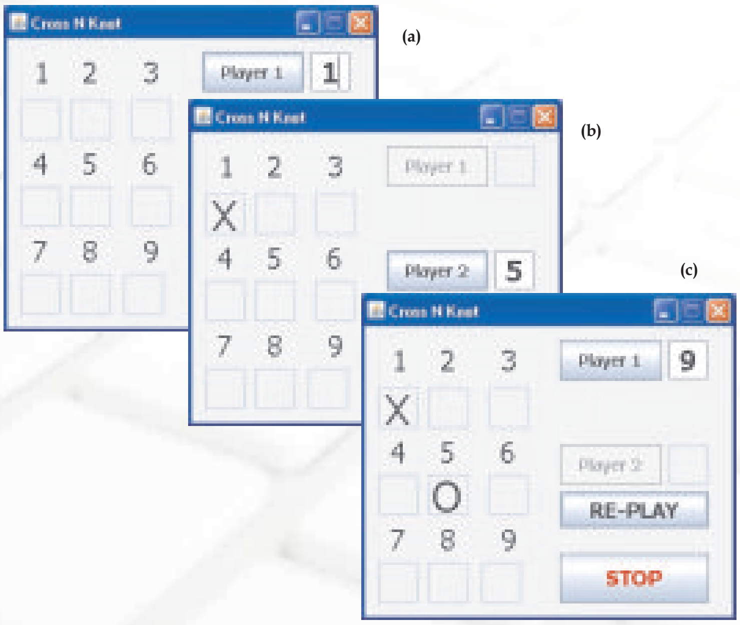
[STOP] button should:

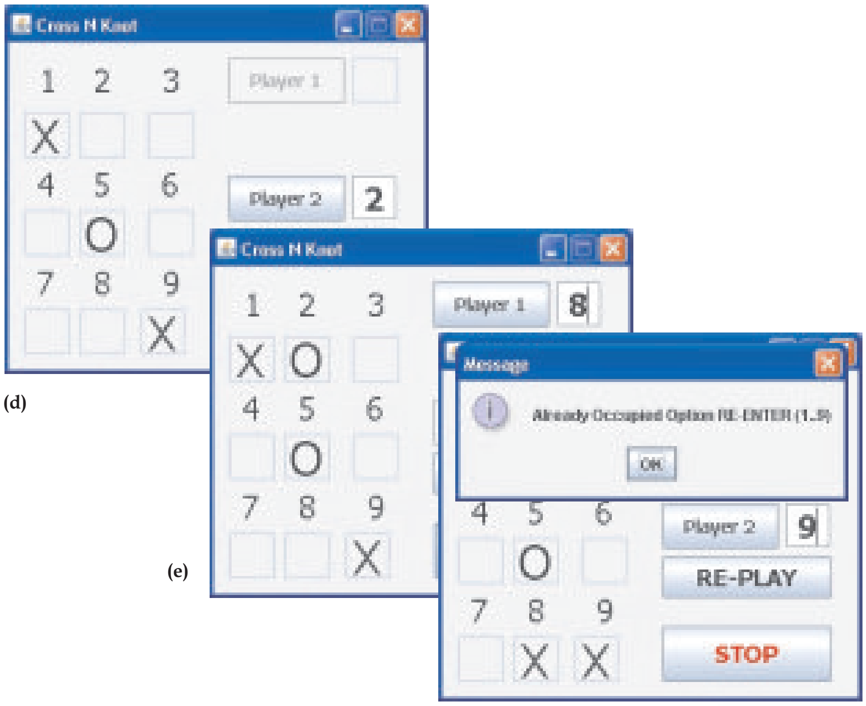
- ❖ Exit from the application

Figure 11.5 CODE FOR [STOP] BUTTON

```
private void jButton3ActionPerformed
(java.awt.event.ActionEvent evt) {
   System.exit(0);
}
```

Figures 11.6(a) - 11.6(k) show a complete sample run of the application along with the appropriate message boxes.





Already Occupied Option RE-ENTER (1..9)

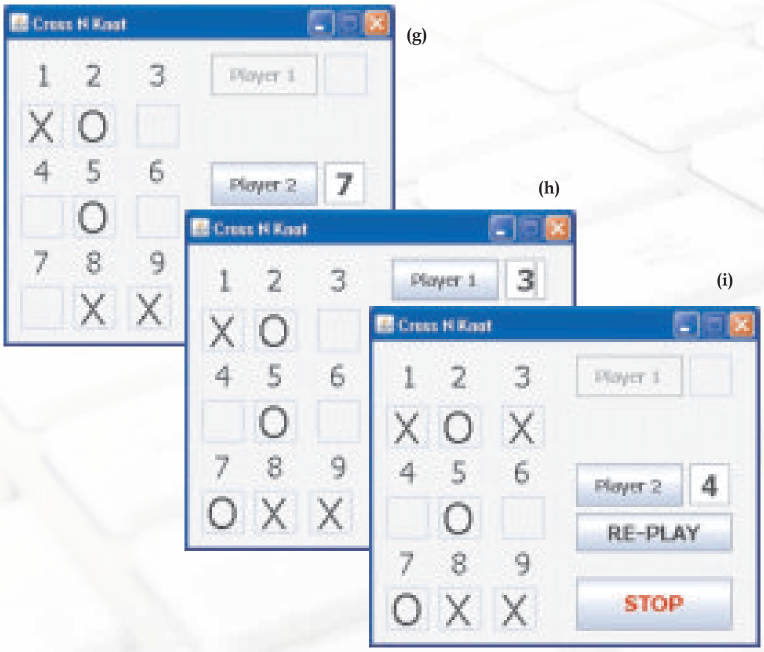
(d)

(e)

(f)

Figure 11.6 Sample Run for the Cross N Knot Game Application

If the user enters a position which is already occupied, an error message is displayed as shown in Figure 11.6(f) and the user has to renter the position to continue with the game.



(g)

(h)

(i)

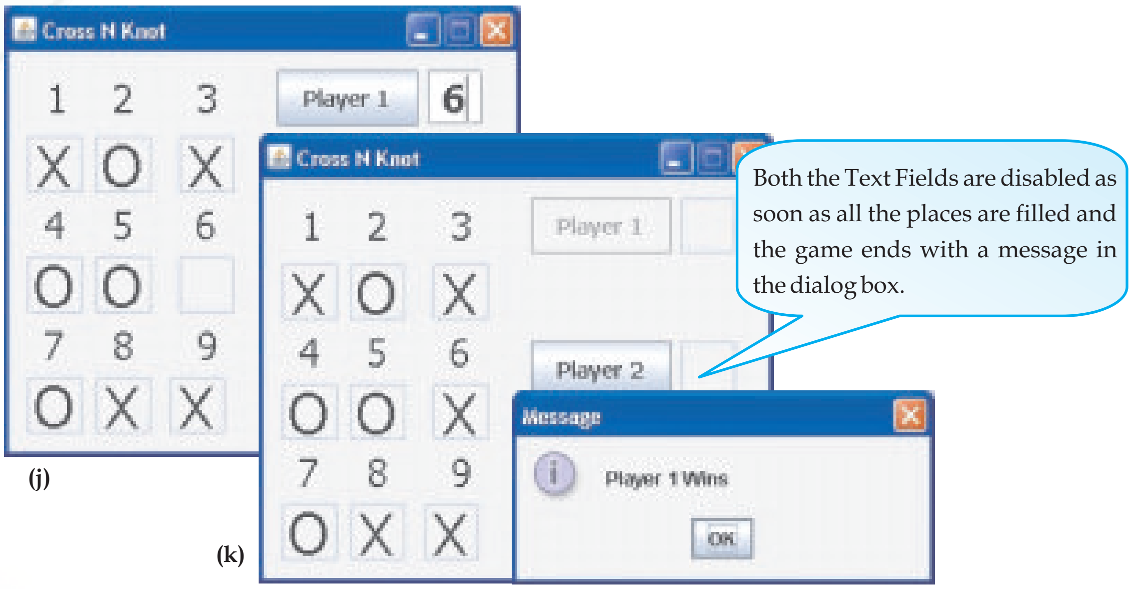

Cross N Knot
1 2 3
Player 1
6
4 5 6
7 8 9
(j)
Cross N Knot
Player 1
Player 2
(k)
Message
Player 1 Wins
OK
Both the Text Fields are disabled as soon as all the places are filled and the game ends with a message in the dialog box.

Figure 11.6 Sample Run for the Cross N Knot Game Application

Case Study 2 - Salary Calculator

Problem Statement

Payroll Calculation is a very commonly used management application. Here we will try to create a Salary Calculator using almost all of the controls learnt in this book. The basic aim of the application is to calculate the Salary in Hand based on input provided by the user.

Problem Analysis

As stated above, the basic aim of the application is to calculate the Salary in Hand based on various criteria selected by the user. The criterion include:

- Basic Salary input using a text field
- Designation input using a Combo Box (as only one of the option from a large number of known set of options is required to be taken)
- Status input using a Radio Button Group (as one of the option out of limited number of known set of options are required to be taken)
- Area input using a Radio Button
- Allowance input using a Check Box (as multiple options are required to be selected from a limited number of known set of options)

Apart from these controls which are used to accept the input, we need a few more controls as enumerated below:

- *Five buttons* - first for calculating the Earnings, second for calculating the Deductions, third for calculating the Salary in Hand, fourth to reset the form and fifth for closing the application. Only the first button should be enabled initially and the rest should be disabled.
- *Eight text fields with relevant labels* - four to display the Earnings as DA, HRA, Other and Total Earnings and the remaining four to display the Deductions due to Income Tax, PF, Social Contribution and Total Deductions. All these text fields should be disabled as they are just being used to display values and hence should not be editable.
- *One text field with appropriate label* - to display the Salary in Hand. This text field should also be disabled as it is also only being used to display value.

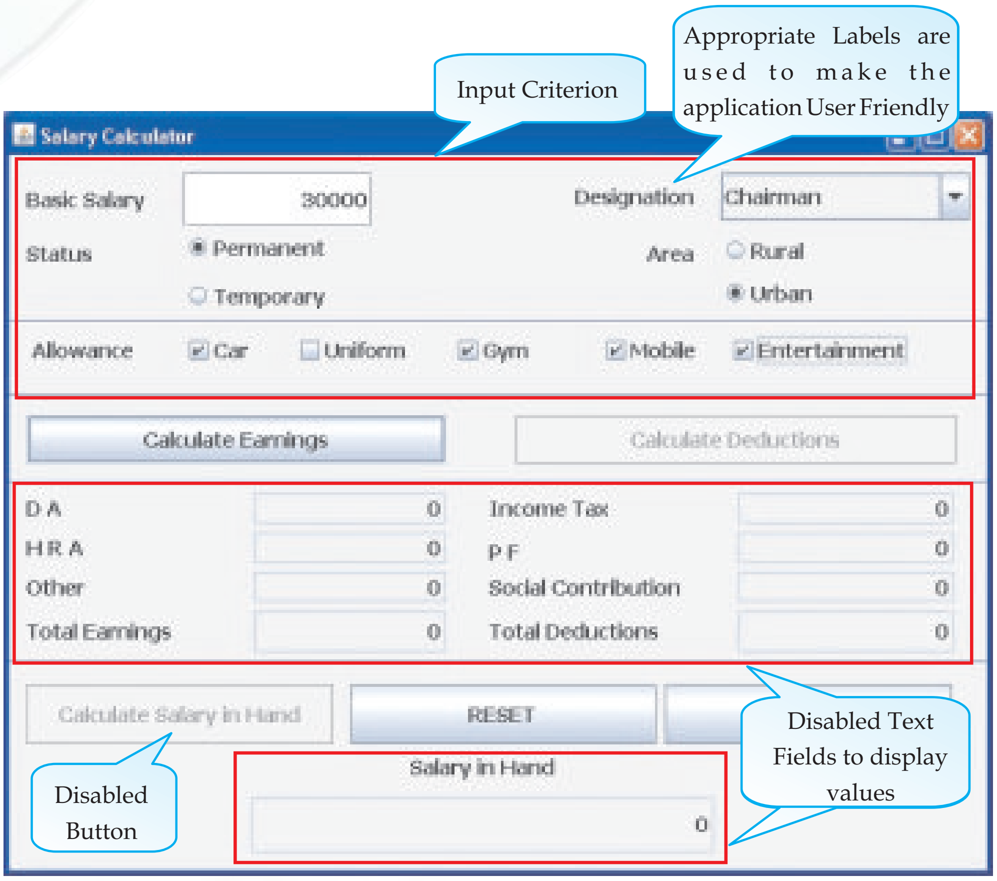
Initially only the Calculate Earnings, RESET and STOP buttons should be enabled. As soon as the user inputs information regarding the Basic Salary, Designation, Status, Area and Allowance, the *CALCULATE EARNINGS* button can be clicked. On the click of this button, the DA, HRA, Other and Total Earnings are calculated and displayed in the disabled text fields.

Immediately the *CALCULATE DEDUCTIONS* button is enabled and the user can click on this button to see the deductions due to Income Tax, PF, Social Contribution and the Total Deductions in the disabled text fields adjacent to the relevant labels.

When the deductions are displayed, the *CALCULATE SALARY IN HAND* button is enabled. On the click of this button, the total salary in hand is calculated and displayed in the disabled text field at the bottom of the form.

Problem Solution

The first step is to design the form according to the above analysis. Carefully observe the form given in Figure 11.7 and design a similar looking form with the above mentioned characteristics.



Basic Salary 30000
Designation Chairman
Status [✓ Permanent] Temporary
Area Rural [✓ Urban]
Allowance [✓ Car] Uniform [✓ Gym] [✓ Mobile] [✓ Entertainment]

D A	0	Income Tax	0
H R A	0	P F	0
Other	0	Social Contribution	0
Total Earnings	0	Total Deductions	0

Salary in Hand 0

Figure 11.7 Accepting the Input to Calculate Salary in Hand using the Salary Calculator

Let us now summarize the functionalities required from each of the five buttons of the form - CALCULATE EARNINGS, CALCULATE DEDUCTIONS, CALCULATE SALARY IN HAND, RESET and STOP and simultaneously look at the code for each of these buttons.

[Calculate Earnings] button should:

- Retrieve the basic salary input in the first text field and convert it into double type.
- Calculate DA, HRA and Other Allowance earnings based on the selection criteria input by the user (using the radio buttons and checkboxes). Also calculate the Total earnings by adding the three values DA, HRA and Other.

- Display the calculated values of DA, HRA, Other Allowance and Total Earnings in the disabled text fields second, third, fourth and eighth respectively after converting them to String type.
- Enable the *CALCULATE DEDUCTIONS* button.

Figure 11.8 CODE FOR [Calculate Earnings] BUTTON

```
 private void jButton1ActionPerformed
 (java.awt.event.ActionEvent evt) {
    double Basic;
    double DA, HRA, Others=0, Earnings;
    Basic=Double.parseDouble(jTextField1.getText());
    // Calculation of DA
    if (jRadioButton1.isSelected())
    {
        if (jComboBox1.getSelectedIndex()==0)
           DA=0.50*Basic;           //50% of Basic Salary
        else if (jComboBox1.getSelectedIndex()==1)
            DA=0.60*Basic;           //60% of Basic Salary
        else if (jComboBox1.getSelectedIndex()==2)
            DA=0.80*Basic;           //80% of Basic Salary
        else if (jComboBox1.getSelectedIndex()==3)
        DA=0.90*Basic;           //90% of Basic Salary
        else
            DA=0;
    }
    else
      DA=1000;
  // Calculation of HRA
if (jRadioButton3.isSelected())      //--> RURAL AREA
```

```
    HRA=4000;
  else if (jRadioButton4.isSelected())//--> URBAN AREA
    HRA=8000;
  else
    HRA=0;

  // Calculation of Other Allowance
  if (jCheckBox1.isSelected())  //CAR
    Others=3000;
  if (jCheckBox2.isSelected())  //Uniform
    Others=1000;
  if (jCheckBox3.isSelected())  //Gym
    Others=2000;
  if (jCheckBox4.isSelected())  //Mobile
    Others=1500;
  if (jCheckBox5.isSelected())  //Entertainment
    Others=500;
  Earnings=DA+HRA+Others;
  jTextField2.setText(Double.toString(DA));
  jTextField3.setText(Double.toString(HRA));
  jTextField4.setText(Double.toString(Others));
  jTextField8.setText(Double.toString(Earnings));
  jButton2.setEnabled(true);
}
```

[Calculate Deductions] button should:

❖ Retrieve the basic salary input in the first text field and the calculated DA displayed in the second text field and convert them to double type.

❖ Calculate Income Tax, Provident Fund and Social deductions based on the status input by the user (using the radio button). Also calculate the Total Deductions by adding the three values IT, PF and Social.

❖ Display the calculated values of IT, PF, Social and Total Deductions in the disabled text fields fifth, sixth, seventh and ninth respectively after converting them to String type.

❖ Enable the *CALCULATE SALARY IN HAND* button.

Figure 11.9 CODE FOR [Calculate Deductions] BUTTON

```
private void jButton2ActionPerformed(java.awt.event.ActionEvent
evt) {
 double BasicDA;
 double DA, IT, PF=0, Social=0, Deductions;
 DA=Double.parseDouble(jTextField2.getText());
 BasicDA=Double.parseDouble(jTextField1.getText())+DA;
 // Calculation of IT (Income Tax)
 if (jRadioButton1.isSelected())
 {
   if (BasicDA>=30000)
     IT=0.30*BasicDA;            //10% of (Basic Salary+DA)
   else if (BasicDA>=15000)
     IT=0.20*BasicDA;            //20% of (Basic Salary+DA)
   else
     IT=0.10*BasicDA;            //30% of (Basic Salary+DA)
   PF=0.10*BasicDA;              //Calculation of Provident Fund
   Social=2000;                 //Calculation of Social Deduction
 }
 else
   IT=1000;
   Deductions=IT+PF+Social;    //Calculation of Total Deductions
 jTextField5.setText(Double.toString(IT));
 jTextField6.setText(Double.toString(PF));
 jTextField7.setText(Double.toString(Social));
 jTextField9.setText(Double.toString(Deductions));
 jButton3.setEnabled(true);
}
```

[Calculate Salary in Hand] button should:

- Retrieve the basic salary input in the first text field and the calculated total earnings and total deductions displayed in the eighth and ninth text field and convert them to double type.
- Calculate the Total In Hand salary by adding Basic Salary and Total Earnings and subtracting the Total Deductions.
- Display the calculated Total in Hand Salary in the tenth text field after converting it to String type.

Figure 11.10 CODE FOR [Calculate Salary in Hand] BUTTON

```
 private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
    double Basic,Earnings,Deductions,InHand;
    Basic=Double.parseDouble(jTextField1.getText());
    Earnings=Double.parseDouble(jTextField8.getText());
    Deductions=Double.parseDouble(jTextField9.getText());
    InHand=Basic+Earnings-Deductions;
    jTextField10.setText(Double.toString(InHand));
 }
```

[RESET] button should:

- Initialize all the 10 text fields to 0
- Reset the Designation combo box

- Deselect all the radio buttons and checkboxes
- Disable the *CALCULATE DEDUCTIONS* and *CALCULATE SALARY IN HAND* buttons

Figure 11.11 CODE FOR [RESET] BUTTON

```
private void jButton4ActionPerformed
(java.awt.event.ActionEvent evt) {
  jTextField1.setText("0");
  jTextField2.setText("0");
  jTextField3.setText("0");
  jTextField4.setText("0");
  jTextField5.setText("0");
  jTextField6.setText("0");
  jTextField7.setText("0");
  jTextField8.setText("0");
  jTextField9.setText("0");
  jTextField10.setText("0");
  jComboBox1.setSelectedIndex(0);
  jRadioButton1.setSelected(false);
  jRadioButton2.setSelected(false);
  jRadioButton3.setSelected(false);
  jRadioButton4.setSelected(false);
  jCheckBox1.setSelected(false);
  jCheckBox2.setSelected(false);
  jCheckBox3.setSelected(false);
  jCheckBox4.setSelected(false);
  jCheckBox5.setSelected(false);
  jButton2.setEnabled(false);
  jButton3.setEnabled(false);
}
```

[STOP] button should:

- ❖ Exit from the application

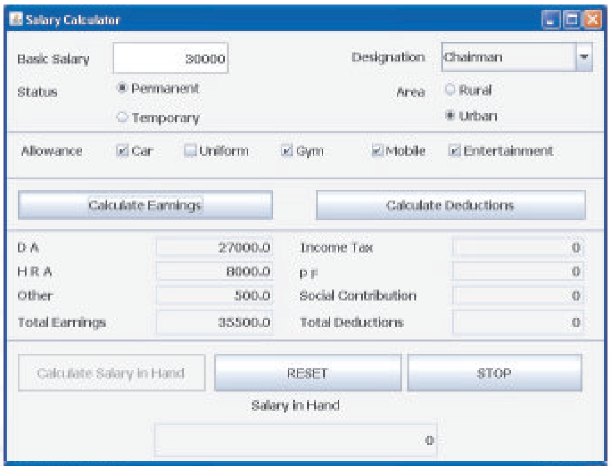
Figure 11.12 CODE FOR [STOP] BUTTON

```
private void jButton5ActionPerformed
(java.awt.event.ActionEvent evt) {

  System.exit(0);

}
```

Figure 11.7 shows the initial form where the user inputs data and is ready to click on the *CALCULATE EARNINGS* button. Figure 11.13 (a) and (b) show the effect of clicking on the Calculate Earnings and Calculate Salary in Hand respectively. At any point of time the user can click on the RESET button to restart or the STOP button to exit out of the Salary Calculator application.



Salary Calculator

Basic Salary: 30000
Designation: Chairman
Status: ● Permanent ○ Temporary
Area: ○ Rural ● Urban
Allowance: ☑ Car ☐ Uniform ☑ Gym ☑ Mobile ☑ Entertainment

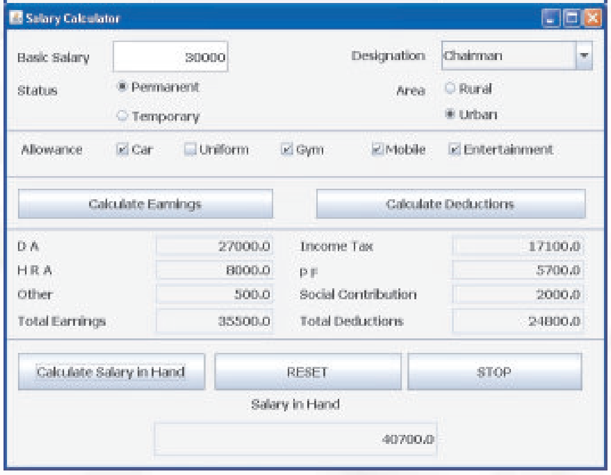
Calculate Earnings | Calculate Deductions

D A	27000.0	Income Tax	0
H R A	8000.0	P F	0
Other	500.0	Social Contribution	0
Total Earnings	35500.0	Total Deductions	0

Calculate Salary in Hand | RESET | STOP

Salary in Hand: 0

(a)



Salary Calculator

Basic Salary: 30000
Designation: Chairman
Status: ● Permanent ○ Temporary
Area: ○ Rural ● Urban
Allowance: ☑ Car ☐ Uniform ☑ Gym ☑ Mobile ☑ Entertainment

Calculate Earnings | Calculate Deductions

D A	27000.0	Income Tax	17100.0
H R A	8000.0	P F	5700.0
Other	500.0	Social Contribution	2000.0
Total Earnings	35500.0	Total Deductions	24800.0

Calculate Salary in Hand | RESET | STOP

Salary in Hand: 40700.0

(b)

Figure 11.13 Displaying all the Earnings on the Click of the Calculate Earnings Button

Case Study 3 - Restaurant Billing System

Problem Statement

Using software for managing orders placed by a buyer is a very common e-Business application. Here we will develop a sample e-business application used to place order in a restaurant. The basic aim of the application is to handle the order placed for maximum five items from a list of available items which can be selected one at a time from a drop down list and show the total bill amount.

Problem Analysis

Keeping in mind the aim of the application, let us first analyze the problem and decide on the various controls required for designing the form. The user needs to select the food items (maximum five) and also a suitable offer both input using combo box. The user also inputs the quantity for each selected item in a text field. Note that each time the user has to select a food item, its corresponding Offer and the Quantity before clicking on the MORE button (to place order for more items). If the user clicks on the MORE button then the corresponding price and amount are to be displayed and the controls for the next selection are activated. But if the user clicks on the FINISH button (marking the end of the order) then the total price and service tax are displayed and the CALCULATE AMOUNT button is to be enabled. So then to accept the input we need to have:

- *10 combo boxes,* five to accept the items and five to accept the offer for each of the five items selected.
- *5 text fields* to accept the quantity of each of the selected items.

Apart from these controls required to accept the inputs we require the following additional controls:

- *10 disabled text fields with appropriate labels,* 5 each to display the price and amount of each selected item.
- Additional *3 disabled text fields with appropriate labels* to display the total of all purchases, service tax and total amount payable.
- Set of *10 buttons* used to give a choice to the user to either purchase more items or end the order. Only one set of 2 of these buttons will be enabled at a time.

- ❖ Additional 3 buttons - one for resetting the form, one for ending the application and one used to calculate the total of all purchases, service tax and total amount payable. Out of these 3 buttons, the RESET and the STOP buttons are enabled throughout the application. The CALCULATE AMOUNT button is initially disabled and is enabled only when any one of the FINISH button is clicked marking the end of the order.

Note that initially only the RESET, STOP, Combo box 1, Combo box 2, Text Field1, first MORE and first FINISH buttons are enabled. After placing the order for an item, the user clicks on either the MORE button or the FINISH button but not both. The user clicks on MORE if he wants to purchase more items and clicks on FINISH if he wants to end the order. After analyzing the input and output requirements, let us now proceed with the problem solution.

Problem Solution

The first step is to design the form according to the above analysis. Carefully observe the Form given in Figure 11.14 and design a similar looking form with the above mentioned characteristics for all the controls.

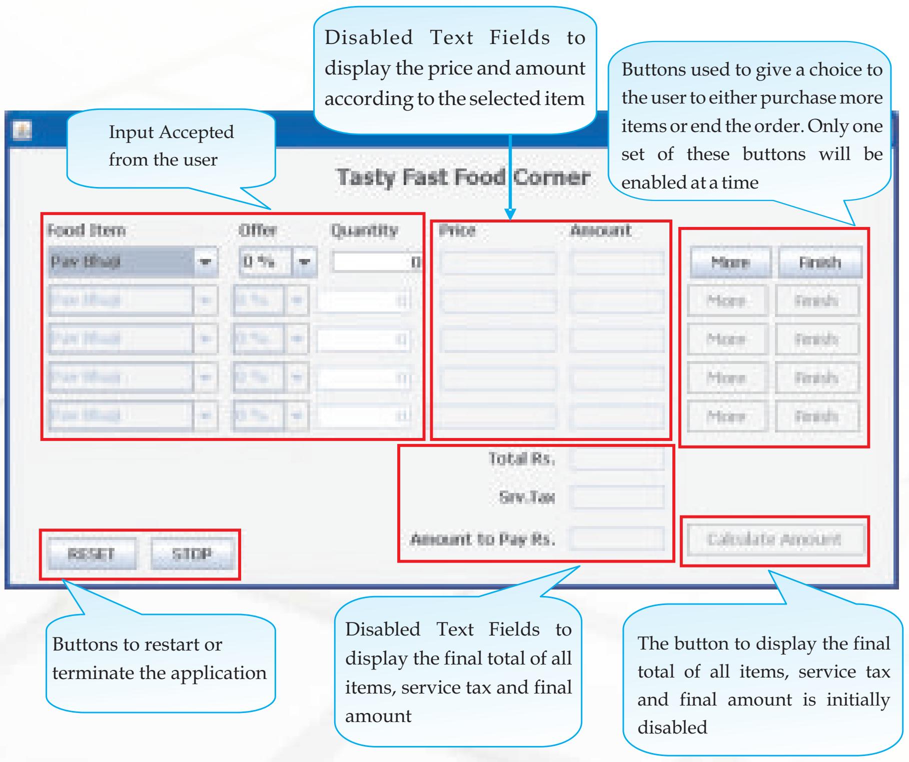

Disabled Text Fields to display the price and amount according to the selected item
Buttons used to give a choice to the user to either purchase more items or end the order. Only one set of these buttons will be enabled at a time
Input Accepted from the user
Tasty Fast Food Corner
Food Item
Offer
Quantity
Price
Amount
Pav Bhaji
0 %
0
More
Finish
Total Rs.
Srv.Tax
Amount to Pay Rs.
RESET
STOP
Calculate Amount
Buttons to restart or terminate the application
Disabled Text Fields to display the final total of all items, service tax and final amount
The button to display the final total of all items, service tax and final amount is initially disabled

Figure 11.14 Controls for the Restaurant Billing System

Let us now summarize the functionalities required from each of the buttons of the form - MORE (There are five such buttons), FINISH (There are five such buttons), CALCULATE AMOUNT, RESET and STOP and simultaneously look at the code for each of these buttons. The functionality of each of the MORE buttons is almost the same with the exception that each time different controls are being enabled and disabled. Note this carefully while observing the code for the first and second MORE button.

[More-1] button should:

- Retrieve the index of the selected item and display the price of the selected item in the second text field.
- Retrieve the quantity and price from the first and second text field respectively and convert these to type double.
- Calculate the amount by multiplying the above two values and display it in the third text field after converting it to type string.
- Enable the controls for the next selection (i.e. enable combo box 3 and 4, text field 3 and buttons 3 and 4) and disable the controls of the current selection (i.e. disable combo box 1 and 2, text field 1 and buttons 1 and 2).

Figure 11.15 CODE FOR [MORE-1] BUTTON

```
private void jButton1ActionPerformed
 (java.awt.event.ActionEvent evt) {
   switch (jComboBox1.getSelectedIndex())
   {
       case 0:jTextField2.setText("100");break; //Pav Bhaji
       case 1:jTextField2.setText("80"); break; //Bhel Puri
       case 2:jTextField2.setText("120");break; //Chhole Kulche
       case 3:jTextField2.setText("110");break; //Burger
       case 4:jTextField2.setText("200");break; //Pizza
       default:jTextField2.setText("0");
   }
```

```
     double Amount,Offer,Price,Qty;
     Qty=Double.parseDouble(jTextField1.getText());
     switch (jComboBox2.getSelectedIndex())
     {
       case 1 :Offer=10;break;
       case 2 :Offer=20;break;
       default:Offer=0;
     }
     Price=Double.parseDouble(jTextField2.getText());
     Amount=(Price*Qty)*(100-Offer)/100;
     jTextField3.setText(Double.toString(Amount));
     jComboBox3.setEnabled(true);
     jComboBox4.setEnabled(true);
     jTextField4.setEnabled(true);
     jButton3.setEnabled(true);
     jButton4.setEnabled(true);
     jComboBox1.setEnabled(false);
     jComboBox2.setEnabled(false);
     jTextField1.setEnabled(false);
     jButton1.setEnabled(false);
     jButton2.setEnabled(false);
   }
```

[Finish-1] button should:

- Retrieve the index of the selected item and display the price of the selected item in the second text field.
- Retrieve the quantity and price from the first and second text field respectively and convert these to type double.

❖ Calculate the amount by multiplying the above two values and display it in the third text field after converting it to type string.

❖ Calculate the total amount and service tax and display it in the text field 16 & 17 corresponding to the Total Rs. and Srv. Tax labels, respectively after converting these to type string.

❖ Disable all the controls used to place the current order and enable the *CALCULATE AMOUNT* button.

Figure 11.16 CODE FOR [FINISH-1] BUTTON

```
private void jButton2ActionPerformed
(java.awt.event.ActionEvent evt) {
  switch (jComboBox1.getSelectedIndex())
  {
    case 0:jTextField2.setText("100");break; //Pav Bhaji
    case 1:jTextField2.setText("80"); break; //Bhel Puri
    case 2:jTextField2.setText("120");break; //Chhole Kulche
    case 3:jTextField2.setText("110");break; //Burger
    case 4:jTextField2.setText("200");break; //Pizza
    default:jTextField2.setText("0");
  }
  double Amount,Offer,Price,Qty;
  Qty=Double.parseDouble(jTextField1.getText());
  switch (jComboBox2.getSelectedIndex())
  {
    case 1 :Offer=10;break;
    case 2 :Offer=20;break;
    default:Offer=0;
  }
```

```
    Price=Double.parseDouble(jTextField2.getText());
    Amount=(Price*Qty)*(100-Offer)/100;
    double Stax;
    Stax=0.1* Amount;
    jTextField3.setText(Double.toString(Amount));
    jTextField16.setText(Double.toString(Amount));
    jTextField17.setText(Double.toString(Stax));
    jComboBox1.setEnabled(false);
    jComboBox2.setEnabled(false);
    jTextField1.setEnabled(false);
    jButton1.setEnabled(false);
    jButton2.setEnabled(false);
    jButton11.setEnabled(true);
  }
```

[More-2] button should:

- Retrieve the index of the second selected item and display the price of the selected item in the relevant text field.
- Retrieve the values of quantity and price from the relevant text fields and convert them to type double.
- Calculate the amount by multiplying the above two values and display it in the relevant text field after converting it to type string.
- Enable the controls for the next selection (i.e. enable combo box 5 and 6, text field 7 and buttons 5 and 6) and disable the controls of the current selection (i.e. disable combo box 3 and 4, text field 4 and buttons 3 and 4).

Figure 11.17 CODE FOR [MORE-2] BUTTON

```
private void jButton3ActionPerformed
(java.awt.event.ActionEvent evt) {
 switch (jComboBox3.getSelectedIndex())
 {
   case 0:jTextField5.setText("100");break; //Pav Bhaji
   case 1:jTextField5.setText("80"); break; //Bhel Puri
   case 2:jTextField5.setText("120");break; //Chhole Kulche
   case 3:jTextField5.setText("110");break; //Burger, Pizza
   case 4:jTextField5.setText("200");break; //Pizza
   default:jTextField5.setText("0");
 }
 double Amount,Offer,Price,Qty;
 Qty=Double.parseDouble(jTextField4.getText());
 switch (jComboBox4.getSelectedIndex())
 {
   case 1 :Offer=10;break;
   case 2 :Offer=20;break;
   default:Offer=0;
 }
 Price=Double.parseDouble(jTextField5.getText());
 Amount=(Price*Qty)*(100-Offer)/100;
 jTextField6.setText(Double.toString(Amount));
 jComboBox5.setEnabled(true);
 jComboBox6.setEnabled(true);
 jTextField7.setEnabled(true);
 jButton5.setEnabled(true);
```

```
  jButton6.setEnabled(true);
  jComboBox3.setEnabled(false);
  jComboBox4.setEnabled(false);
  jTextField4.setEnabled(false);
  jButton3.setEnabled(false);
  jButton4.setEnabled(false);
}
```

[Finish-2] button should:

- Retrieve the index of the second selected item and display the price of the selected item in the relevant text field.
- Retrieve the quantity and price from the relevant text fields and convert these to type double.
- Calculate the amount by multiplying the above two values and display it in the relevant text field after converting it to type string.
- Retrieve the amount values of all previous items selected and convert them to type double.
- Calculate the total amount and service tax and display it in the text fields 16 & 17 corresponding to the Total Rs. and Srv. Tax labels, respectively after converting them to type string.
- Disable all the controls used to place the current order and enable the *CALCULATE AMOUNT* button.

Figure 11.18 CODE FOR [FINISH-2] BUTTON

```
private void jButton4ActionPerformed
 (java.awt.event.ActionEvent evt) {
 switch (jComboBox3.getSelectedIndex())
 {
   case 0:jTextField5.setText("100");break; //Pav Bhaji
   case 1:jTextField5.setText("80"); break; //Bhel Puri
   case 2:jTextField5.setText("120");break; //Chhole Kulche
   case 3:jTextField5.setText("110");break; //Burger, Pizza
   case 4:jTextField5.setText("200");break; //Pizza
   default:jTextField5.setText("0");
 }
 double Amount,Amount1,Amount2,Offer,Price,Qty;
 Qty=Double.parseDouble(jTextField4.getText());
 switch (jComboBox4.getSelectedIndex())
 {
   case 1 :Offer=10;break;
   case 2 :Offer=20;break;
   default:Offer=0;
 }
 Price=Double.parseDouble(jTextField5.getText());
 Amount1=Double.parseDouble(jTextField3.getText());
 Amount2=(Price*Qty)*(100-Offer)/100;;
 Amount=Amount1+Amount2;
 double Stax;
 Stax=0.1* Amount;
 jTextField6.setText(Double.toString(Amount2));
 jTextField16.setText(Double.toString(Amount));
```

```
 jTextField17.setText(Double.toString(Stax));
 jComboBox3.setEnabled(false);
 jComboBox4.setEnabled(false);
 jTextField4.setEnabled(false);
 jButton3.setEnabled(false);
 jButton4.setEnabled(false);
 jButton11.setEnabled(true);
}
```

Now try and write the coding for the other MORE and FINISH buttons keeping in mind the fact that different controls have to be enabled and disabled each time. Also keep in mind that the MORE-5 button need not be enabled while writing the code for MORE-4 button. The functioning of the last FINISH button is given below to help you.

[Finish-5] button should:

- Retrieve the index of the selected item and display the price of the selected item in the relevant text field.
- Retrieve the quantity and price from the relevant text fields and also retrieve the values of all previous amounts and convert them to type double.
- Calculate the amount of the current selected item by multiplying the price and quantity and display it in the relevant text field after converting it to type string.
- Calculate the total amount (by adding all the previous amounts along with the current amount) and service tax and display it in the text field 16 & 17 corresponding to the Total Rs. and Srv. Tax labels, respectively after converting it to type string.
- Disable all the controls used to place the current order and enable the *CALCULATE AMOUNT* button.

Figure 11.19 CODE FOR [FINISH-5] BUTTON

```
private void jButton10ActionPerformed
                              (java.awt.event.ActionEvent evt)
{ switch (jComboBox9.getSelectedIndex())
 { case 0:jTextField14.setText("100");break; //Pav Bhaji
   case 1:jTextField14.setText("80"); break; //Bhel Puri
   case 2:jTextField14.setText("120");break; //Chhole Kulche
   case 3:jTextField14.setText("110");break; //Burger
   case 4:jTextField14.setText("200");break; //Pizza
   default:jTextField14.setText("0");
 }
 double Amount,Amount1,Amount2,Amount3,Amount4,
                       Amount5,Offer,Price,Qty,Stax;
 Qty=Double.parseDouble(jTextField13.getText());
 switch (jComboBox10.getSelectedIndex())
 {
   case 1 :Offer=10;break;
   case 2 :Offer=20;break;
   default:Offer=0;
 }
 Price=Double.parseDouble(jTextField14.getText());
 Amount1=Double.parseDouble(jTextField3.getText());
 Amount2=Double.parseDouble(jTextField6.getText());
 Amount3=Double.parseDouble(jTextField9.getText());
 Amount4=Double.parseDouble(jTextField12.getText());
 Amount5=(Price*Qty)*(100-Offer)/100;
 Amount=Amount1+Amount2+Amount3+Amount4+Amount5;
```

```
  Stax=0.1* Amount;
  jTextField15.setText(Double.toString(Amount5));
  jTextField16.setText(Double.toString(Amount));
  jTextField17.setText(Double.toString(Stax));
  jComboBox9.setEnabled(false);
  jComboBox10.setEnabled(false);
  jTextField15.setEnabled(false);
  jButton10.setEnabled(false);
  jButton11.setEnabled(true);
}
```

[Calculate Amount] button should:

- ❖ Retrieve the values of the Total Amount and the Service Tax from the text fields adjacent to the Total Rs. And Srv.Tax labels and convert them to type double.
- ❖ Calculate the Payable Amount by adding the above two values and display it in the relevant text field after converting the value to type string.

Figure 11.20 CODE FOR [Calculate Amount] BUTTON

```
private void jButton11ActionPerformed
                              (java.awt.event.ActionEvent evt)
{ double Amount,Stax,ToPay;
  Amount=Double.parseDouble(jTextField16.getText());
  Stax=Double.parseDouble(jTextField17.getText());
  ToPay=Amount+Stax;
  jTextField18.setText(Double.toString(ToPay));
}
```

[RESET] button should:

- Enable the first and second combo box and disable all the other combo boxes.
- Enable the first text field and set the display text as 0. Set the display text of all the other text fields as "" (i.e. blank).
- Enable the first *MORE* and first *FINISH* buttons and disable all the other *MORE* and *FINISH* buttons and disable the *CALCULATE AMOUNT* button.

Figure 11.21 CODE FOR [RESET] BUTTON

```
private void jButton12ActionPerformed
                  (java.awt.event.ActionEvent evt)
{
  jComboBox1.setEnabled(true);
  jComboBox2.setEnabled(true);
  jTextField1.setEnabled(true);
  jTextField1.setText("0");
  jTextField2.setText("");
  jTextField3.setText("");
  jButton1.setEnabled(true);
  jButton2.setEnabled(true);

  jComboBox3.setEnabled(false);
  jComboBox4.setEnabled(false);
  jTextField4.setEnabled(false);
  jTextField4.setText("0");
  jTextField5.setText("");
  jTextField6.setText("");
  jButton3.setEnabled(false);
  jButton4.setEnabled(false);

  jComboBox5.setEnabled(false);
  jComboBox6.setEnabled(false);
  jTextField7.setEnabled(false);
  jTextField7.setText("0");
  jTextField8.setText("");
  jTextField9.setText("");
  jButton5.setEnabled(false);
  jButton6.setEnabled(false);

  jComboBox7.setEnabled(false);
  jComboBox8.setEnabled(false);
  jTextField10.setEnabled(false);
  jTextField10.setText("0");
  jTextField11.setText("");
  jTextField12.setText("");
  jButton7.setEnabled(false);
  jButton8.setEnabled(false);

  jComboBox9.setEnabled(false);
  jComboBox10.setEnabled(false);
  jTextField13.setEnabled(false);
  jTextField13.setText("0");
  jTextField14.setText("");
  jTextField15.setText("");
  jButton9.setEnabled(false);
  jButton10.setEnabled(false);

  jTextField16.setText("");
  jTextField17.setText("");
  jTextField18.setText("");

  jButton11.setEnabled(false);
}
```

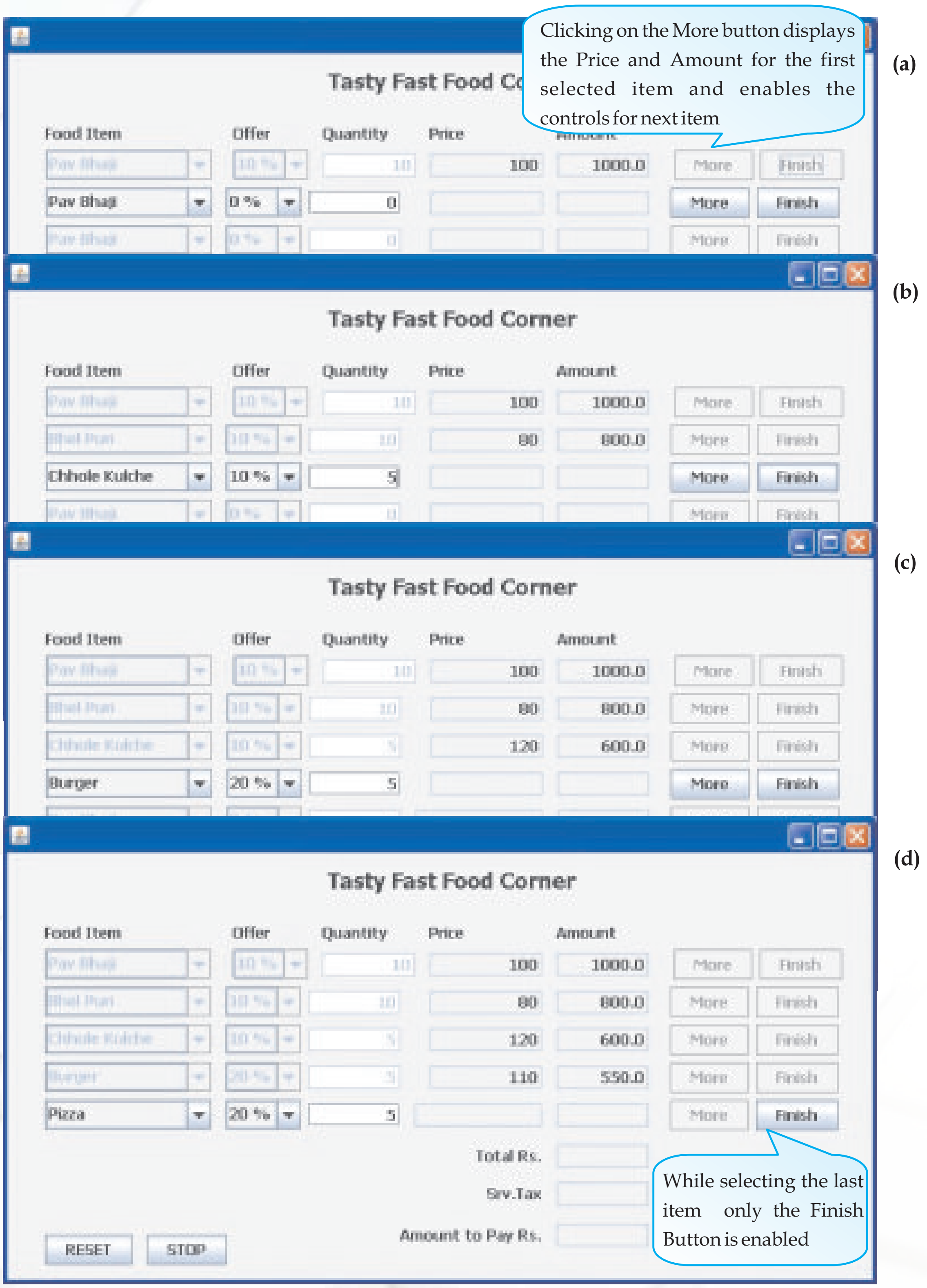
[STOP] button should:

- Exit from the application after displaying a message.

Figure 11.22 CODE FOR [STOP] BUTTON

```
private void jButton13ActionPerformed
                                        (java.awt.event.ActionEvent evt)
{
  JOptionPane.showMessageDialog
                      (null,"Bye - Come Back To order for more FOOD");
  System.exit(0);
}
```

The following figures show a sample run of the Restaurant Billing System application. Figures 11.23 (a) - (d) show the execution when the user places order for first four items. Figure 11.24 shows the effect of clicking on the *FINISH* button and Figure 11.25 shows the effect of clicking on the *CALCULATE AMOUNT* button. At any point of time the user can click on the *RESET* button to restart or the *STOP* button to exit out of the application.



(a)

Clicking on the More button displays the Price and Amount for the first selected item and enables the controls for next item

Tasty Fast Food Co

Food Item	Offer	Quantity	Price	Amount		
Pav Bhaji	10 %	10	100	1000.0	More	Finish
Pav Bhaji	0 %	0			More	Finish
Pav Bhaji	0 %	0			More	Finish

(b)

Tasty Fast Food Corner

Food Item	Offer	Quantity	Price	Amount		
Pav Bhaji	10 %	10	100	1000.0	More	Finish
Bhel Puri	10 %	10	80	800.0	More	Finish
Chhole Kulche	10 %	5			More	Finish
Pav Bhaji	0 %	0			More	Finish

(c)

Tasty Fast Food Corner

Food Item	Offer	Quantity	Price	Amount		
Pav Bhaji	10 %	10	100	1000.0	More	Finish
Bhel Puri	10 %	10	80	800.0	More	Finish
Chhole Kulche	10 %	5	120	600.0	More	Finish
Burger	20 %	5			More	Finish

(d)

Tasty Fast Food Corner

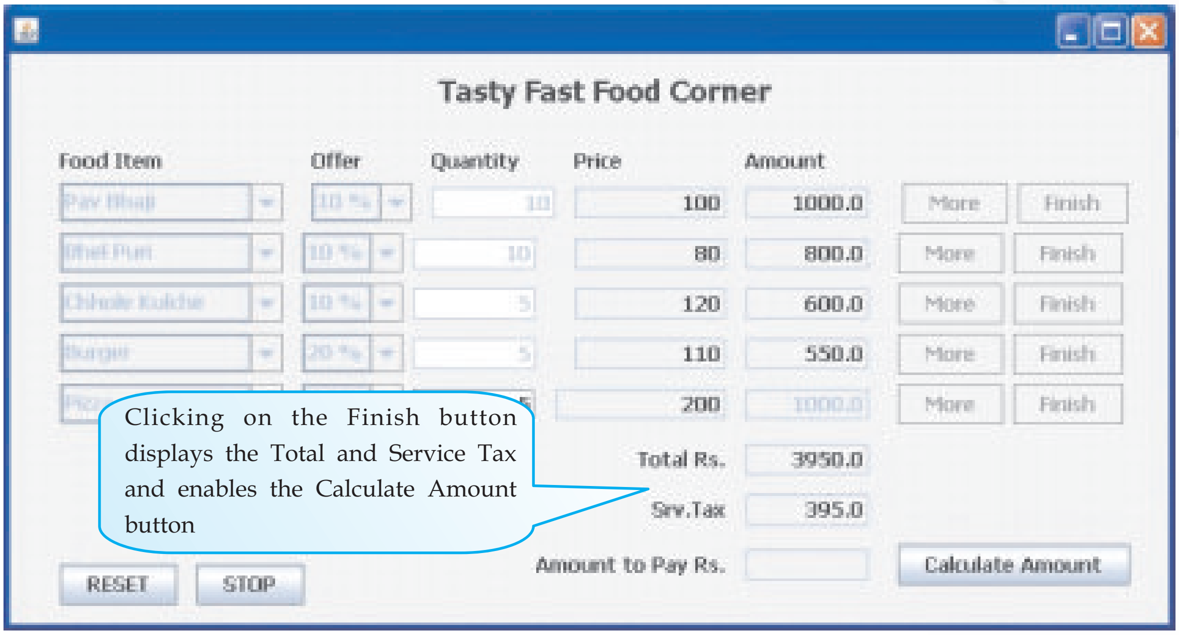
Food Item	Offer	Quantity	Price	Amount		
Pav Bhaji	10 %	10	100	1000.0	More	Finish
Bhel Puri	10 %	10	80	800.0	More	Finish
Chhole Kulche	10 %	5	120	600.0	More	Finish
Burger	20 %	5	110	550.0	More	Finish
Pizza	20 %	5			More	Finish

Total Rs.
Srv.Tax
Amount to Pay Rs.

RESET STOP

While selecting the last item only the Finish Button is enabled

Figure 11.23 Sample Run of the Restaurant Billing System showing the Selection of 5 items one after the other



Tasty Fast Food Corner

Food Item	Offer	Quantity	Price	Amount		
Pav Bhaji	10 %	10	100	1000.0	More	Finish
Bhel Puri	10 %	10	80	800.0	More	Finish
Chhole Kulche	10 %	5	120	600.0	More	Finish
Burger	20 %	5	110	550.0	More	Finish
Pizza		5	200	1000.0	More	Finish

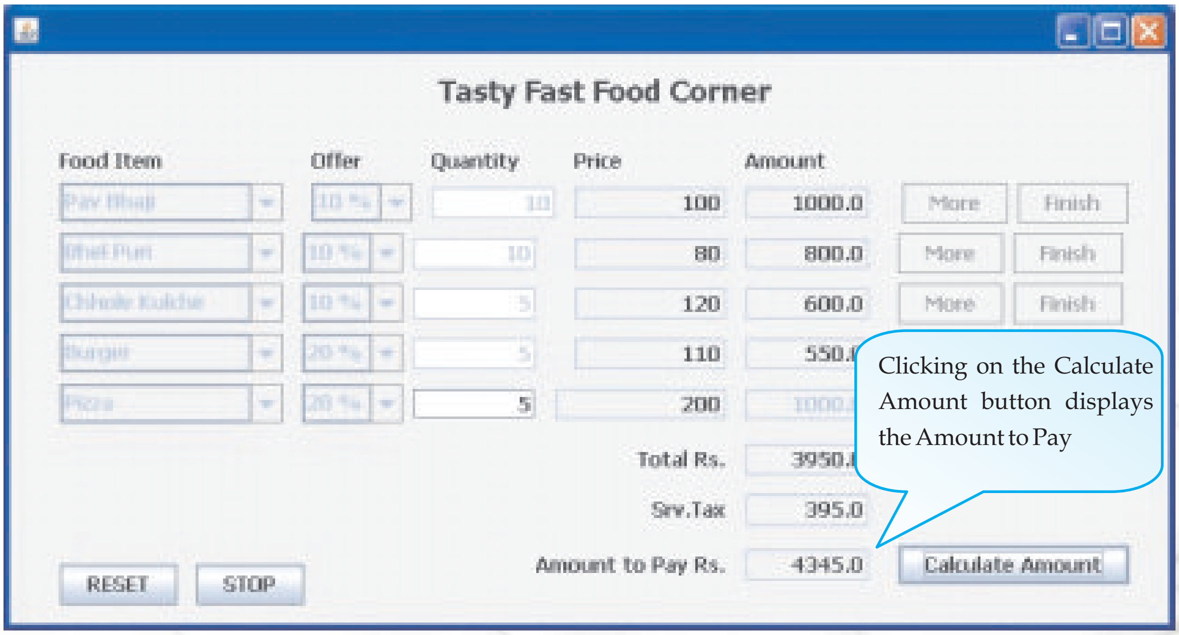
Total Rs. 3950.0

Srv.Tax 395.0

Amount to Pay Rs.

Clicking on the Finish button displays the Total and Service Tax and enables the Calculate Amount button

Figure 11.24 Effect of Clicking on the Finish Button



Tasty Fast Food Corner

Food Item	Offer	Quantity	Price	Amount		
Pav Bhaji	10 %	10	100	1000.0	More	Finish
Bhel Puri	10 %	10	80	800.0	More	Finish
Chhole Kulche	10 %	5	120	600.0	More	Finish
Burger	20 %	5	110	550.0		
Pizza	20 %	5	200	1000.0		

Total Rs. 3950.0

Srv.Tax 395.0

Amount to Pay Rs. 4345.0

Clicking on the Calculate Amount button displays the Amount to Pay

Figure 11.25 Effect of Clicking on the Calculate Amount Button