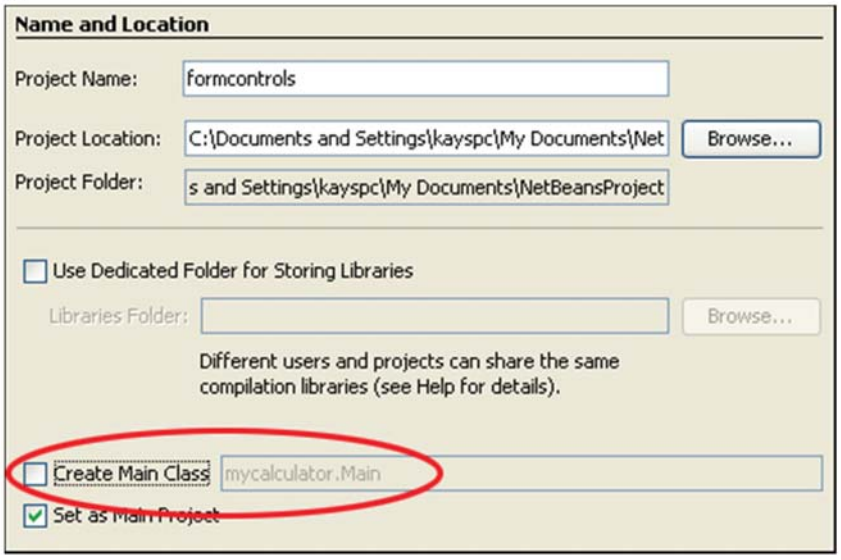
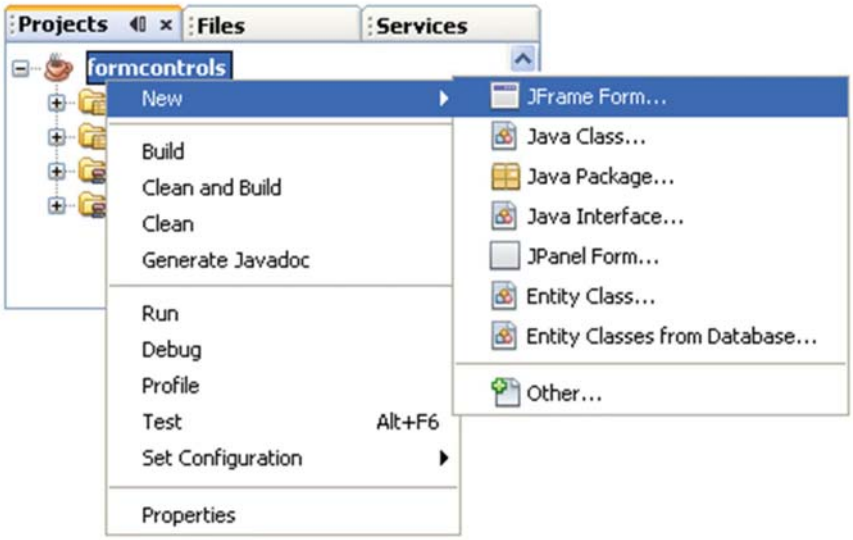
Lecture (05)
Forms II

Dr. Ahmed ElShafee

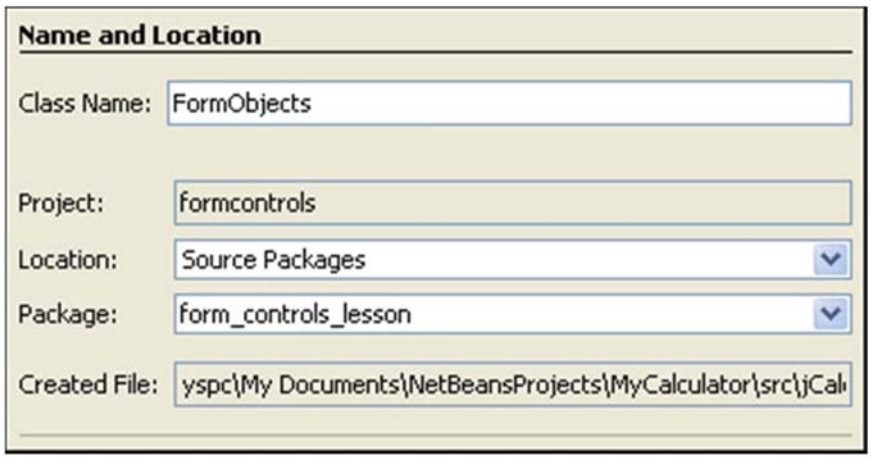
- Create a new project for this (**Java > Application).**
- Call the project **formcontrols,** and uncheck the "Create main class" box:



- Now add a form by right-clicking the project name in Projects window and selecting New > Jframe Form:



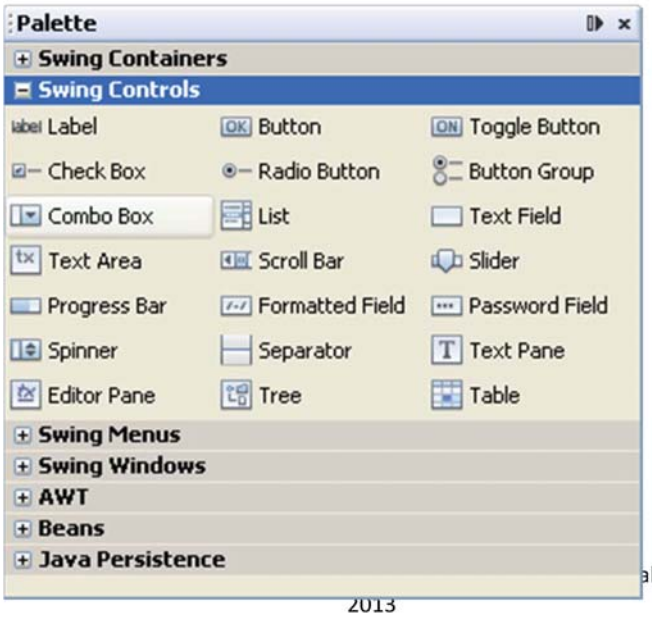
- When the dialogue box appears, enter **FormObjects** as the Class name, and **form_controls_lesson** as the package name:



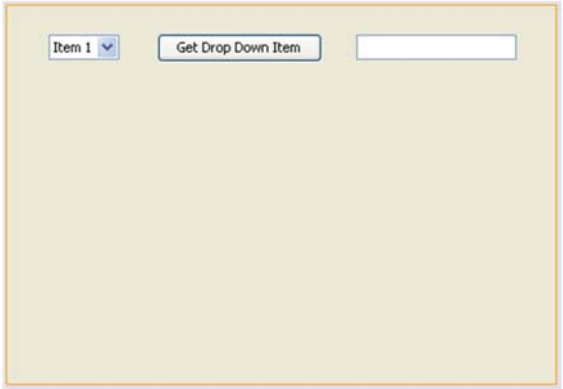
- You will then have a Class called **FormObjects,** which is in the package called **form_controls_lesson,** which in the **formcontrols** project.

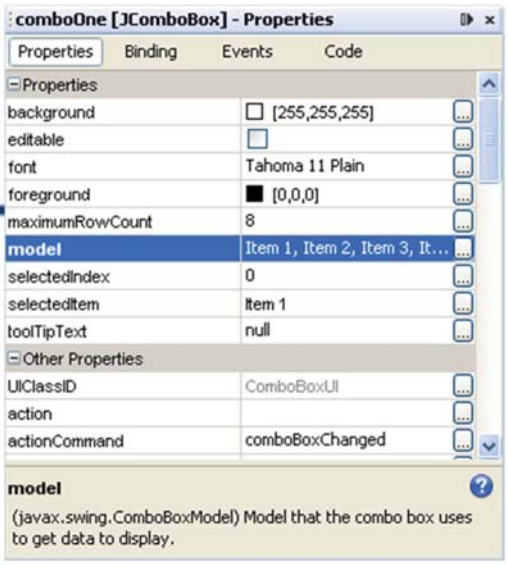
The JComboBox Control

- A combo box is a drop down list of items that can be selected by a user.
- It can be found in the NetBeans palette, under Swing Controls:

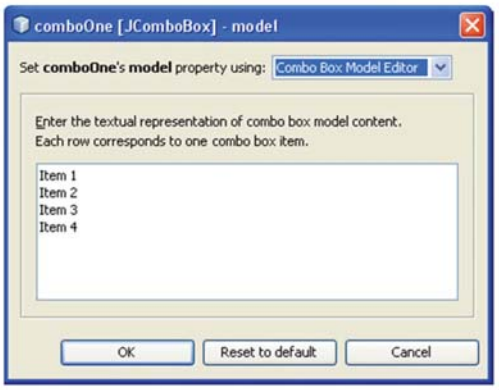


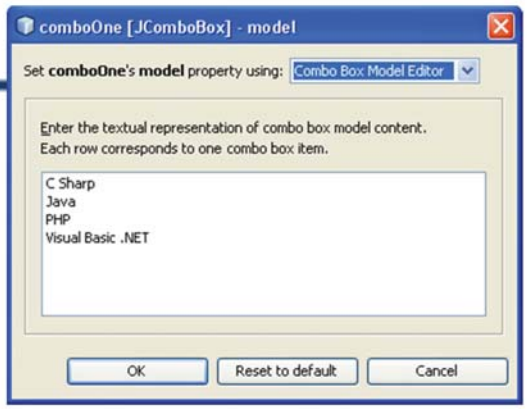
- Add another button and text field
- Change the name of the button in the same way, rename it btnComboBox.
- Change the text on the button to **Get Drop Down Item.**
- Change the name of the Text Field to **txtComboBoxItem.**
- Delete the default text and leave it blank.
- Your form should then look something like this one:



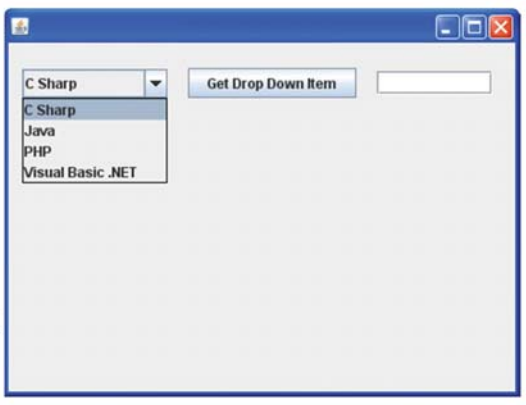


- The default items in the Combo Box are Item 1, Item 2, etc. We'll add our own items.
- Click back onto your Combo Box to select it.
- Now look at the properties window on the right of NetBeans.
- Locate the **model property:**
- Click the small button to the right of the model row, the one with the three dots in it.
- The following dialogue box appears:





- You can highlight the items in the white text area, and delete them. Replace them with the following items: C Sharp, Java, PHP, Visual Basic .NET.
- Your dialogue box will then look like this:

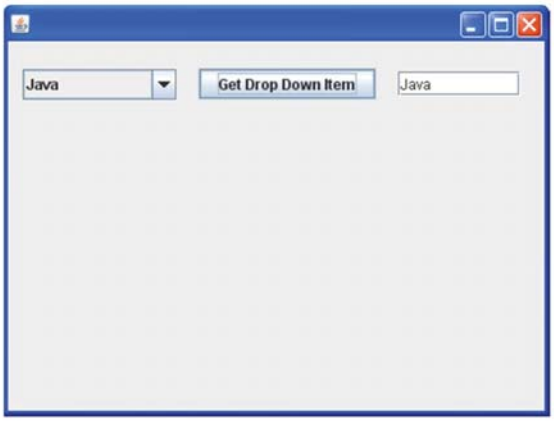


- Click OK when you've made the changes. Your combo box will now be filled with your own items.
- Run your program and test it out: (Just click OK when it asks you to choose the Main Class.)

- When the button is clicked, we want the item chosen to appear in the text field.
- So double-click your button to create a code stub.
- Te get selected text in combo box use **comboOne.getSelectedItem( );**
- But it returned as object to get text from it use casting **(String)comboOne.getSelectedItem( );**

```
private void btnComboBoxActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    String itemText = (String)comboOne.getSelectedItem( );
    txtComboBoxItem.setText( itemText );
}
```

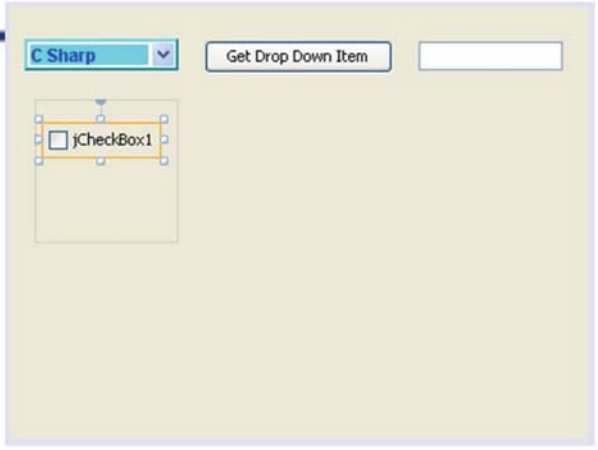

- Run your programme again and try it out.
- Select an item from your drop down list.
- Then click your button.
- The item you selected should appear in the text field:

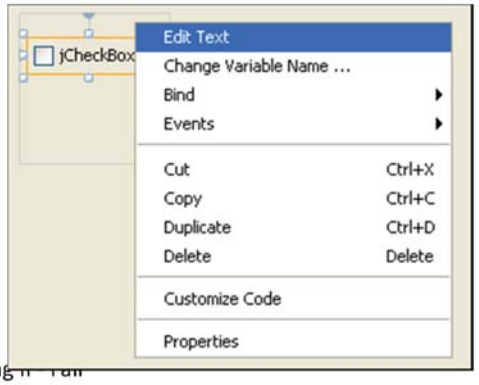


- Click on your combo box to select it.
- Now have a look at the properties window again.
- Try setting the following:
- **Background Colour**
- **Foreground Colour**
- **Font**
- **Border**
- You'll need to play around with them for a while.
- For the colours, RGB seems to work best.

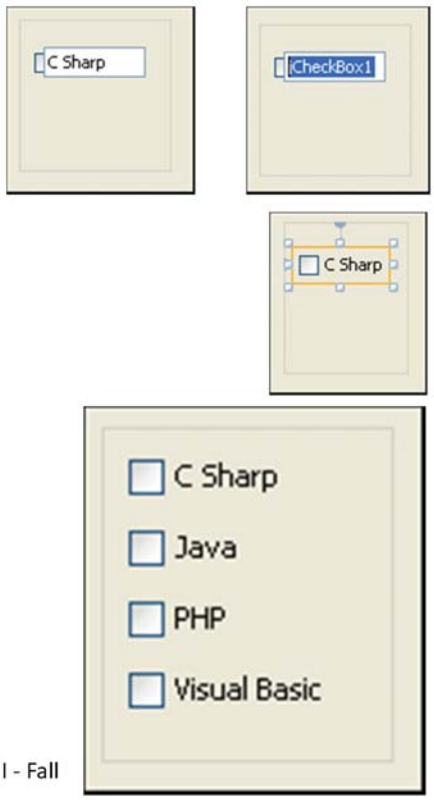
Check Boxes

- A check box is a way to allow your users to select and deselect items.
- add a panel to your form,
- Drag a check box onto your panel.
- The text jCheckBox1 is the default text
- You can change this either in the properties window, or by right-clicking the check box.

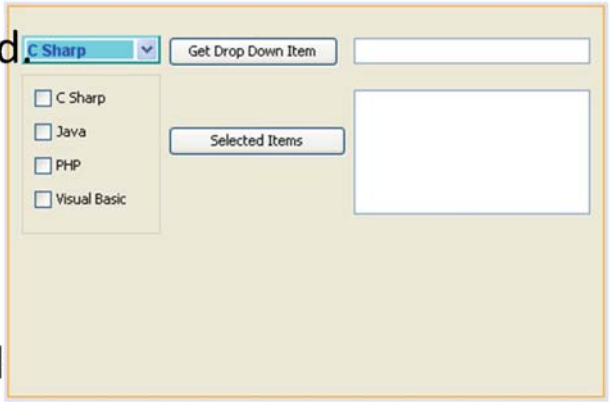




- When you click on Edit Text, the default text will be highlighted:
- Type **C Sharp** over the top of the highlighted text:
- Press the enter key on your keyboard to confirm the change. The text will change for your check box:
- Now that you have added one check box to your panel, add three more. Change the text of the three to: Java, PHP, and Visual Basic.
- Your check boxes will then look like this:



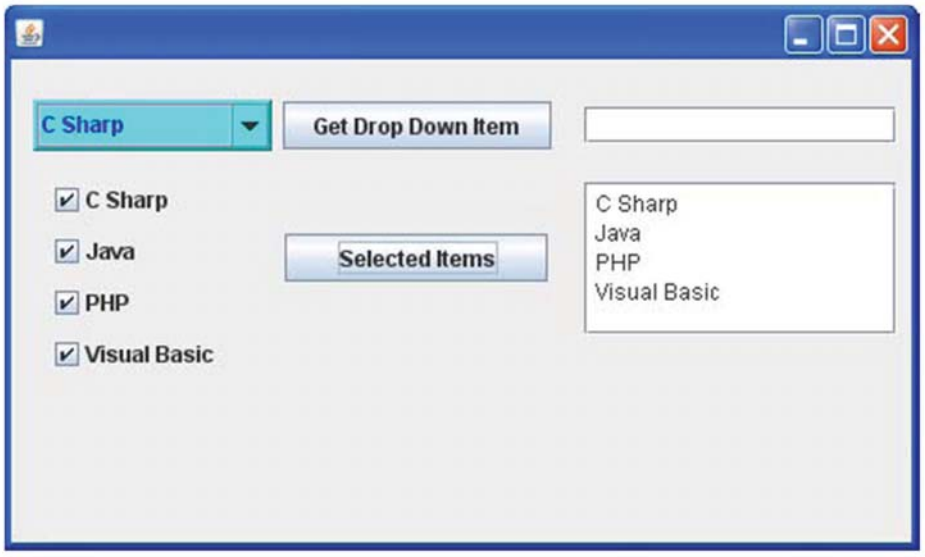
- What we'll do is to get the items that a user has checked.
- We'll do this when a button is clicked.
- To display the items, we'll use the Text Area control, rather than a text field.
- Add button change its name to **btnCheckBoxes** and text to **Selected Items.**
- Add Text Area control and change its name to **taOne**



- Check boxes have a property called isSelected.
- We can use this in a series of IF Statements to see if each box is selected or not.
- If they are, we can build up a string, adding the text from each check box.

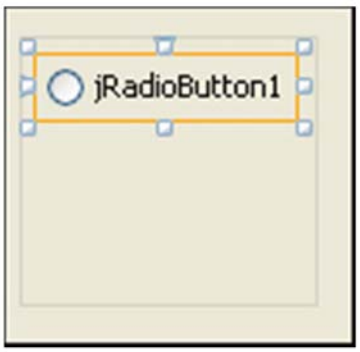
```
String s1 = "";
if (jCheckBox1.isSelected()){
    s1 = s1 + " " + jCheckBox1.getText() + '\n';
}

if (jCheckBox2.isSelected()){
    s1 =  s1 + " " +  jCheckBox2.getText() + '\n';
}

if (jCheckBox3.isSelected()){
    s1 =  s1 + " " +  jCheckBox3.getText() + '\n';
}

if (jCheckBox4.isSelected()){
    s1 =  s1 + " " +  jCheckBox4.getText() + '\n';
}

taOne.setText(s1);
```


- run

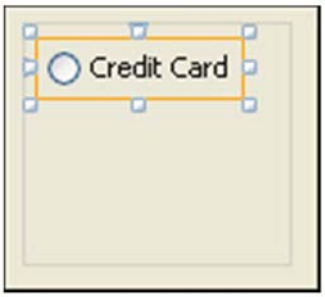


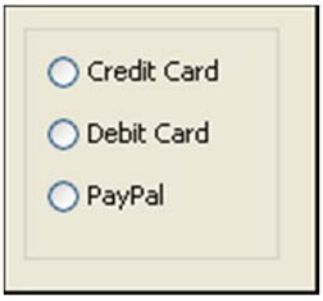
Radio Buttons

- Radio buttons are usually used to select just one item from a list, rather than the multiple items available with check boxes. Let's see how they work.
- Drag and drop a panel onto your form.
- Then locate the Radio Button control in it
- The default text for the first radio button is **jRadioButton1.**



- **We'll use our radio** buttons to allow a user to select a payment method.
- So change the text of your radio button to **Credit Card.**
- (The text can be changed in the same way as you did for check boxes)
- Add two more radio buttons to the panel. Change the text to
- **Debit Card, and**
- **PayPal:**





- problem with the radio buttons you've just added.
- To see what the problem is, run your programme again.
- Now select one of the radio buttons.
- Try selecting another radio button and you'll find that you can indeed select more than one at the same time:
- To solve the problem, Java lets you to create something called a **ButtonGroup.**

- **As its name suggest, this allows you to group buttons under one** name.
- You can then add radio buttons to the group.
- Once you've added buttons to the group, only one option is available for selection.
- To see how the ButtonGroup works, add the following method to your code, somewhere near the top:
- We can call the **groupButton** method from the constructor

```
package form_controls_lesson;
import javax.swing.ButtonGroup;
/**
 *
 * @author Dr. Ahmed Elshafee
 */
public class FormObjects extends jav

    /** Creates new form FormObjects
    public FormObjects() {
        initComponents();
    }

    @SuppressWarnings("unchecked")
    Generated Code
private void groupButton( ) {
ButtonGroup bg1 = new ButtonGroup( );
bg1.add(jRadioButton1);
bg1.add(jRadioButton2);
bg1.add(jRadioButton3);
}
    private void btnComboBoxActionPerformed(ja
        // TODO add your handling code here:
        String itemText = (String)comboOne.get
        txtComboBoxItem.setText( itemText );


    /** Creates new form FormObjects */
    public FormObjects() {
        initComponents();
        groupButton();
    }
```

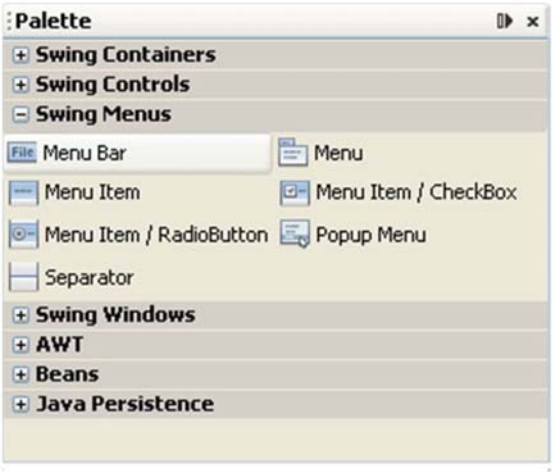
- To get at which radio button was selected, again there's an **isSelected** method we can use.
- Add a normal button to your form, Change the variable name of your button to **btnRadios.** Change the text property to **Payment Option.**
- We can have a message box to display which payment option was selected.

```
    private void btnRadiosActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        String radioText="";
        if(jRadioButton1.isSelected())
            radioText=radioText+jRadioButton1.getText();
        if(jRadioButton2.isSelected())
            radioText=radioText+jRadioButton2.getText();
        if(jRadioButton3.isSelected())
            radioText=radioText+jRadioButton3.getText();
        javax.swing.JOptionPane.showMessageDialog( FormObjects.this, radioText );

    }
```

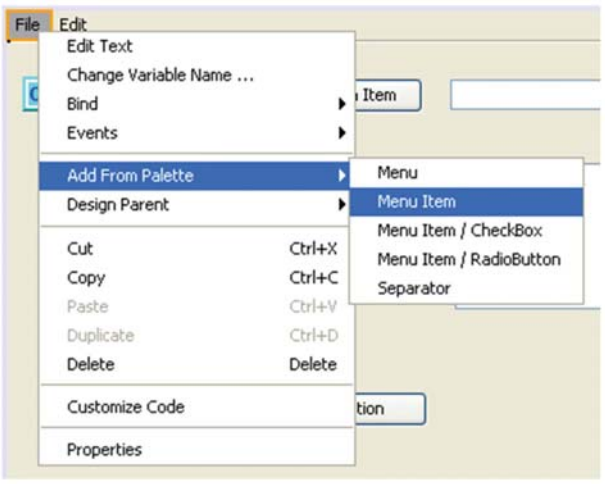

- Because we were using a console, the first item was **null.**
- **Now** we have: **FormObjects.this**
- The first item between the round brackets is for the window in which you want to display the message box.
- Null meant no window.
- FormObjects.this means **this** component (the form) of the **FormObjects class.**

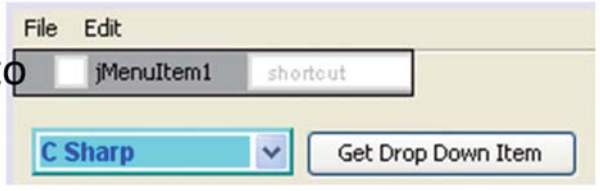
Java Menus

- You can add menus to your Java forms, things like File, Edit, View, etc.
- Each menu has menu items, and these in turn can have sub menus.
- Return to Design view. In the NetBeans palette, locate the Menu Bar item:
- Drag one to the top of your form. When you let the mouse button go, you'll have a default File and Edit menu bar:



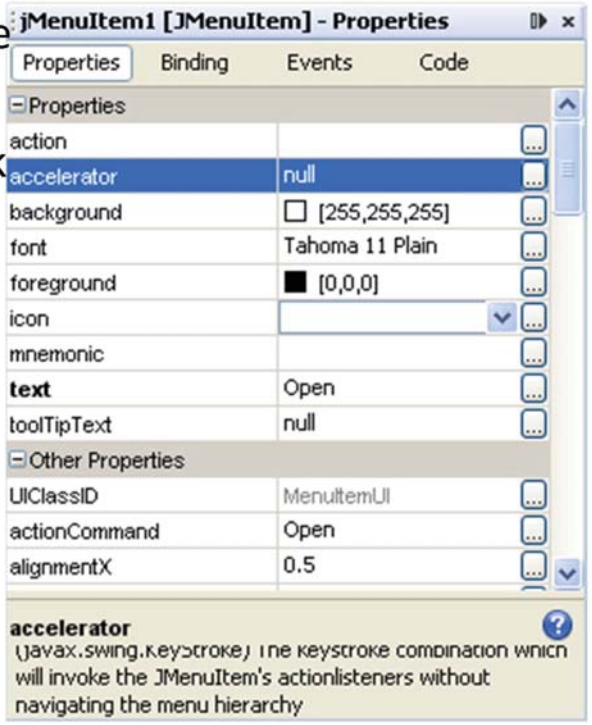
- To add your own, click on the **File menu** item to select it.
- With the File menu item selected, right-click.
- A new menu will appear.
- Select **Add From Palette > Menu Item:**
- A Menu Item will be added to your File menu:
- What we'll do is to add menu items to open and save a file.



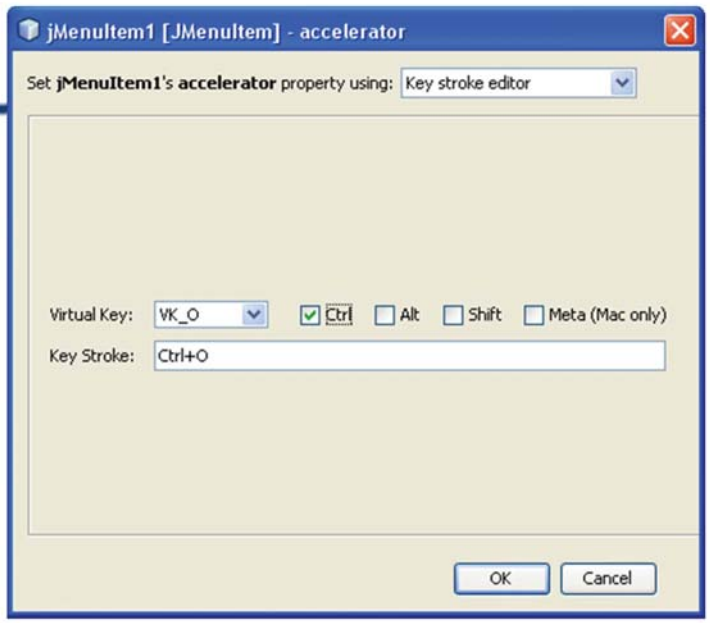


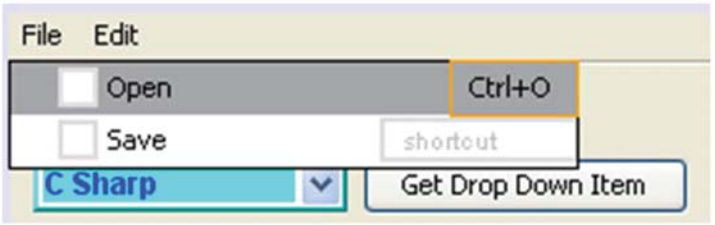
- Double click on the default text **jMenuItem1.**
- It will then be highlighted, so that you can type over it:
- Type Open, then press the enter key on your keyboard:
- Add another menu item in the way.
- This time, type Save as the menu item:
- You can add shortcuts for your menu items.
- Click on to the Open menu item, then onto the shortcut for it:

- With the shortcut item selected, have a look at the properties window
- Locate the Accelerator item, and click the small button to the right of the row.
- A dialogue box appears.
- You can set which shortcut keys you want for a menu item from this dialogue box.



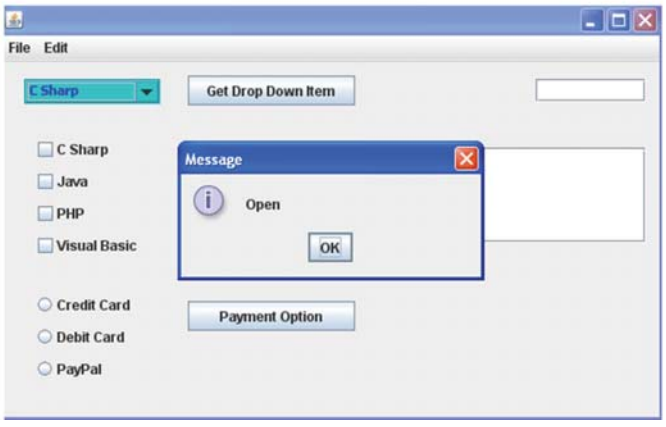
- An open shortcut is usually CTRL + O.
- Type an O in the box, and Shift + O will appear.
- Uncheck the Shift item and check Ctrl instead:
- Click OK, and the shortcut will be added to your menu item:





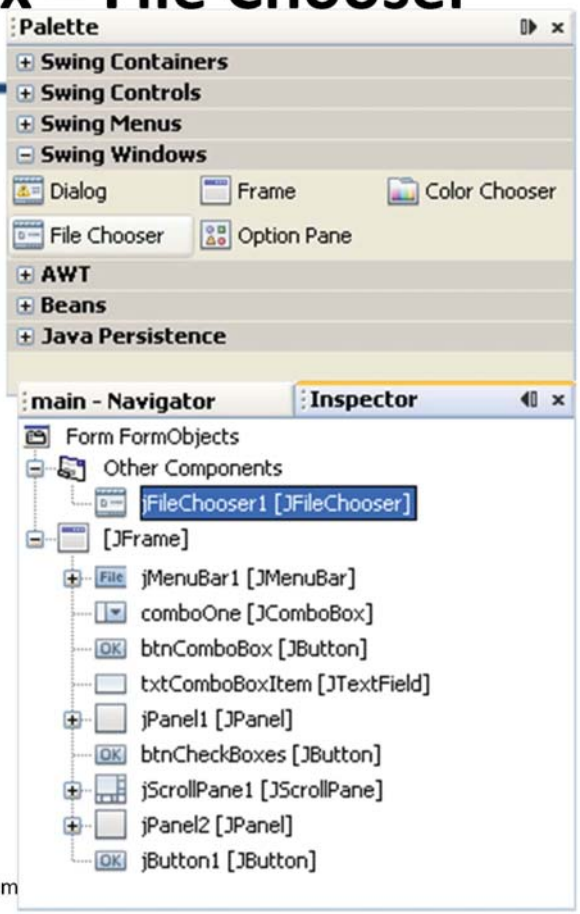
- Now right click From the menu that appears, select **Events > Action > Action Performed.**
- This will create a code stub for the menu item.
- Enter the follow**ing** for the code:

```
private void jMenuItem1ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    javax.swing.JOptionPane.showMessageDialog( FormObjects.this, "Open" );
}
```

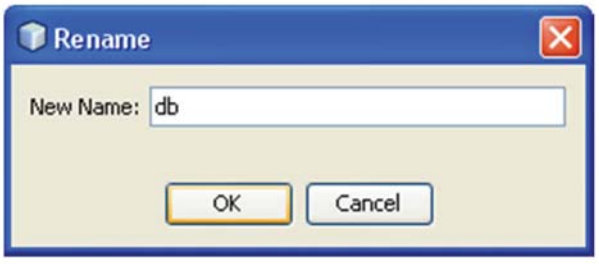


The Open File Dialogue Box – File Chooser

- In the NetBeans palette, locate the **File Chooser control,**
- which is under Swing Windows:
- Drag a File Chooser near your form, but not onto it.



- The default name for the File Chooser is **jFileChooser1.**
- Right click on **jFileChooser1** in the Inspector window.
- From the menu that appears, select Change Variable Name. When the dialogue box appears, type **db** as the name:

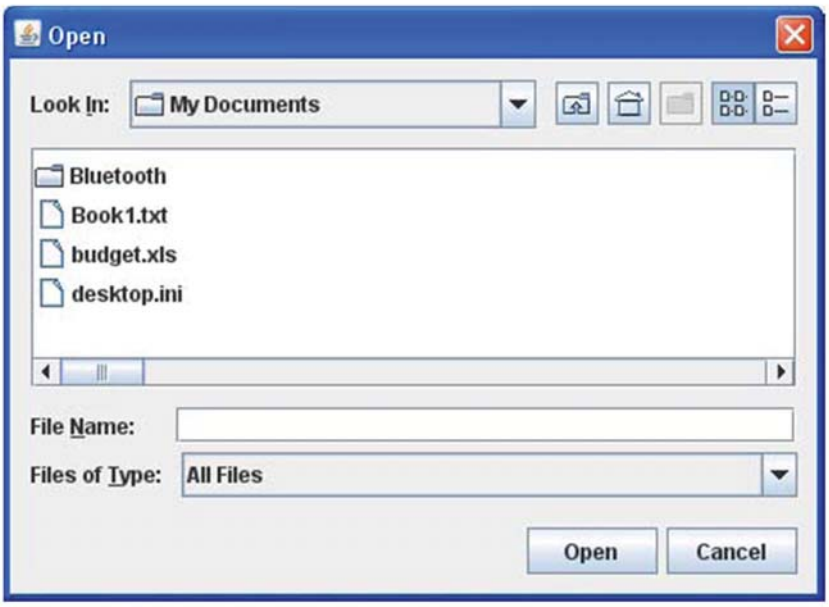


- To display JFileChooser1,

```
private void jMenuItem1ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    //javax.swing.JOptionPane.showMessageDialog( FormObjects.this, "Open" );
    int returnVal = db.showOpenDialog( this );

}
```

- **ShowOpenDialog method of the** File Chooser class.
- In between the round brackets of **ShowOpenDialog** you type the name of the window that's going to be holding the dialogue box.
- We've typed **this, meaning "this form".**
- The **ShowOpenDialog** method returns a value.
- This value is an integer**.**
- The value tells you which button was clicked on the dialogue box: open, cancel, etc.
- We're storing the value in a variable called **returnVal.**

- Run



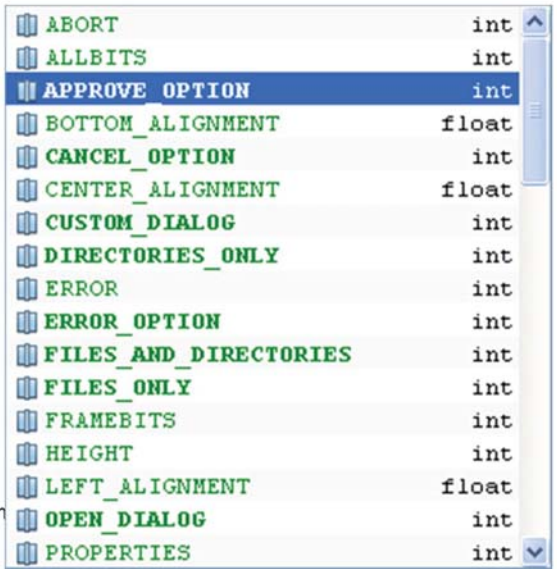
- If you want to open a file, you have to write the code yourself.

- First, add the following IF Statement to your code, just below your other two lines (the returnVal one and your commented-out message box):

if (returnVal = = javax.swing.JFileChooser.APPROVE_OPTION) {

}

- So we're using a Swing class called JFileChooser.
- With this class, you can examine which button was clicked. When you type the dot after JFileChooser you should see a popup list:



- The APPROVE_OPTION means things like the OK and Yes buttons
- So we're testing the returnVal variable to see if it matches the APPROVE_OPTION (did the user click OK?).
- To get at the file chosen by the user there is a method called **getSelectedFile.**
- Creating object file from selected file

java.io.File file = db.getSelectedFile( );

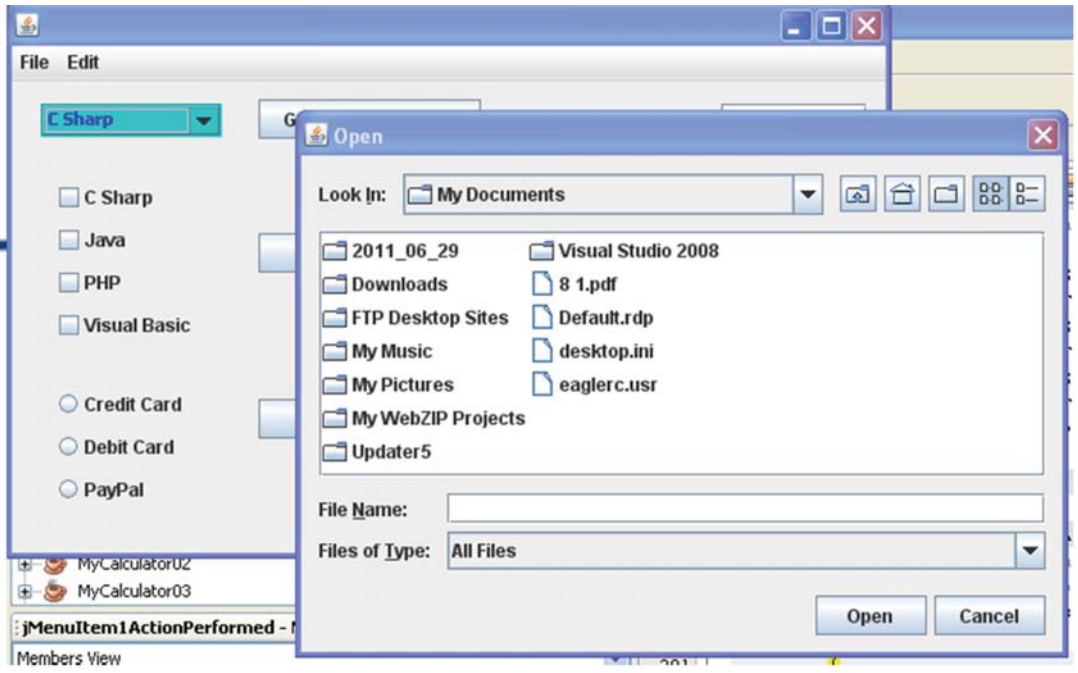
- To get file name as string

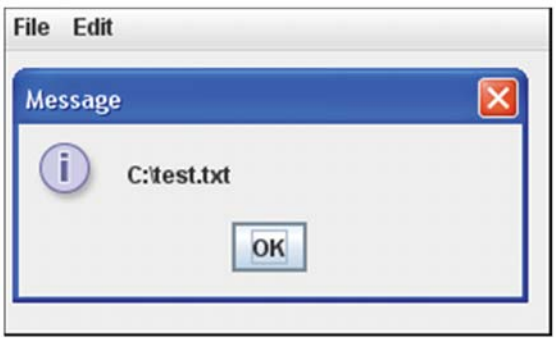
String file_name = file.toString( );

- Whole code

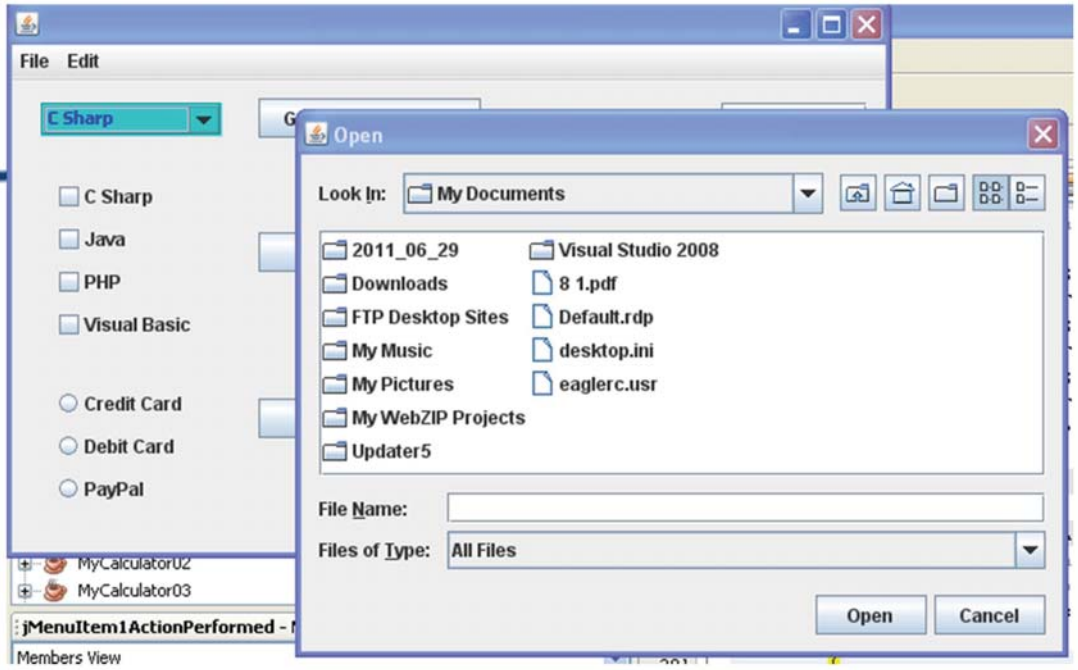
```
private void jMenuItem1ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    //javax.swing.JOptionPane.showMessageDialog( FormObjects.this, "Open" );
    int returnVal = db.showOpenDialog( this );
    if(returnVal==javax.swing.JFileChooser.APPROVE_OPTION)
    {
        java.io.File file = db.getSelectedFile( );
        String file_name = file.toString( );
        javax.swing.JOptionPane.showMessageDialog(FormObjects.this, file_name);
    }

}
```


- Run
- Open file
- Select file press open





- Run
- Open
- cancel



- **Files of Type** on your Open file dialogue box is set to **"All files".**
- You can filter the files on this list, so that the user can open only, say, text files, or just images with certain extensions (jpeg, gif, png).
- Import following

import javax.swing.filechooser.FileFilter;

import javax.swing.filechooser.FileNameExtensionFilter;

```
package form_controls_lesson;
import javax.swing.ButtonGroup;
import javax.swing.filechooser.FileFilter;
import javax.swing.filechooser.FileNameExtensionFilter;
/**
 *
 * @author Dr. Ahmed Elshafee
 */
```

- create a new **FileFilter object.**
- Add the following line just before the first line of your code (before the **int returnVal line):**

FileFilter ft = new FileNameExtensionFilter("Text Files", "txt");

- Arguments, the **name of files** that you want to display, and **extension.**
- You can add more than one extension. Just type a comma and then the files you want to display:

FileNameExtensionFilter("Text Files", "txt", “html”);

- Then add filter

db.addChoosableFileFilter(ft);

- Code

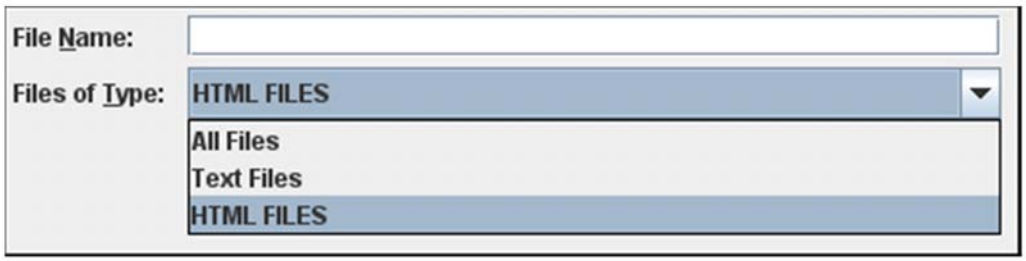
```
private void jMenuItem1ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    //javax.swing.JOptionPane.showMessageDialog( FormObjects.this, "Open" );
    FileFilter ft = new FileNameExtensionFilter("Text Files", "txt");
    db.addChoosableFileFilter( ft );
    int returnVal = db.showOpenDialog( this );
    if(returnVal==javax.swing.JFileChooser.APPROVE_OPTION)
    {
        java.io.File file = db.getSelectedFile( );
        String file_name = file.toString( );
        javax.swing.JOptionPane.showMessageDialog(FormObjects.this, file_name);
    }

}
```

- Run your program again, and have a look at your dialogue box. Click the arrow on the dropdown list:

- If you want another line on the list, (to display html files, for example), you can set up another FileFilter object:

```
FileFilter ft = new FileNameExtensionFilter("Text Files", "txt");
FileFilter ft2 = new FileNameExtensionFilter("HTML FILES", "html");

db.addChoosableFileFilter(ft);
db.addChoosableFileFilter(ft2);
```

- run



Dealing with files (text)

```
private void jMenuItem2ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    FileFilter ft = new FileNameExtensionFilter("Text Files", "txt");
    db.addChoosableFileFilter( ft );
    int returnVal = db.showSaveDialog( this );
    if(returnVal==javax.swing.JFileChooser.APPROVE_OPTION)
    {
        try
        {
         FileWriter fw = new FileWriter(db.getSelectedFile( ), false);
        PrintWriter pw = new PrintWriter(fw);
        pw.println(comboOne.getSelectedIndex());
        pw.println(jCheckBox1.isSelected());
        pw.println(jCheckBox2.isSelected());
        pw.println(jCheckBox3.isSelected());
        pw.println(jCheckBox4.isSelected());
        pw.println(jRadioButton1.isSelected());
        pw.println(jRadioButton2.isSelected());
        pw.println(jRadioButton3.isSelected());
        pw.close();
        fw.close();
        }catch (IOException e) {}

    }
}
```

```
private void jMenuItem1ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    //javax.swing.JOptionPane.showMessageDialog( FormObjects.this, "Open" );
    FileFilter ft = new FileNameExtensionFilter("Date Files", "txt");
    db.addChoosableFileFilter( ft );
    int returnVal = db.showOpenDialog( this );
    if(returnVal==javax.swing.JFileChooser.APPROVE_OPTION)
    {
        try
        {
        FileReader fr = new FileReader(db.getSelectedFile( ));
        BufferedReader br = new BufferedReader(fr);

        comboOne.setSelectedIndex(Integer.parseInt(br.readLine()));
        jCheckBox1.setSelected(Boolean.parseBoolean(br.readLine()));
        jCheckBox2.setSelected(Boolean.parseBoolean(br.readLine()));
        jCheckBox3.setSelected(Boolean.parseBoolean(br.readLine()));
        jCheckBox4.setSelected(Boolean.parseBoolean(br.readLine()));
        jRadioButton1.setSelected(Boolean.parseBoolean(br.readLine()));
        jRadioButton2.setSelected(Boolean.parseBoolean(br.readLine()));
        jRadioButton3.setSelected(Boolean.parseBoolean(br.readLine()));
        btnComboBoxActionPerformed(null);
        btnCheckBoxesActionPerformed(null);
        br.close();
        fr.close();
        }catch (IOException e) {}
    }
```

Dealing with files (data)

```
private void jMenuItem2ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    FileFilter ft = new FileNameExtensionFilter("Text Files", "txt");
    db.addChoosableFileFilter( ft );
    int returnVal = db.showSaveDialog( this );
    if(returnVal==javax.swing.JFileChooser.APPROVE_OPTION)
    {
        try
        {
        java.io.FileOutputStream f0 = new java.io.FileOutputStream(db.getSelectedFile( ));
        DataOutputStream d0=new DataOutputStream(f0);
        d0.writeInt(comboOne.getSelectedIndex());
        d0.writeBoolean(jCheckBox1.isSelected());
        d0.writeBoolean(jCheckBox2.isSelected());
        d0.writeBoolean(jCheckBox3.isSelected());
        d0.writeBoolean(jCheckBox4.isSelected());
        d0.writeBoolean(jRadioButton1.isSelected());
        d0.writeBoolean(jRadioButton2.isSelected());
        d0.writeBoolean(jRadioButton3.isSelected());
        d0.close();
        f0.close();
        }catch (IOException e) {}

    }
}
```

```
private void jMenuItem1ActionPerformed(java.awt.event.ActionEvent evt) {
    FileFilter ft = new FileNameExtensionFilter("Date Files", "txt");
    db.addChoosableFileFilter( ft );
    int returnVal = db.showOpenDialog( this );
    if(returnVal==javax.swing.JFileChooser.APPROVE_OPTION)
    {
        try
        {
        FileInputStream f0 = new java.io.FileInputStream(db.getSelectedFile( ));
        DataInputStream d0=new DataInputStream(f0);
        comboOne.setSelectedIndex(d0.readInt());
        jCheckBox1.setSelected(d0.readBoolean());
        jCheckBox2.setSelected(d0.readBoolean());
        jCheckBox3.setSelected(d0.readBoolean());
        jCheckBox4.setSelected(d0.readBoolean());
        jRadioButton1.setSelected(d0.readBoolean());
        jRadioButton2.setSelected(d0.readBoolean());
        jRadioButton3.setSelected(d0.readBoolean());
        btnComboBoxActionPerformed(null);
        btnCheckBoxesActionPerformed(null);
        d0.close();
        f0.close();
        }catch (IOException e) {}
    }

}
```

Thanks,

See you next Lecture, isA