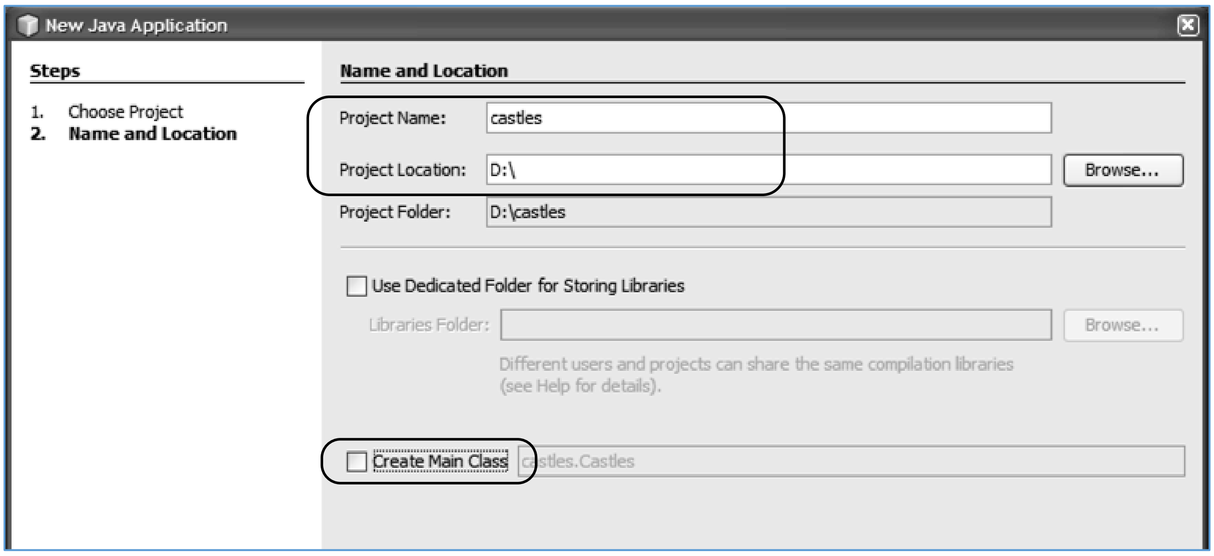
3 Multiple windows

Many larger computer applications use more than one form window, perhaps inputting data in one window and outputting results in another. In this chapter we will examine how multiple windows can be linked together in a project, and data transferred from one window to another.

The first application we will develop provides information about castles in Wales. This consists of an index page, linked to pages to display picture images of different castles. We will begin by setting up the index page.

Close all projects, then select the ***New Project*** option from the NetBeans ***File*** menu. Check that ***Java / Java Application*** is highlighted on the first screen, then click the ***Next*** button to enter the ***Project Name*** as ***castles***. Use the ***Browse*** button to choose the storage location, and ensure that ***Create Main Class*** is not selected.


New Java Application

Steps
1. Choose Project
2. Name and Location

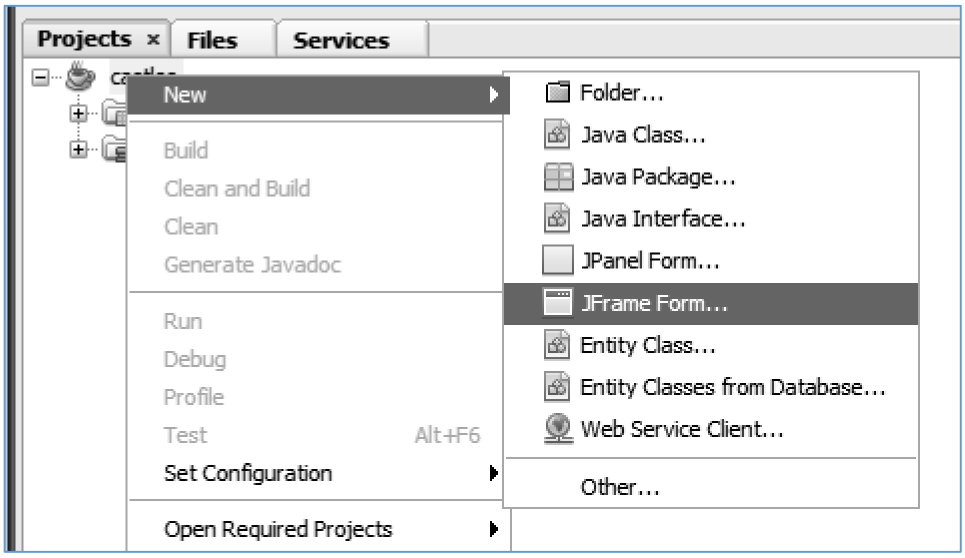
Name and Location

Project Name: castles
Project Location: D:\
Project Folder: D:\castles

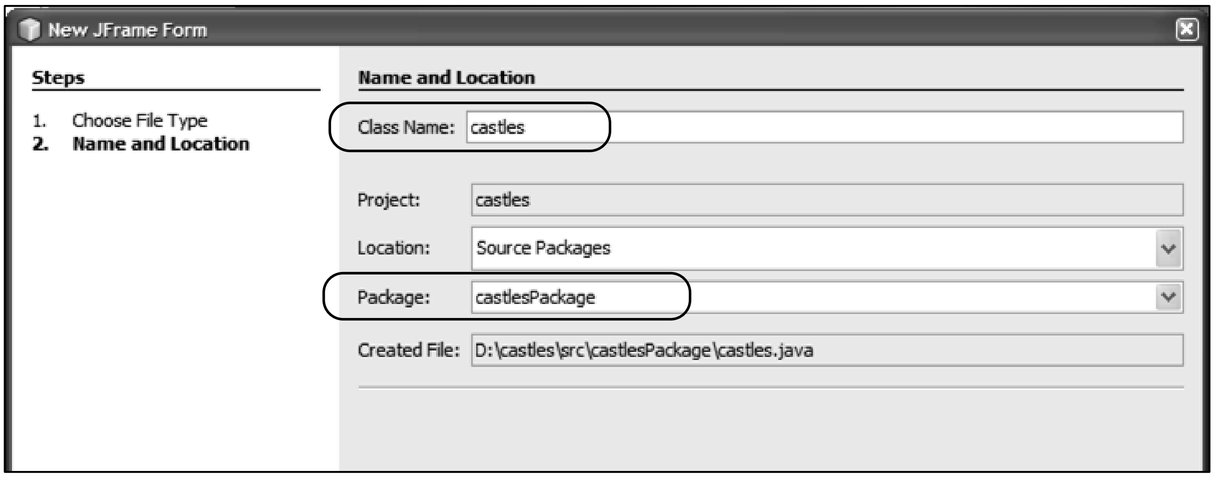
Use Dedicated Folder for Storing Libraries
Libraries Folder:
Different users and projects can share the same compilation libraries (see Help for details).

Create Main Class castles.Castles

Click the ***Finish*** button to return to the NetBeans editing screen. Right-click the ***castles*** project icon and select ***New / JFrame Form***:



Enter the ***Class Name*** as ***castles***, and the ***Package*** as ***castlesPackage***:



Class Name: castles

Project: castles

Location: Source Packages

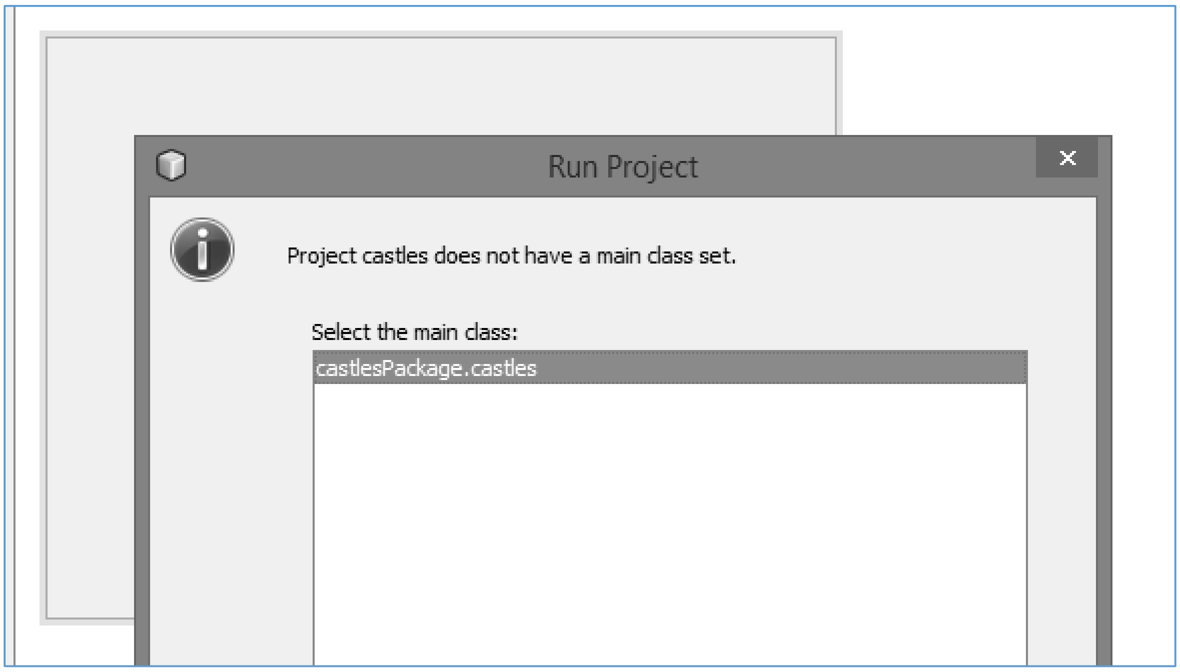
Package: castlesPackage

Created File: D:\castles\src\castlesPackage\castles.java

Return to the NetBeans editing page. Carry out the steps to set the correct format for the ***form***:

- Go to the ***Code*** tab in the ***Properties*** window and select ***Form Size Policy*** / ***Generate pack()*** / ***Generate Resize Code***.
- Right-click on the form and select ***Set Layout*** / ***Absolute Layout***.
- Click the ***Source*** tab to change to the program code window. Locate the ***main*** method. Open the lines of ***'Look and feel'*** code by clicking the ***+ icon***, then repalce ***"Nimbus"*** by ***"Windows".***

Run the program to check the appearence of the form. Accept the ***main class*** which is offered:



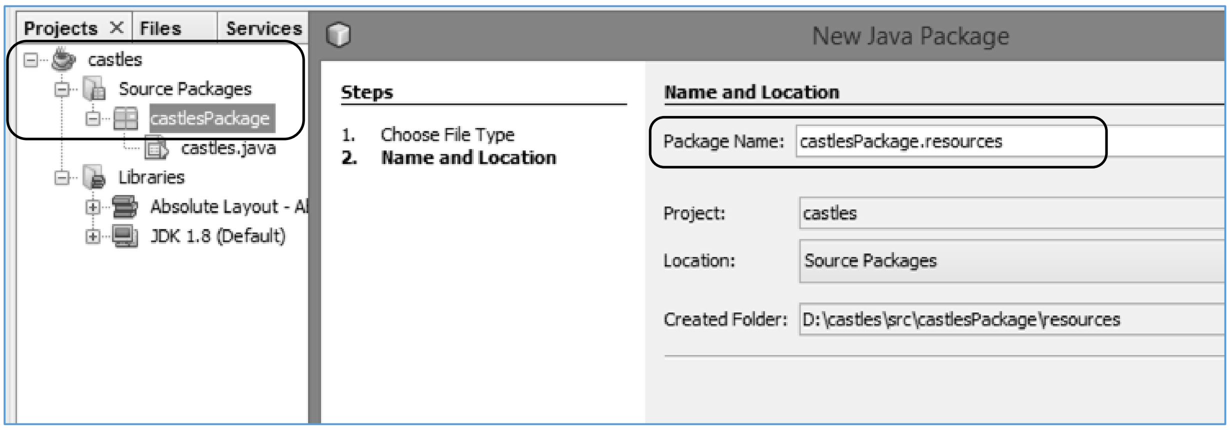
Project castles does not have a main class set.

Select the main class:

castlesPackage.castles

Close the program window and return to the NetBeans editing page.

We will be using picture images in the program, so we should set up a ***resources*** folder to store these. Use the ***+ icons*** to open the ***castles*** project structure until ***castlesPackage*** is reached. Right-click on ***castlesPackage*** and select ***New*** / ***Java Package***. Set the ***Package name*** to ***castlesPackage.resources***:



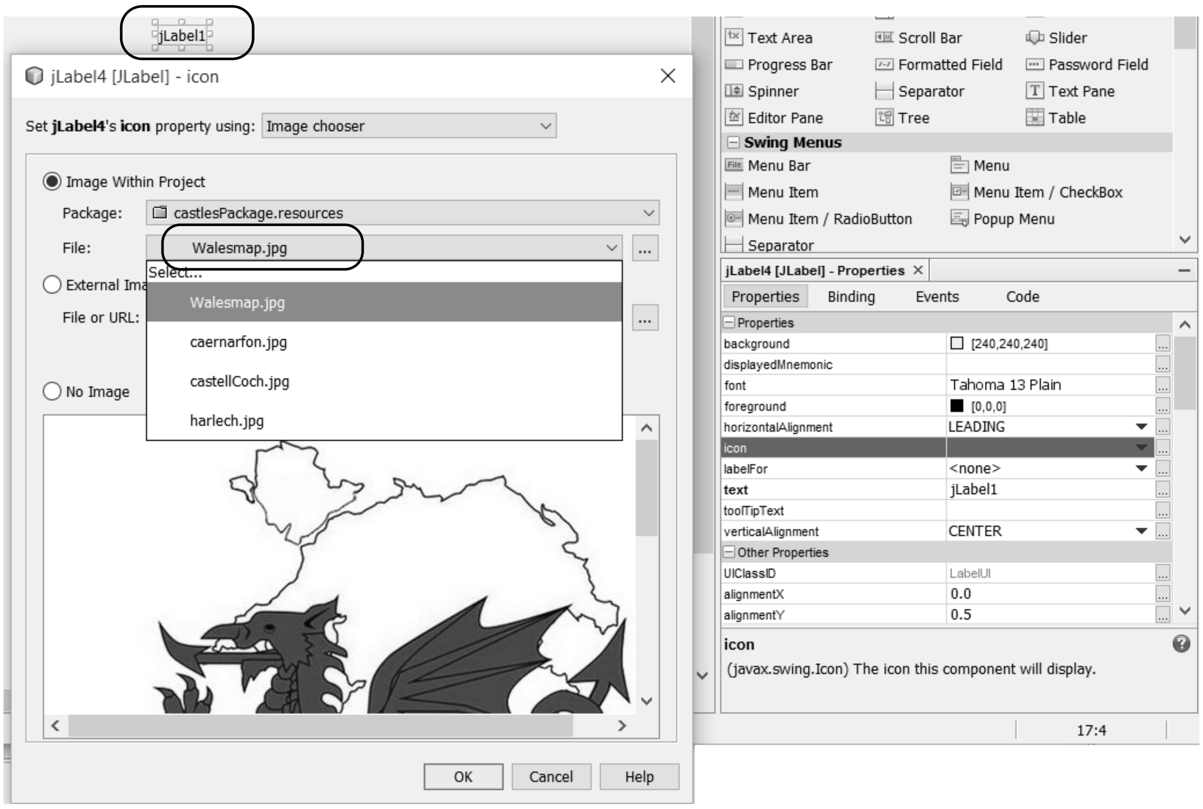
Package Name: castlesPackage.resources
Project: castles
Location: Source Packages
Created Folder: D:\castles\src\castlesPackage\resources

Return to the NetBeans editing page. Click the ***Design*** tab to open the form layout view.

Before continuing, obtain a map of Wales and photograph images of Caernarfon, Harlech and Castell Coch castles. Use an image editing program such as Photoshop or Paint to set the widths of the images to approximately 500 pixels.

Place a ***label*** on the ***form***. Select the ***icon*** property, and click the ellipsis ***(" … ")*** symbol. Use the Import to Project option to upload the map and castle images.

Select the map image as the ***File*** to be displayed.



Set **jLabel4**'s **icon** property using: Image chooser

(●) Image Within Project
Package: castlesPackage.resources
File: Walesmap.jpg
( ) External Ima
File or URL:
( ) No Image

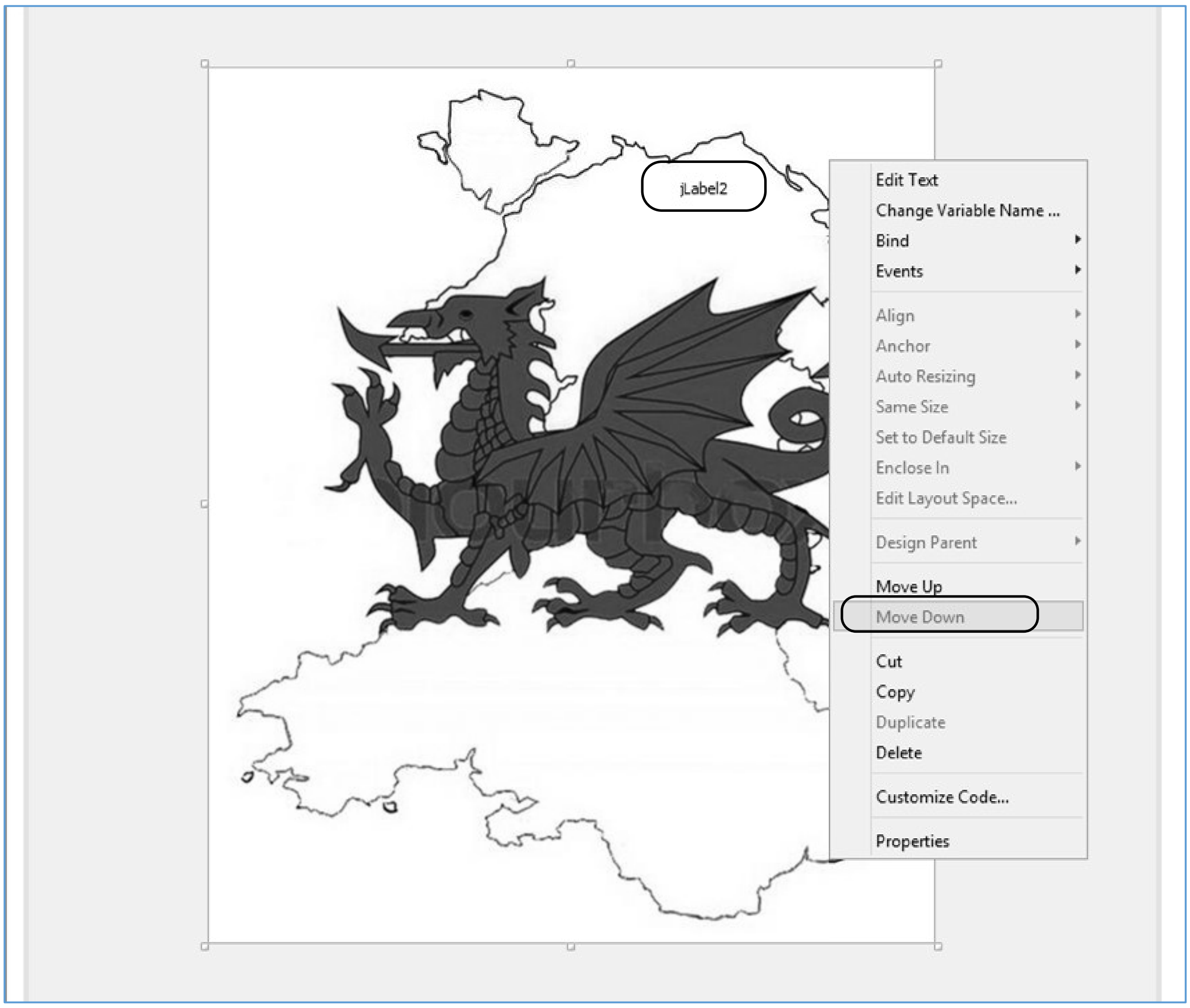
Select...
Walesmap.jpg
caernarfon.jpg
castellCoch.jpg
harlech.jpg

Properties	
background	☐ [240,240,240]
displayedMnemonic	
font	Tahoma 13 Plain
foreground	■ [0,0,0]
horizontalAlignment	LEADING
icon	
labelFor	<none>
text	jLabel1
toolTipText	
verticalAlignment	CENTER
Other Properties	
UIClassID	LabelUI
alignmentX	0.0
alignmentY	0.5

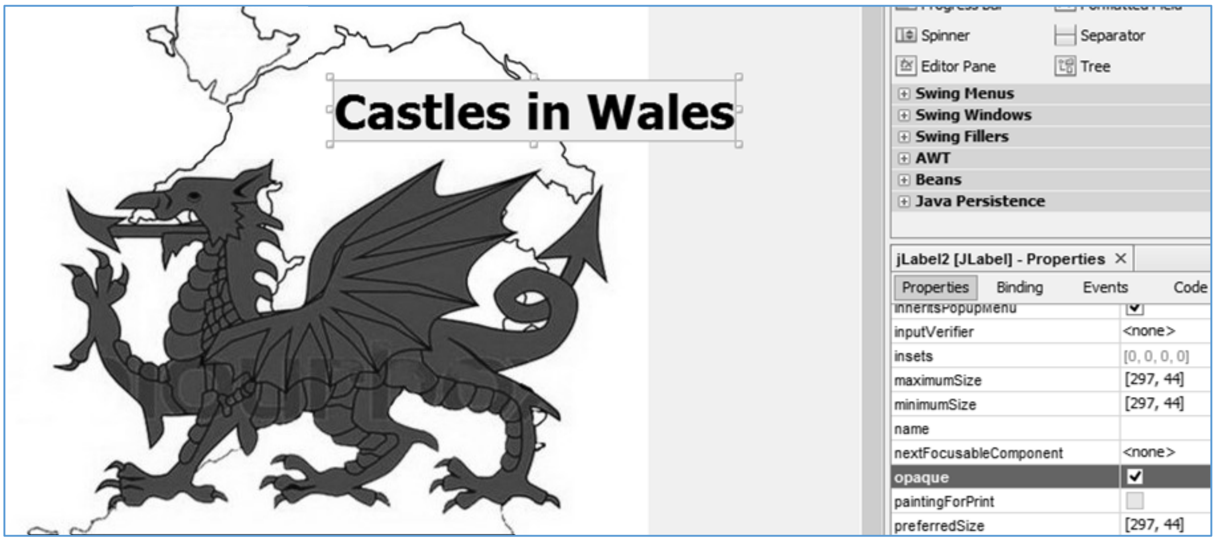
icon
(javax.swing.Icon) The icon this component will display.

Run the program and check that the map image is displayed. Right-click on the map and delete the label text.

Add another ***label*** and place this on the map image. (If the label is not visible, right-click on the map and select ***'Move Down***' to bring the label to the front of the display.)



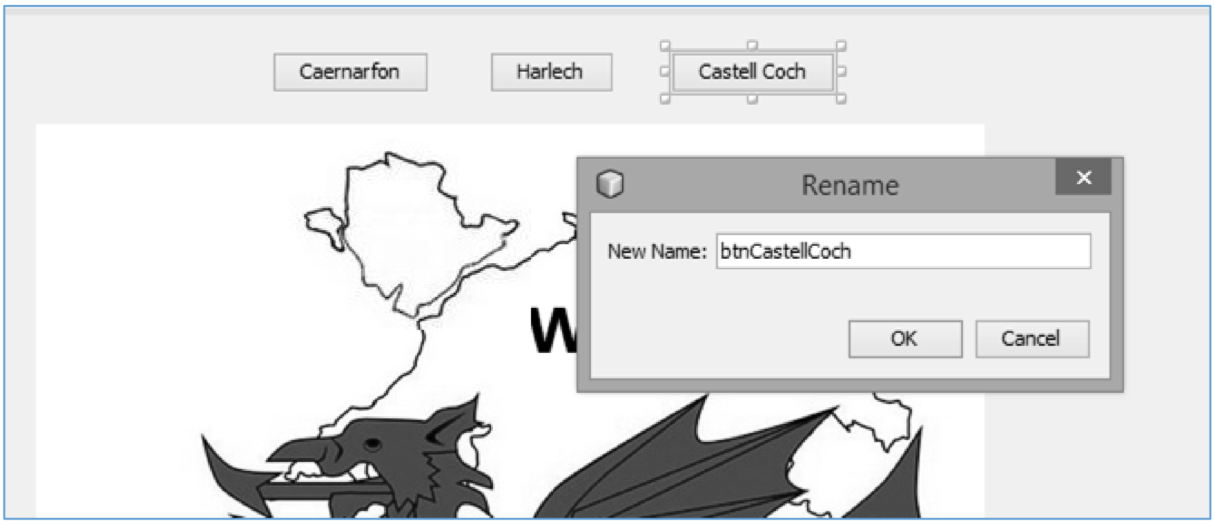
Right-click on the ***label*** and edit the text to "***Castles in Wales***". Increase the font size by means of the ***font*** property, and tick the box for the ***opaque*** property:



inheritsPopupMenu	✓
inputVerifier	<none>
insets	[0, 0, 0, 0]
maximumSize	[297, 44]
minimumSize	[297, 44]
name	
nextFocusableComponent	<none>
opaque	✓
paintingForPrint	
preferredSize	[297, 44]

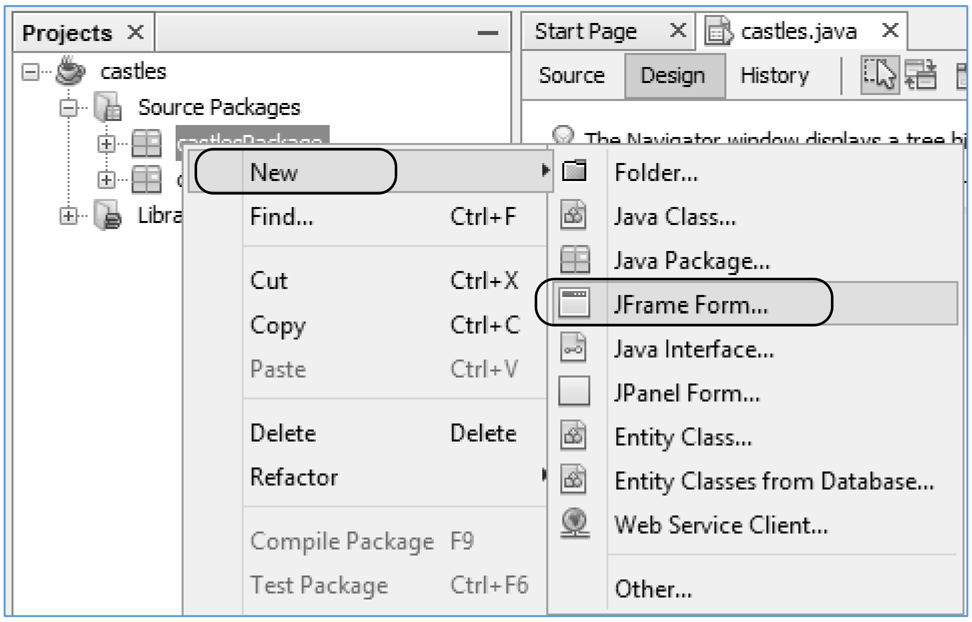
Add buttons to provide links to three pages for different castles. Change the button captions, and give suitable variable names for the buttons, such as:

btnCaernarfon
btnHarlech
btnCastellCoch

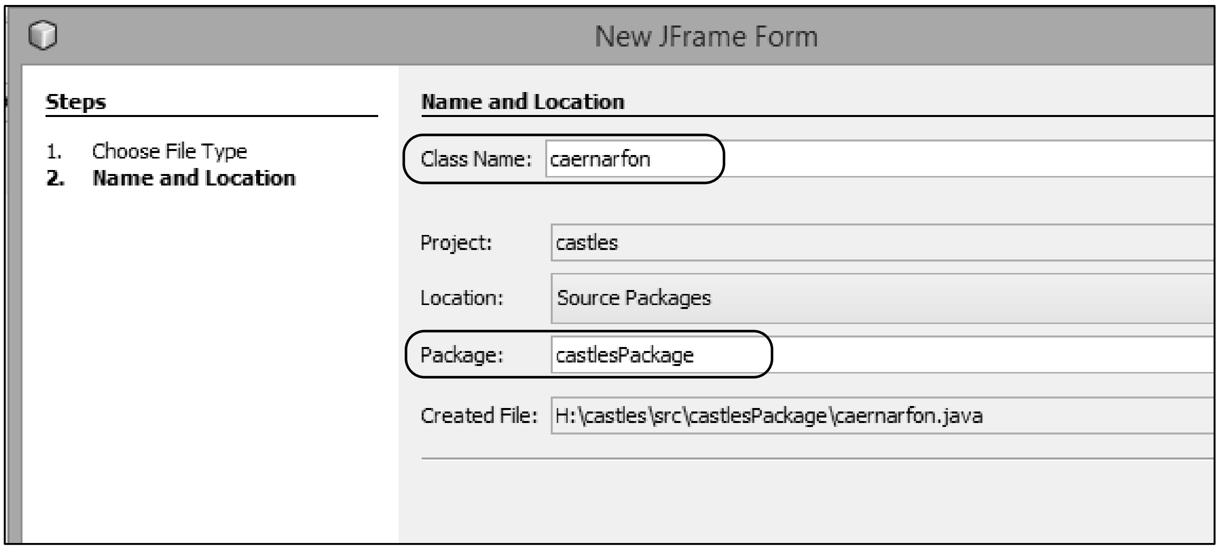


New Name: btnCastellCoch

We will now create a page for Caernarfon Castle. Locate the ***castlePackage*** in the project structure. Right-click on ***castlePackage*** and select ***New*** / ***JFrame Form***:



Set the ***Class Name*** as ***caernarfon***. Leave the ***Package*** name as ***castlesPackage***, as the new form with be part of the current program.



New JFrame Form

Steps

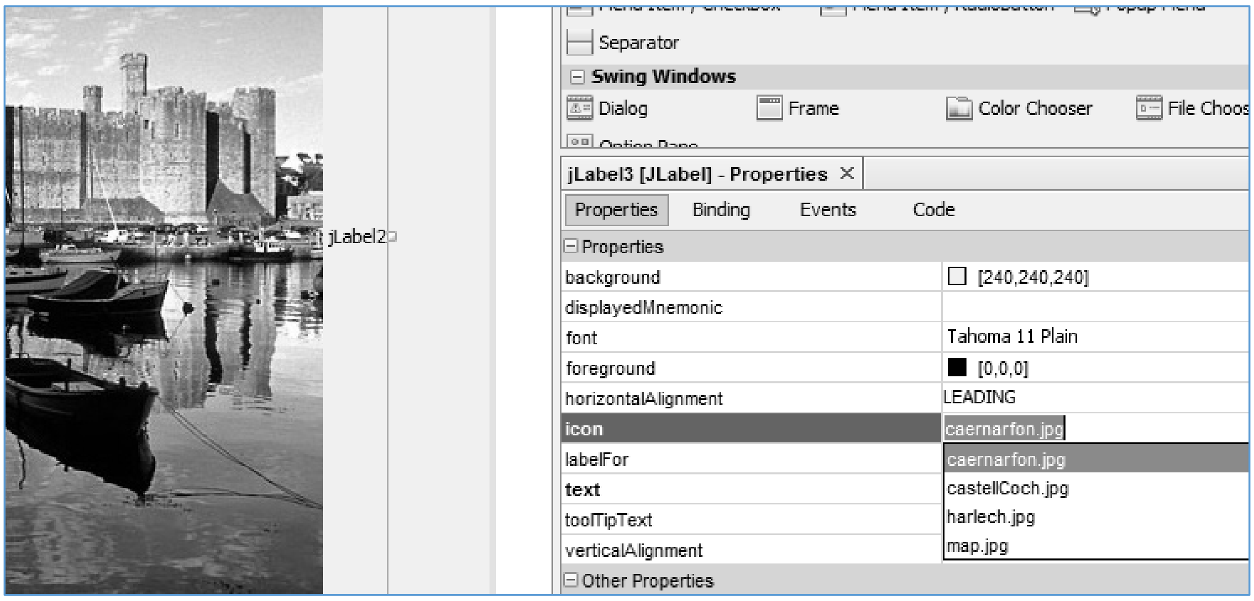
1. Choose File Type
2. **Name and Location**

Name and Location

Class Name:	caernarfon
Project:	castles
Location:	Source Packages
Package:	castlesPackage
Created File:	H:\castles\src\castlesPackage\caernarfon.java

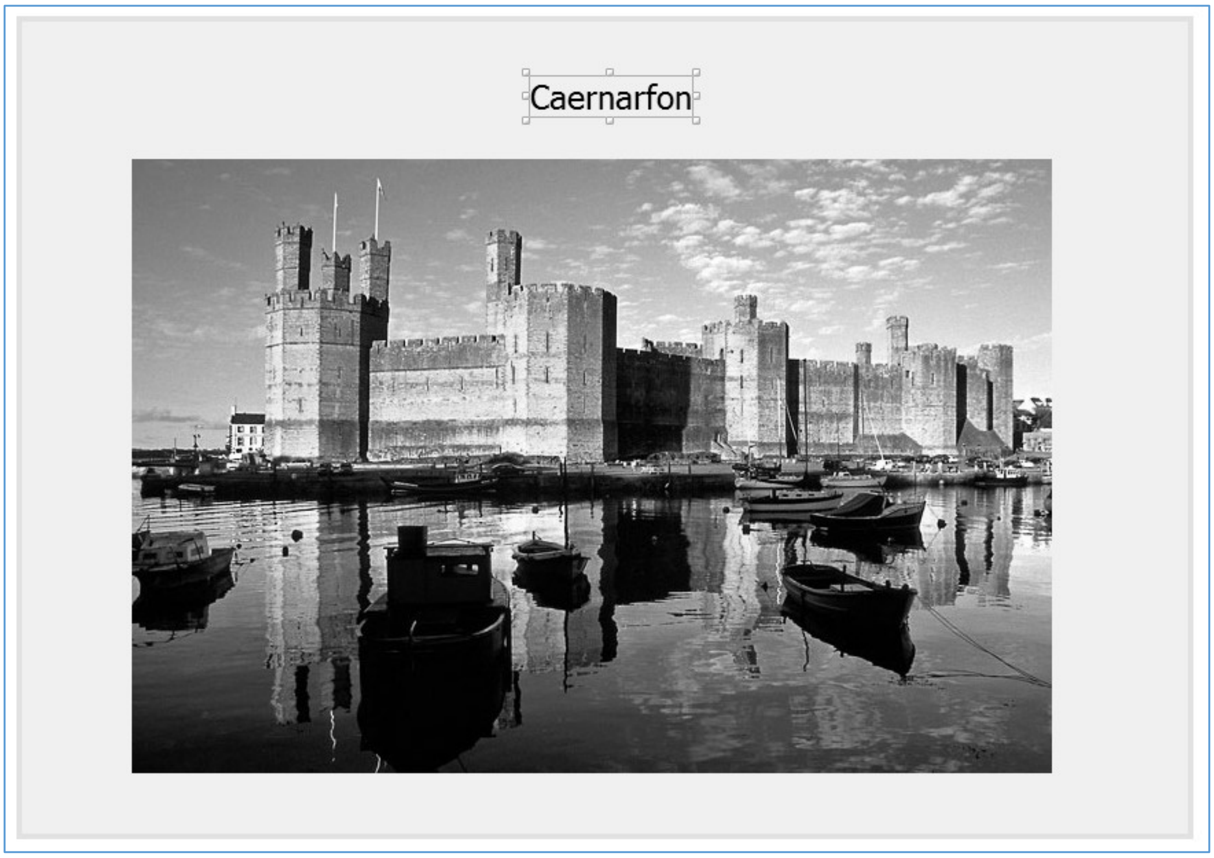
Return to the NetBeans editing page.

Right-click on the form and select ***Set Layout*** / ***Absolute Layout***. Place a label on the form. Go to the icon property and select the Caernarfon picture image from the drop down list. Finally, right-click on the picture image and delete the label text:

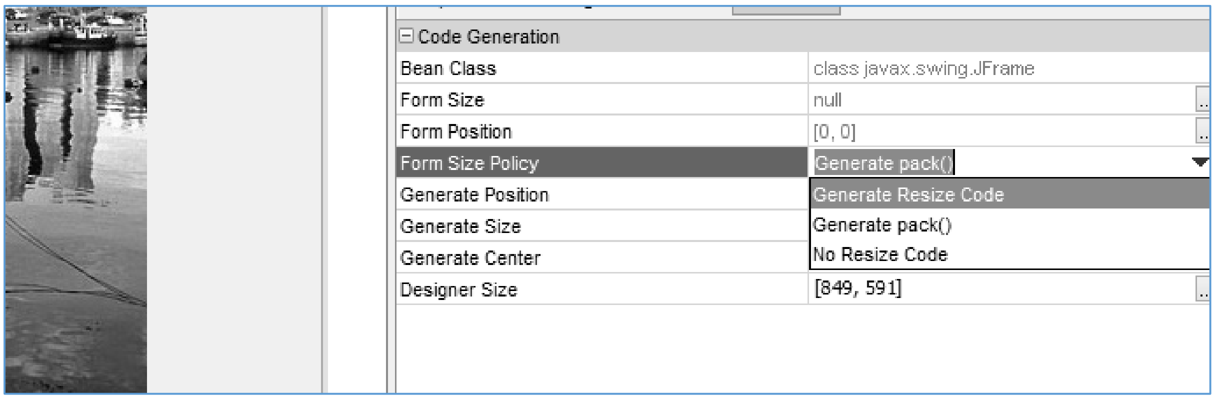


Properties	
background	☐ [240,240,240]
displayedMnemonic	
font	Tahoma 11 Plain
foreground	■ [0,0,0]
horizontalAlignment	LEADING
icon	caernarfon.jpg
labelFor	caernarfon.jpg
text	castellCoch.jpg
toolTipText	harlech.jpg
verticalAlignment	map.jpg

Add another ***label*** to provide a heading on the form. Set the text to "***Caernarfon***":



Click to select the grey area of the form, then go to the ***Code*** tab in the ***Properties*** window and select ***Form Size Policy*** / ***Generate pack()*** / ***Generate Resize Code***:

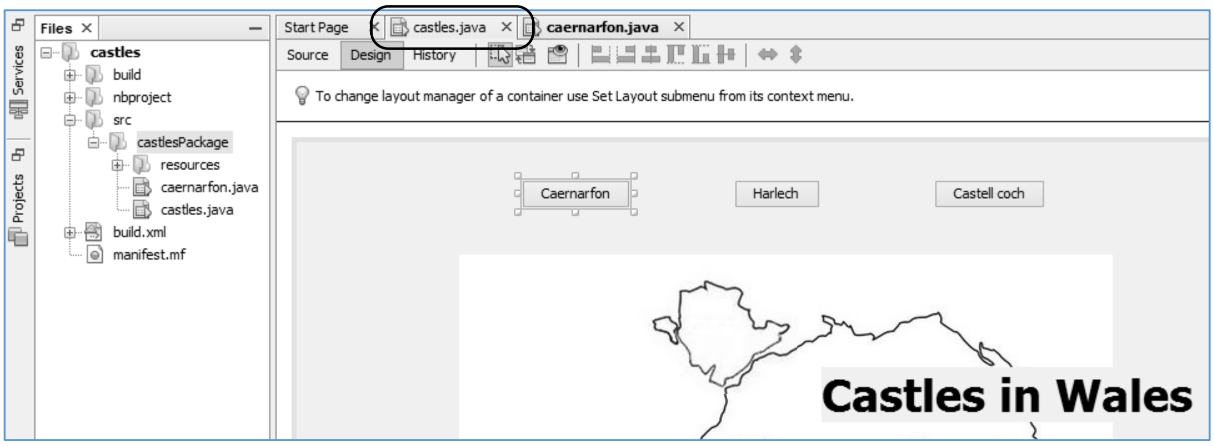


Code Generation

Bean Class	class javax.swing.JFrame
Form Size	null
Form Position	[0, 0]
Form Size Policy	Generate pack()
Generate Position	Generate Resize Code
Generate Size	Generate pack()
Generate Center	No Resize Code
Designer Size	[849, 591]

The form window is now ready to display, but we must link this to the map page so that the window is opened when the "***Caernarfon***" button is clicked.

Click the "***castles.java***" tab to return to the map display. Double click the "***Caernarfon***" button to create a button click method:

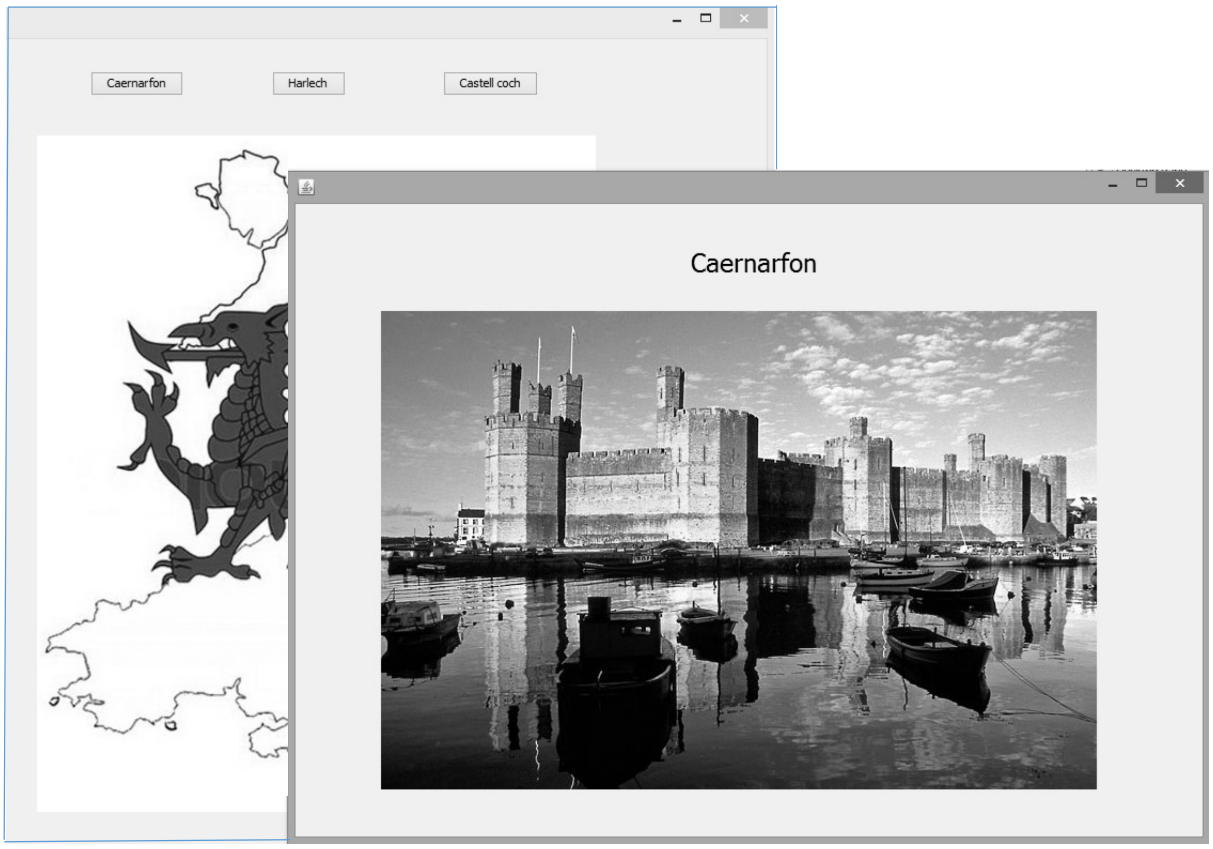


To change layout manager of a container use Set Layout submenu from its context menu.

Add the line of code which will open the Caernarfon window:

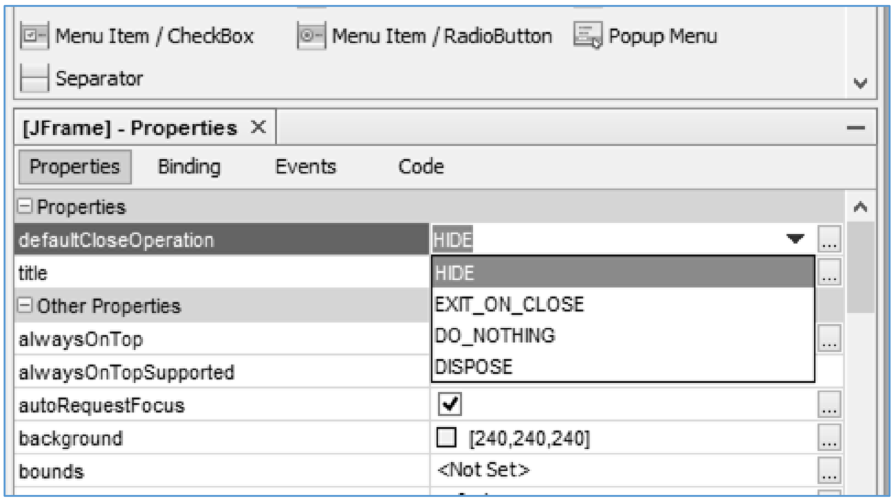
```
private void txtCaernarfonActionPerformed(java.awt.event.ActionEvent evt) {
      new caernarfon().setVisible(true);
  }
```

Run the program. From the map page, click the "***Caernarfon***" button to open the second window:



At this point we will discover a problem. If the ***cross icon*** is clicked to close the Caernarfon page, the program closes completely, rather than returning the user to the map page.

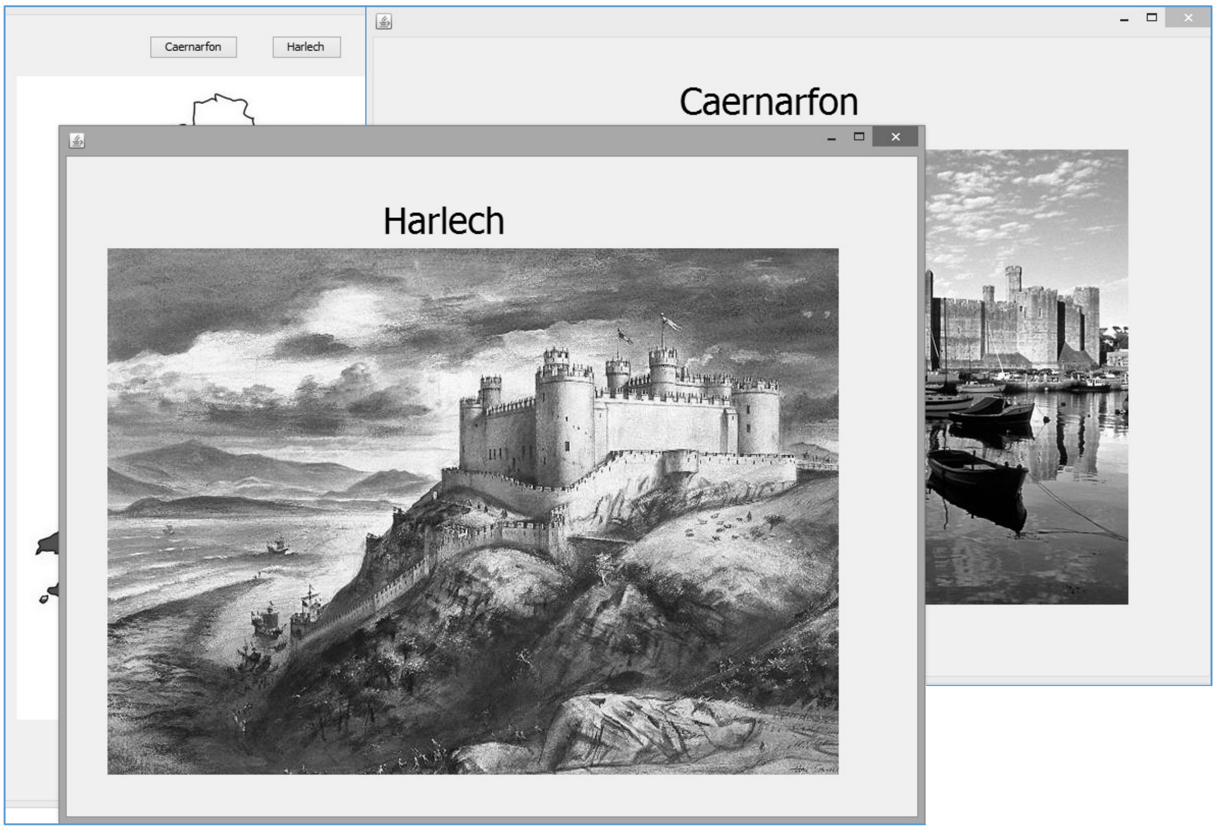
We must set the way in which the program responds when the Caernarfon Castle window is closed by the user. Return to the Caernarfon form, and go to the Properties window. Open the drop-down list for the ***defaultCloseOperation*** property, and select '***Hide***'.



Properties	
defaultCloseOperation	HIDE
title	
Other Properties	
alwaysOnTop	
alwaysOnTopSupported	
autoRequestFocus	✓
background	[240,240,240]
bounds	<Not Set>

Re-run the program. It should now be possible to close the Caernarfon page and return to the map display.

Use the same techniques to set up pages to display pictures of the other two castles:



For the second project in this section, we will produce a program to issue tickets for passengers travelling by train. The details of the tickets will be input on one form, then the ticket displayed on a different form. This will involve transferring data from one form to the other.

Specification for the project:

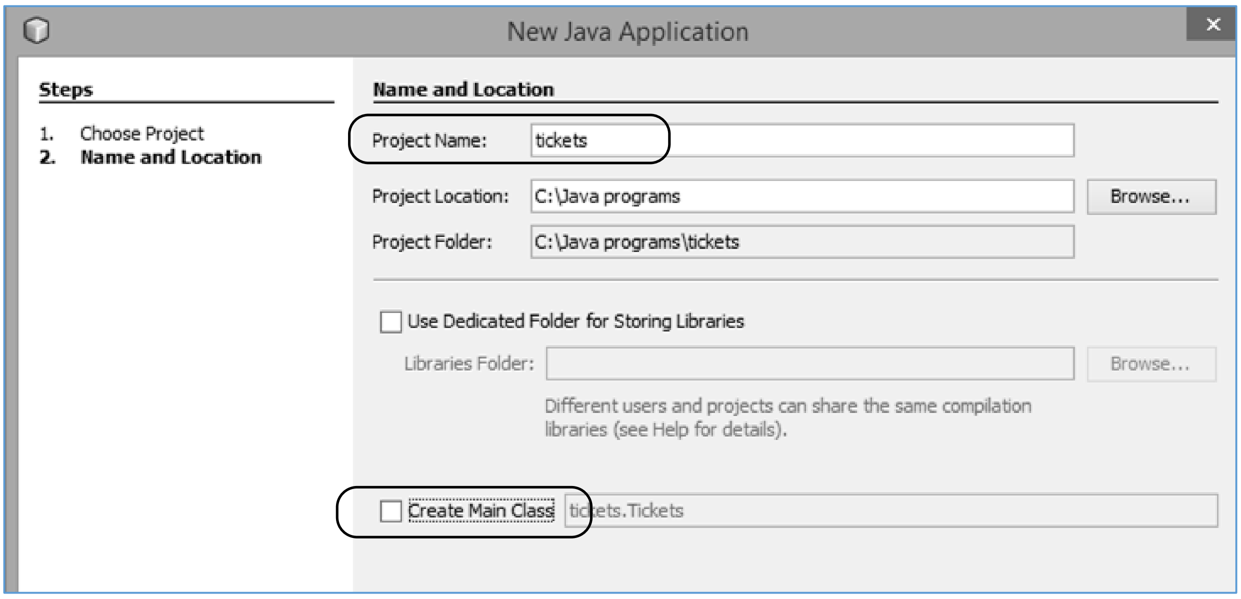
A narrow gauge railway in North Wales requires a computer program to calculate ticket prices and keep a record of the total value of tickets issued each day.

The fares are calculated according to the following rules:

- The adult single fare for a journey along the line is £8.20.
- The child single fare is 60% of the adult fare.
- The return fare is 1½ times the single fare.
- Each passenger wishing to travel first class pays a supplement of £3.00, which covers either a single or return journey.
- Groups of 4 or more people travelling together receive a discount of 10% on the total fare.

You are asked to design and produce a program to issue the tickets.

Close all projects, then select ***New Project*** / ***Java*** / ***Java Application***. Set the ***Project Name*** to ***tickets*** and select the folder location for storing the project. Ensure that ***Create Main Class*** is not ticked:


New Java Application

Steps
1. Choose Project
2. **Name and Location**

Name and Location

Project Name: tickets

Project Location: C:\Java programs

Project Folder: C:\Java programs\tickets

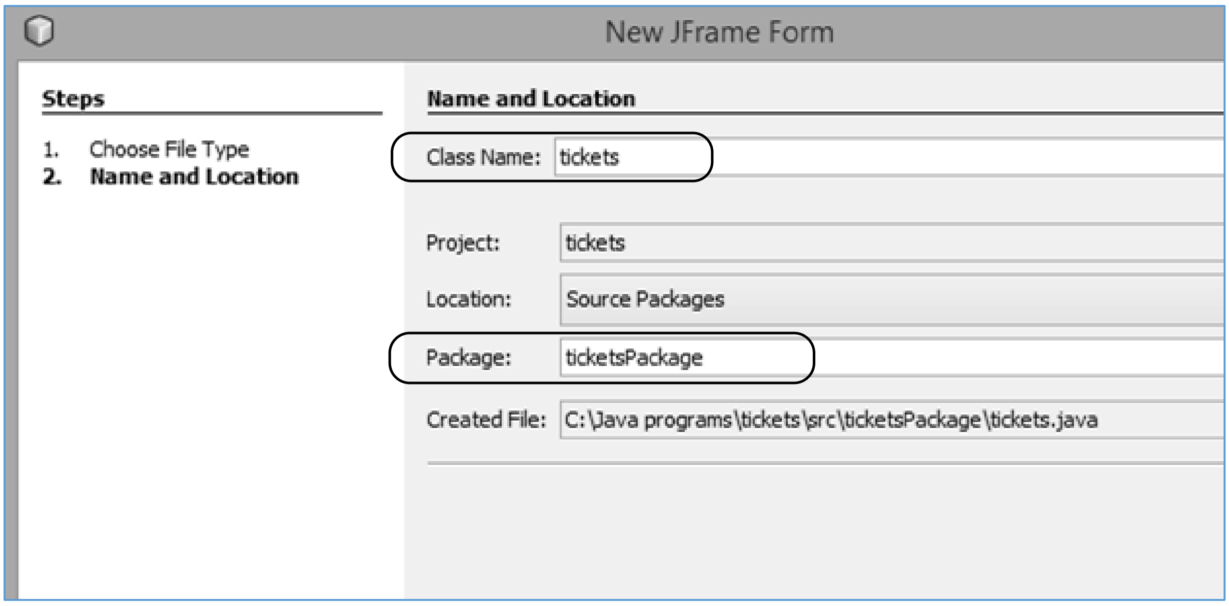
☐ Use Dedicated Folder for Storing Libraries

Libraries Folder:

Different users and projects can share the same compilation libraries (see Help for details).

☐ Create Main Class tickets.Tickets

Return to the NetBeans editing page. Right-click on ***tickets*** in the ***project window***, and select ***New*** / ***JFrame Form***. Set the ***Class Name*** as ***tickets***, and the ***Package*** as ***ticketsPackage***:


New JFrame Form

Steps
1. Choose File Type
2. Name and Location

Name and Location

Class Name: tickets

Project: tickets

Location: Source Packages

Package: ticketsPackage

Created File: C:\Java programs\tickets\src\ticketsPackage\tickets.java

Return to the NetBeans editing page. Carry out the steps to set the correct format for the ***form***:

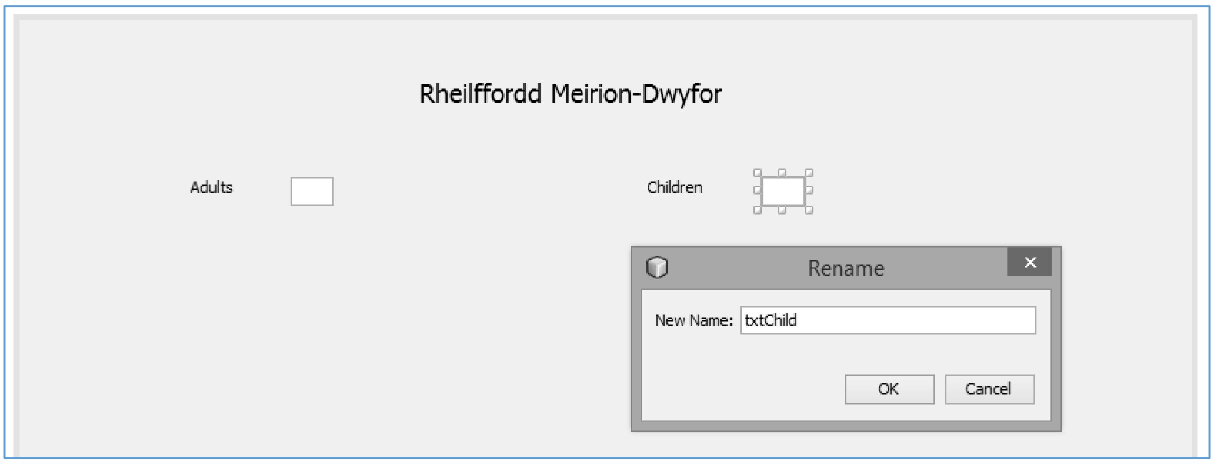
- Go to the ***Code*** tab in the ***Properties*** window and select ***Form Size Policy*** / ***Generate pack()*** / ***Generate Resize Code***.
- Right-click on the form and select ***Set Layout*** / ***Absolute Layout***.
- Click the ***Source*** tab to change to the program code window. Locate the ***main*** method. Open the lines of ***'Look and feel'*** code by clicking the ***+ icon***, then repalce ***"Nimbus"*** by ***"Windows".***

Run the program and accept the ***main class*** which is offered. Check the appearence of the form, then close the program window and return to the NetBeans editing page.

The ticket program will begin by asking the number of adults and children who will be travelling. Add the labels 'Adults' and 'Children', along with text fields. Set the variable names for the text fields to:

txtAdult
txtChild

Also add a heading with the name of the railway, and increase the font to a suitable size:


Rheilffordd Meirion-Dwyfor

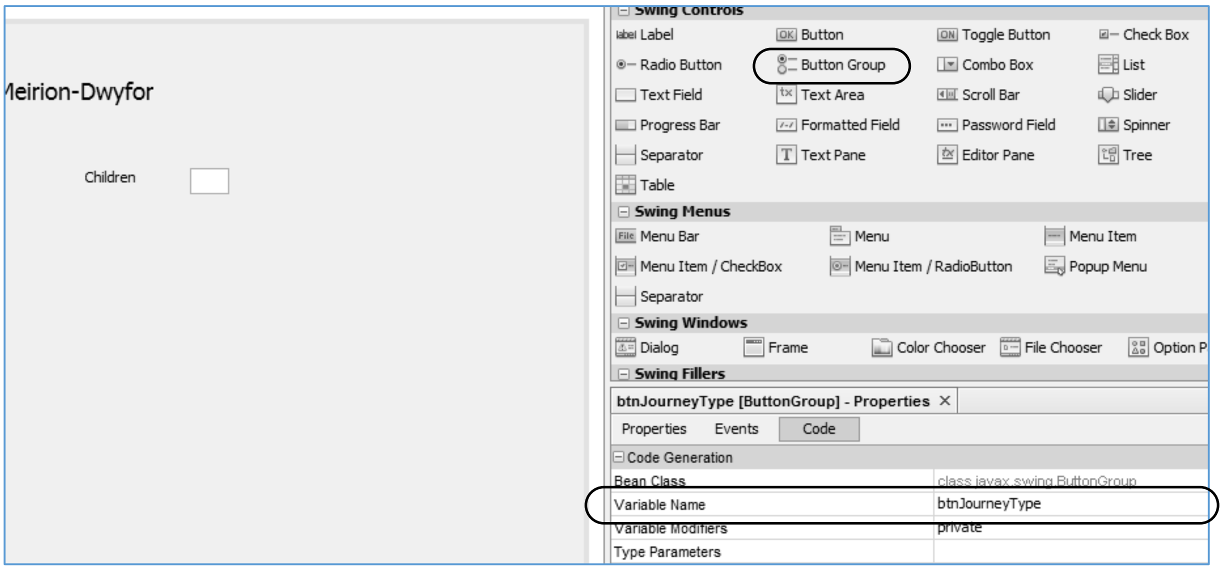
Adults

Children

Rename

New Name: txtChild

We will now set up a group of two ***radio buttons*** for selecting either a single or return ticket. Begin by locating the ***ButtonGroup*** icon and dragging this onto the ***form***. Do not be concerned that absolutely nothing appears on the form! This is a hidden component.

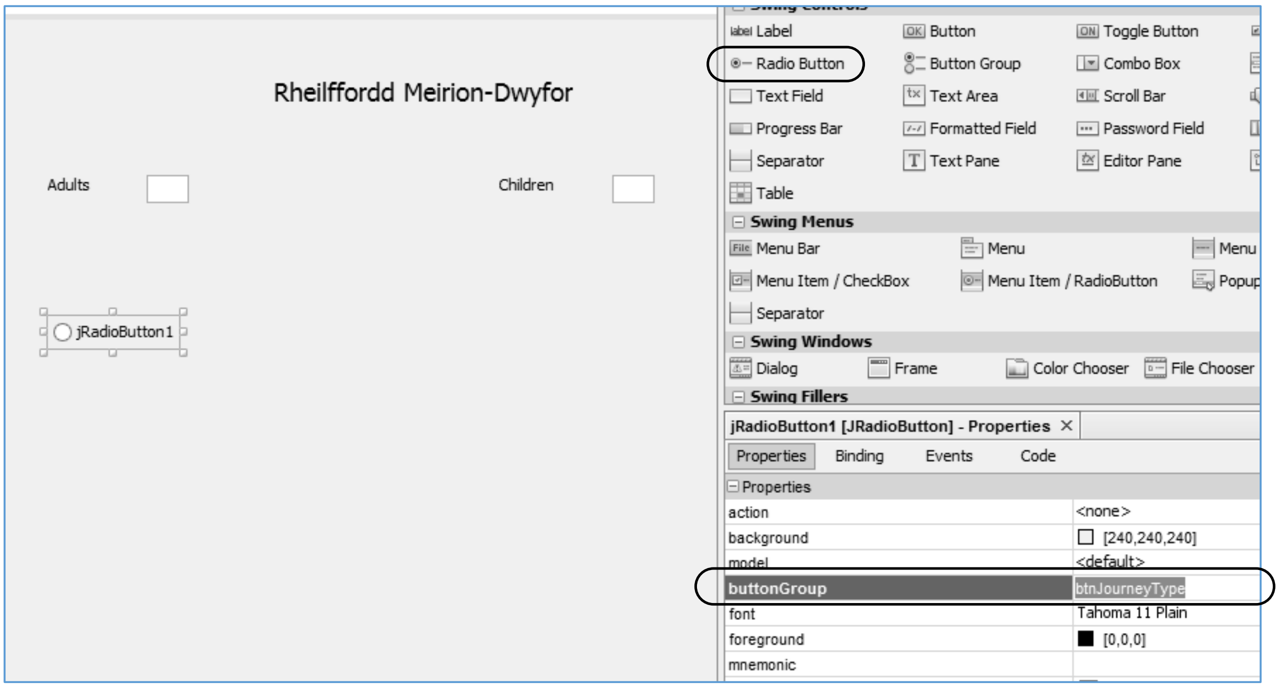


Bean Class	class javax.swing.ButtonGroup
Variable Name	btnJourneyType
Variable Modifiers	private
Type Parameters	

Go to the ***Code*** window and change the ***Variable Name*** for the ***Button Group*** to:

btnJourneyType

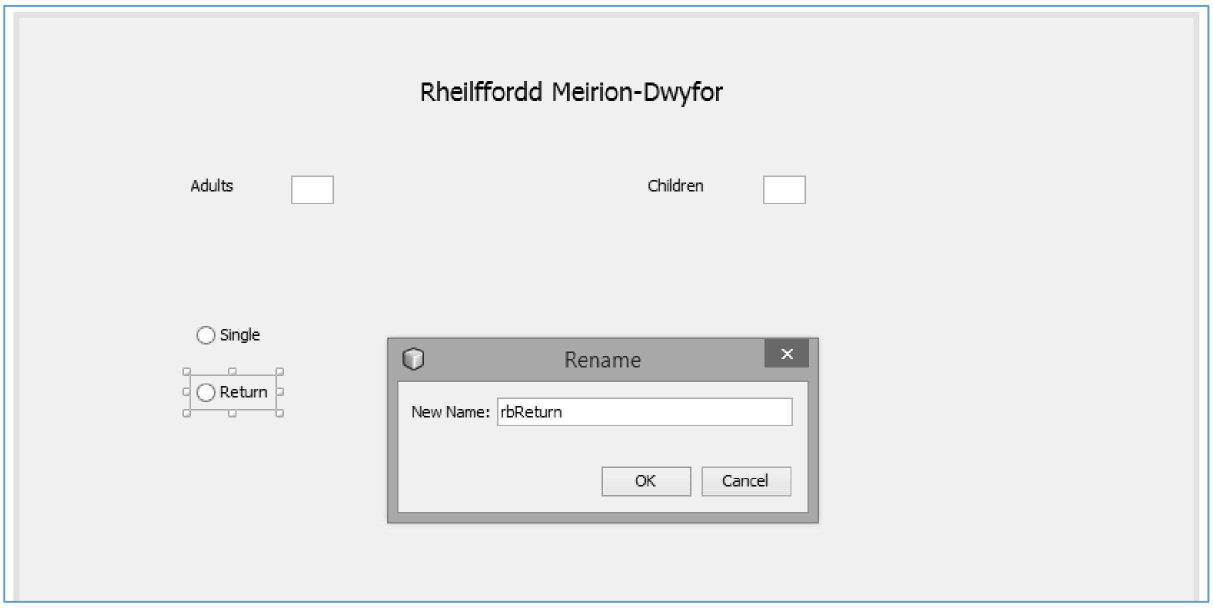
Select a ***Radio Button*** component from the Palette, and drag and drop this on the form. Go to the Properties window and locate the ***buttonGroup*** property. Select ***btnJourneyType*** from the drop-down list:



action	<none>
background	[240,240,240]
model	<default>
buttonGroup	btnJourneyType
font	Tahoma 11 Plain
foreground	[0,0,0]
mnemonic	

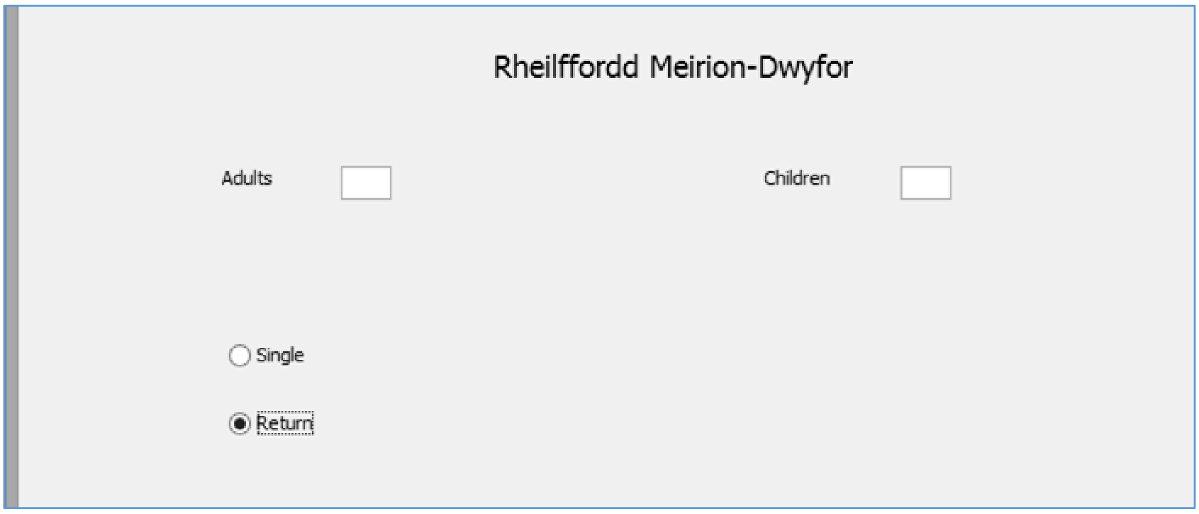
Change the ***Text*** of the ***radio button*** to '***Single***', and the ***Variable Name*** to '***rbSingle***'.

Add a second ***radio button*** in the same way, again setting the ***buttonGroup*** property to ***btnJourneyType***. Change the Text of this radio button to '***Return***', and the Variable Name to '***rbReturn***':



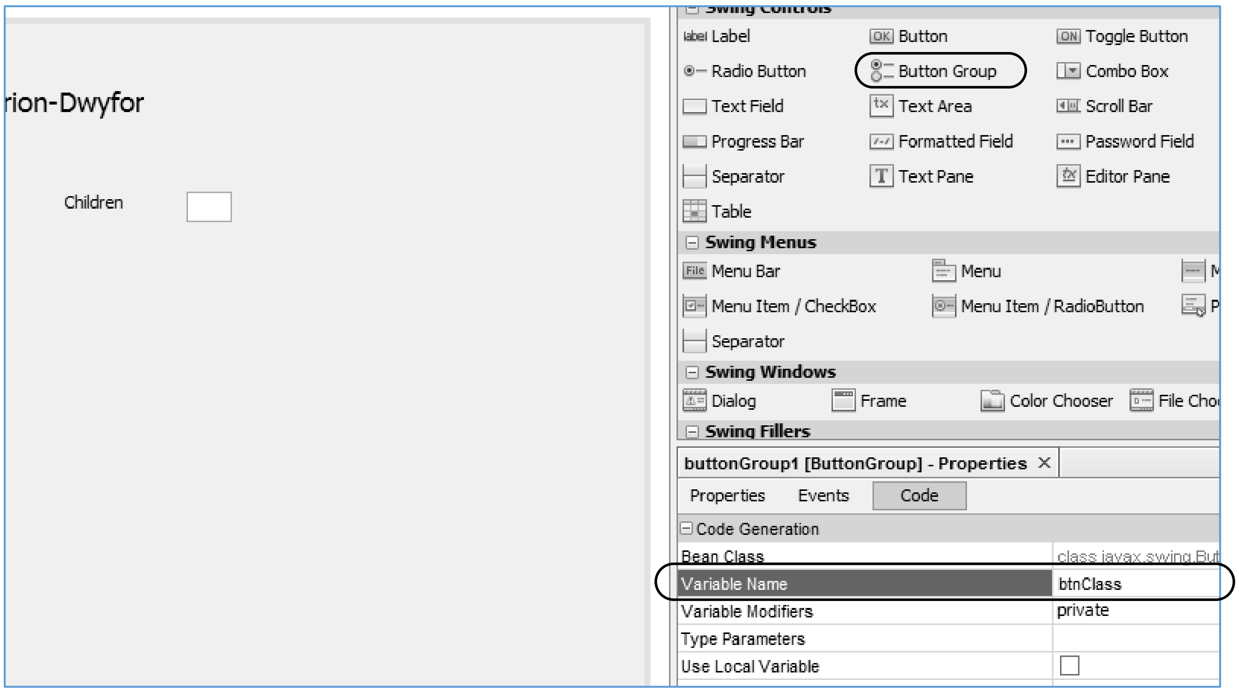
New Name: rbReturn

Run the program. It should only be possible to select one of the radio buttons at a time. We have arranged that they work together as a group, so as one button is selected then the other is cancelled:



We are now going to set up a second radio group for selecting a First Class or Second Class ticket. Begin by dragging a ***Button Group*** component onto the form, although nothing will be displayed. Select the ***Code*** tab and set the ***Variable Name*** for the ***Radio Group*** to:

btnClass



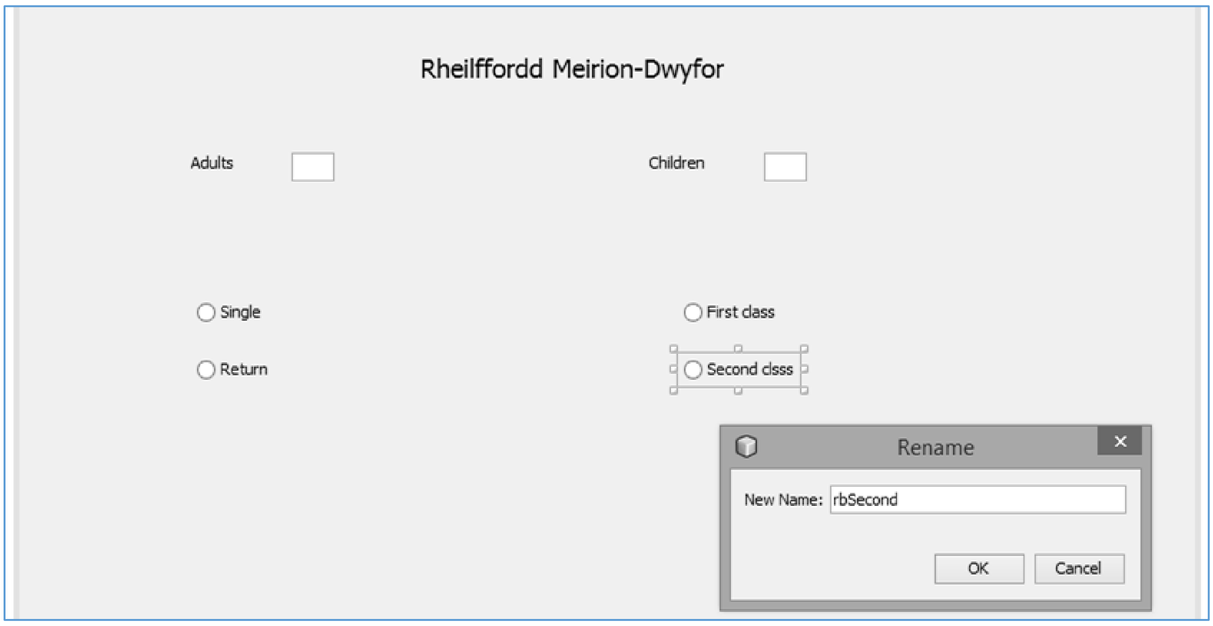
buttonGroup1 [ButtonGroup] - Properties

Code Generation

Bean Class	class javax.swing.But
Variable Name	btnClass
Variable Modifiers	private
Type Parameters	
Use Local Variable	☐

Add two ***Radio Button*** components, changing the Text to '***First Class***' and '***Second Class***'. Give the radio buttons the variable names:

rbFirst
rbSecond

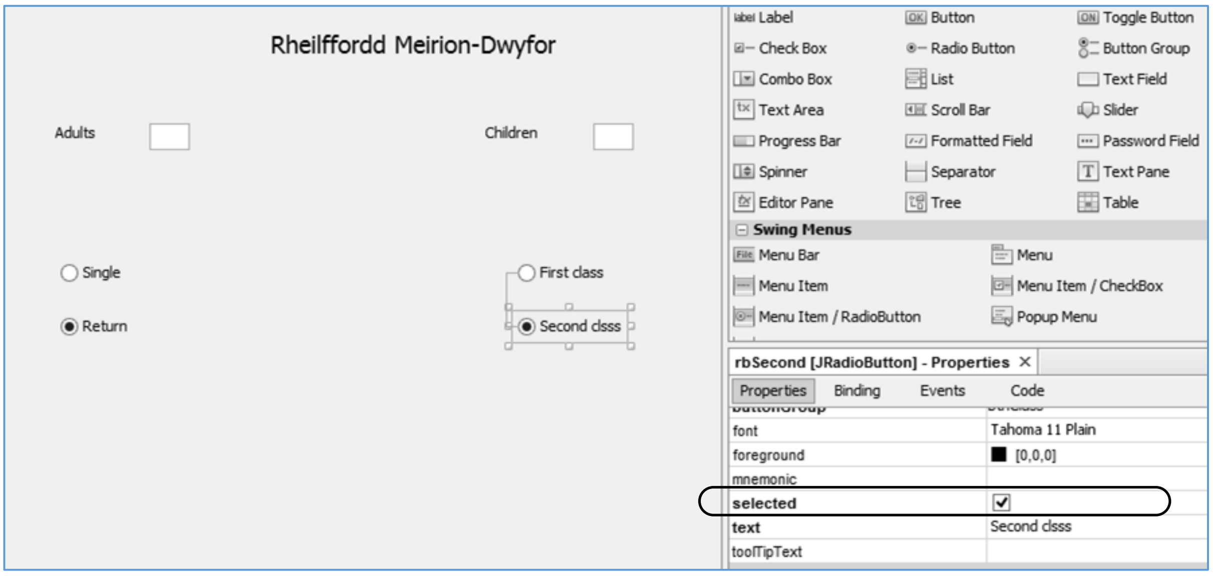


New Name: rbSecond

Click on each of these radio buttons in turn and go to the Properties window. Locate the ***buttonGroup*** property and select ***btnClass*** from the drop-down list.

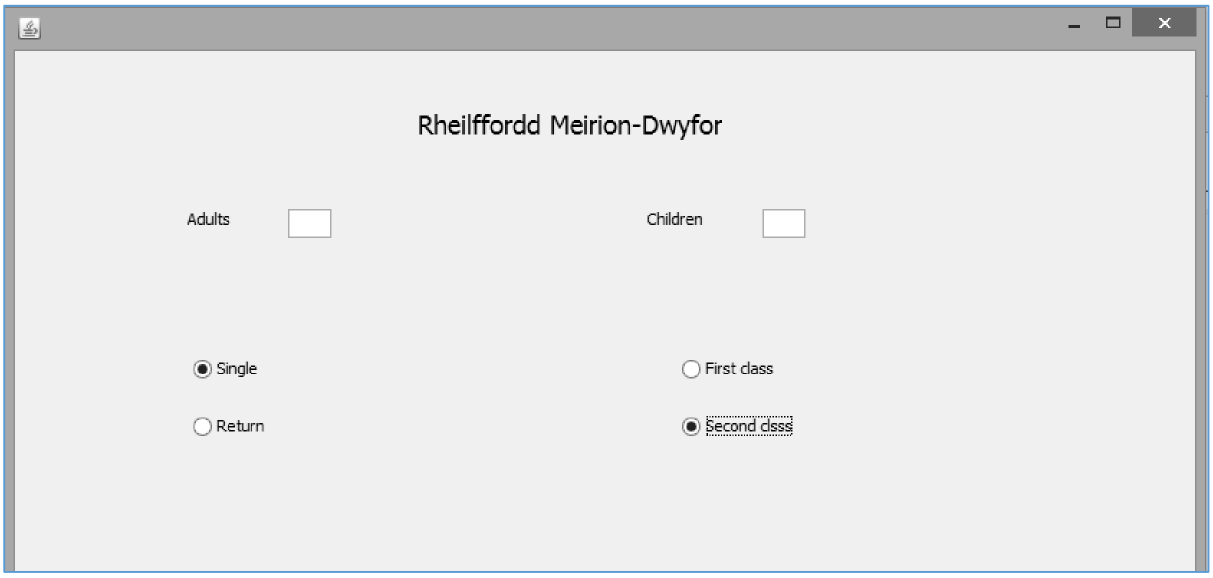
It is likely that most railway passengers will want second class return tickets, so we can save time for the booking office staff by setting these radio buttons to be initially selected.

Click on the ***Return*** radio button and put a tick in the box for the ***selected*** property. Do the same for the ***Second Class*** radio button:

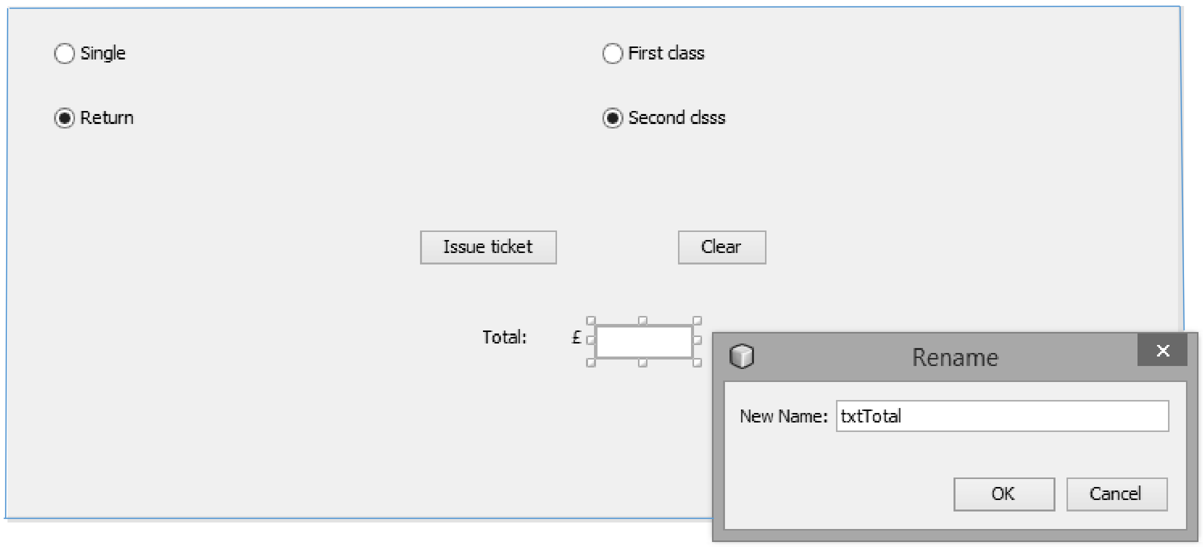


font	Tahoma 11 Plain
foreground	[0,0,0]
mnemonic	
selected	✓
text	Second clsss
toolTipText	

Run the program. The ***Return*** and ***Second Class*** options should be selected, but these can be changed by clicking the other radio buttons. The two radio groups should be working independently of one another, with a single option selected in each group:



The final step in setting up the input page is to add ***buttons*** with the captions "***Issue ticket***" and "***Clear***", a ***label*** with the caption "***Total: £***", and a ***text field*** to display the ticket cost. Rename the buttons as ***btnIssue*** and ***btnClear***, and the text field as ***txtTotal***:



Click the ***Source*** tab to change to the program code page.

We can now begin work on the processing stages of the program. It will be helpful to give the user a warning message if data is entered in an incorrect format. Add a line near the top of the program to load the code to display a message box:

```
package ticketsPackage;

import javax.swing.JOptionPane;

public class tickets extends javax.swing.JFrame {
```

Return to the ***Design*** page and double click the "***Issue ticket***" button to create a button click method.

The first step in producing the ticket is to obtain the numbers of ***adults*** and ***children*** who will be travelling.

- It is possible that a ticket will be issued only for a group of adults, or only for a group of children. One of the text fields, ***txtAdult*** or ***txtChild*** will therefore be left blank. We do not want this to cause a problem when the ticket price is calculated, so the number of persons should be recorded as ***zero*** if the text field is blank.

Add program lines to the ***btnIssueActionPerformed*** method which will carry out this action:

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

      int adult;
      int child;
      boolean returnWanted;
      boolean firstWanted;
      double ticketcost;

      try
      {
          if (txtAdult.getText().equals(""))
          {
             adult = 0;
          }
          else
          {
              adult = Integer.parseInt(txtAdult.getText());
          }

          if (txtChild.getText().equals(""))
          {
              child = 0;
          }
          else
          {
              child = Integer.parseInt(txtChild.getText());
          }
      }

    catch(NumberFormatException e)
    {
       JOptionPane.showMessageDialog(tickets.this, "Incorrect number format");
    }

}
```

Notice that we began the method by defining ***variables*** to hold data during the calculation of the ticket price. Since these variables will only be used within this one method, they have been defined within the method itself, and are known as ***local variables***. If variables are defined at the start of the program, they are available for use in any method and are known as ***global variables***. Programmers often prefer to use local variables; these only exist within a single method so there is less chance of accidentally causing errors in other parts of the program.

The variable definitions:

```
int adult;
int child;
```

set up integer whole numbers, which is sensible for counting the numbers of passengers.

The definition:

```
double ticketcost;
```

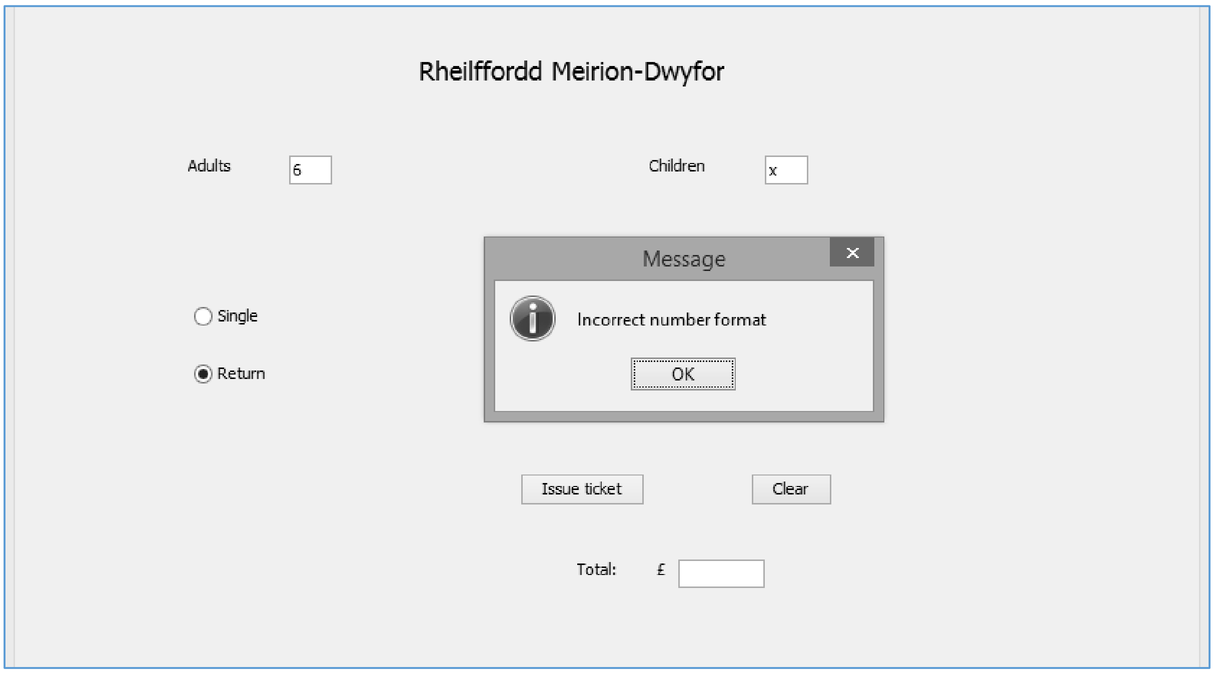
sets up a decimal number which will be needed to represent the ticket cost in pounds and pence.

The definitions:

```
boolean returnWanted;
boolean firstWanted;
```

set up TRUE/FALSE variables. For example, if the passenger buys a return ticket then ***returnWanted*** will be ***TRUE***, but if they buy a single ticket then ***returnWanted*** will be ***FALSE***.

Run the program to test the user interface. Enter correct ***integer*** numbers for adults and children, then click the "***Issue ticket***" button. These numbers should be accepted by the program. There should also be no problem if the text fields are left blank. However, a message box should appear if letters or other symbols are entered in a text field:

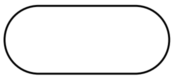


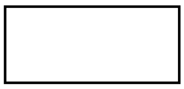
Close the program window and return to the ***Source*** code page.

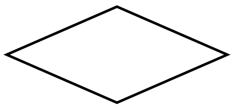
We now need to devise a strategy for calculating the ticket cost. We can:

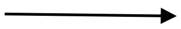
- Begin by assuming that the passengers will be making a ***second class single*** journey. The total ticket cost can then be provisionally calculated.
- Check if the radio button has been clicked for a ***Return*** journey. If so, the cost of the ticket is increased by 50%.
- Check if the radio button has been clicked for ***First Class***. If so, the first class supplement can be added for each of the passengers.
- Count the number of passengers. If this is four or more, then the 10% group discount will be deducted.

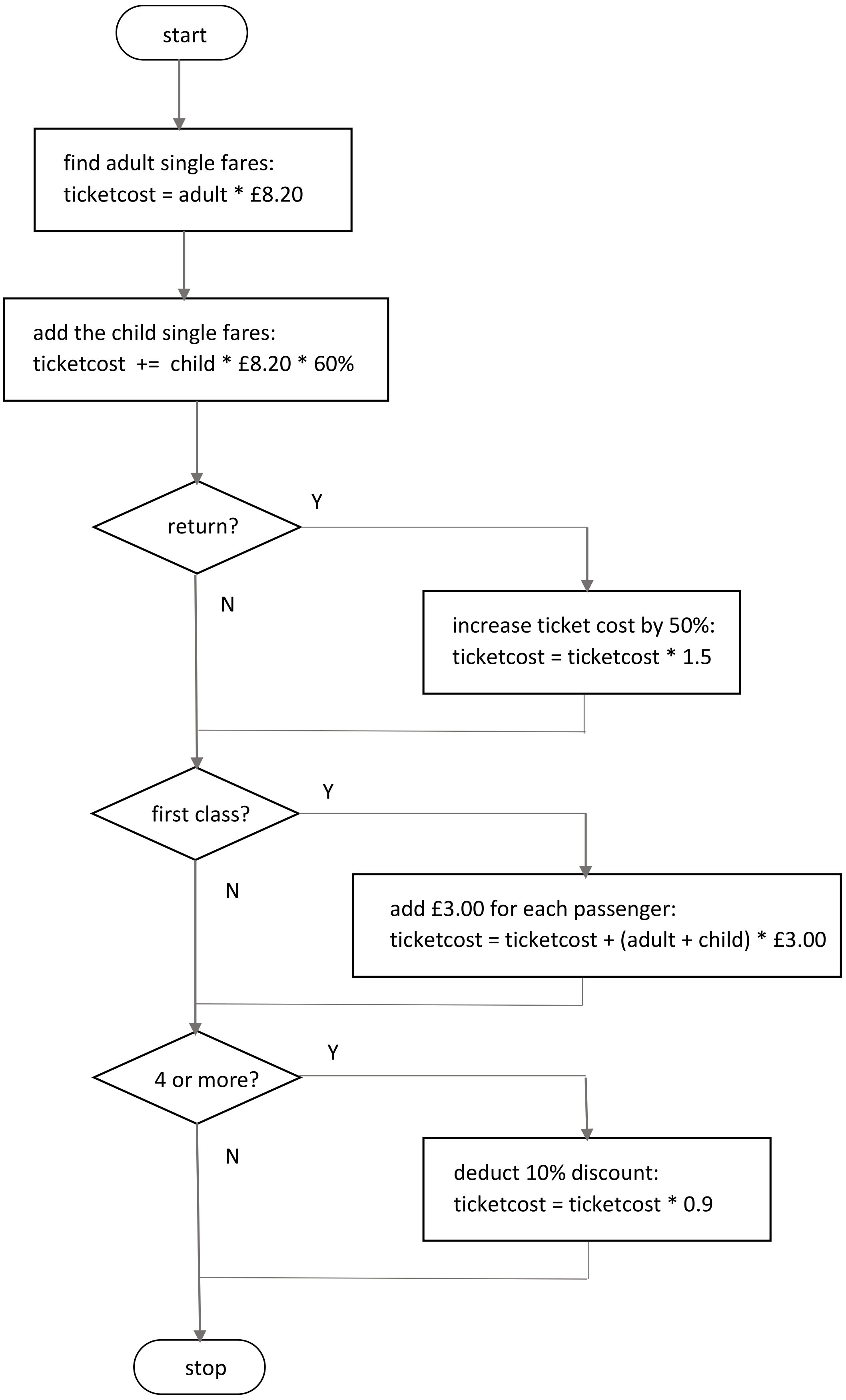
This sequence of steps can be conveniently displayed as a program flow chart. Standard symbols are used:


start or stop


process


decision


direction through the program

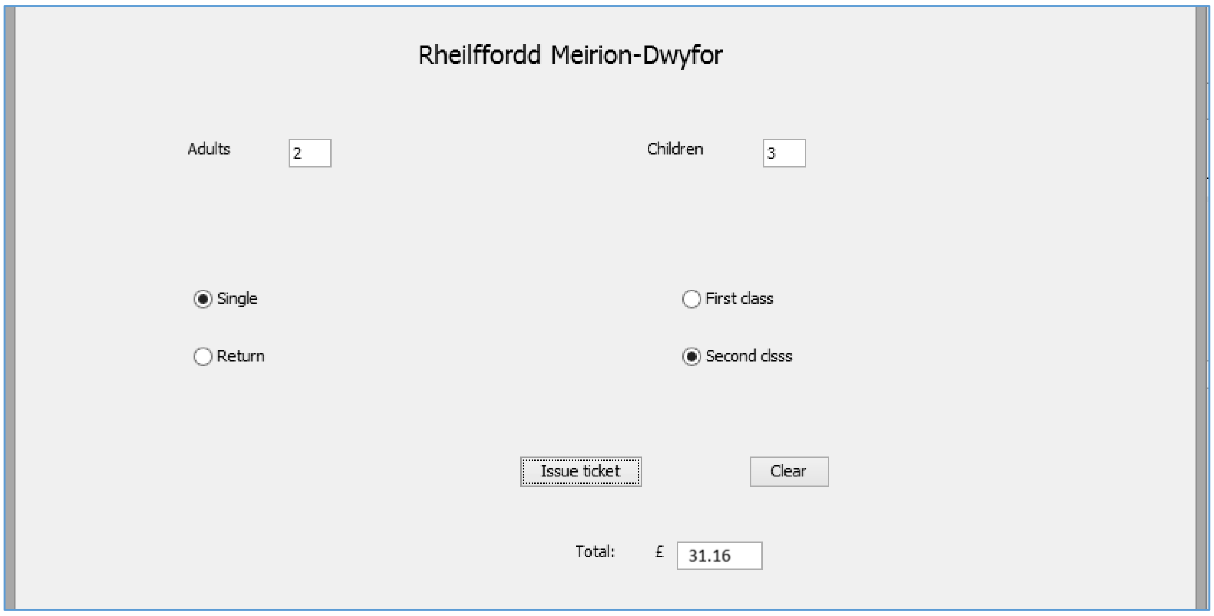

start
find adult single fares:
ticketcost = adult * £8.20
add the child single fares:
ticketcost += child * £8.20 * 60%
return?
Y
N
increase ticket cost by 50%:
ticketcost = ticketcost * 1.5
first class?
Y
N
add £3.00 for each passenger:
ticketcost = ticketcost + (adult + child) * £3.00
4 or more?
Y
N
deduct 10% discount:
ticketcost = ticketcost * 0.9
stop

Begin by adding the program code to calculate the cost of the tickets assuming that the passengers require ***second class single***. We then check whether the radio button ***rbReturn*** is selected. If so, the price is increased by 50% for return tickets.

Output the result to the ***txtTotal*** text field:

```
        if (txtChild.getText().equals(""))
        {
            child = 0;
        }
        else
        {
            child = Integer.parseInt(txtChild.getText());
        }

        ticketcost = adult*8.20;
        ticketcost += child*8.20*0.6;

        if (rbReturn.isSelected())
        {
            returnWanted=true;
            ticketcost = ticketcost * 1.5;
        }
        else
        {
            returnWanted=false;
        }

        txtTotal.setText(String.format("%.2f",ticketcost));

    }
```

Run the program to check that it is working correctly so far. Enter various numbers of adults and children, for single and return journeys. Use a calculator or spreadsheet to confirm that the ticket totals are correct.



Close the program window and return to the Source code page.

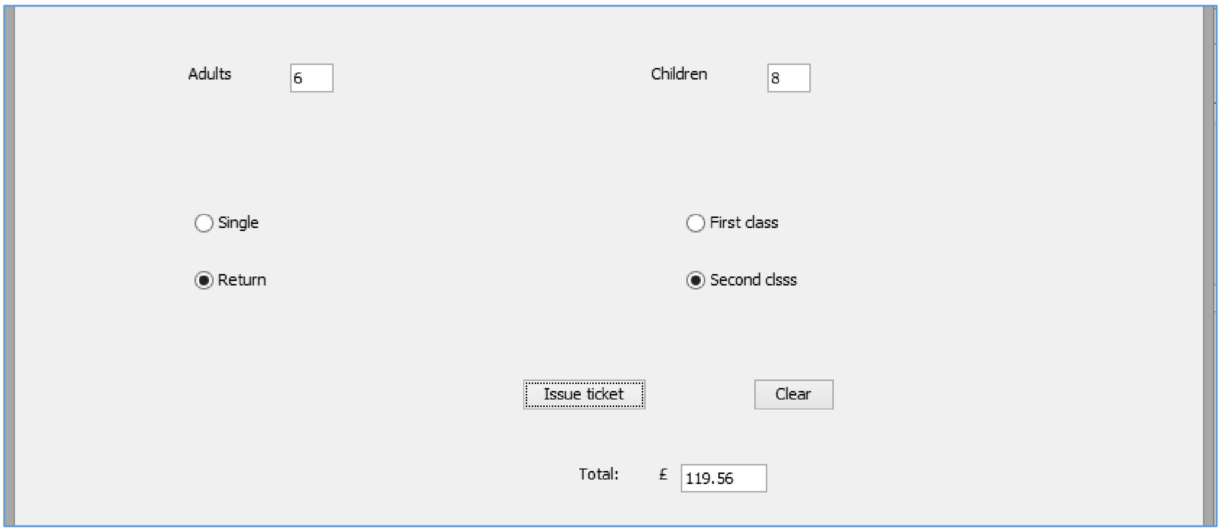
Complete the calculation by inserting program lines to check the ***First Class*** radio button ***rbFirst***, and add the £3.00 supplement for each passenger. We also check for a group of four or more passengers travelling together, and deduct the 10% group discount:

```
        else
        {
            returnWanted=false;
        }

        if (rbFirst.isSelected())
        {
            firstWanted=true;
            ticketcost= ticketcost +(adult+child)*3.00;
        }
        else
        {
            firstWanted=false;
        }


        if ((adult + child) >=4)
        {
           ticketcost = ticketcost * 0.90;
        }

        txtTotal.setText(String.format("%.2f",ticketcost));
```

Return to the ***Design*** page and double click the "***Clear***" button to create a button click method, then add lines of program to clear the text fields:

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

     txtAdult.setText("");
     txtChild.setText("");
     txtTotal.setText("");

  }
```

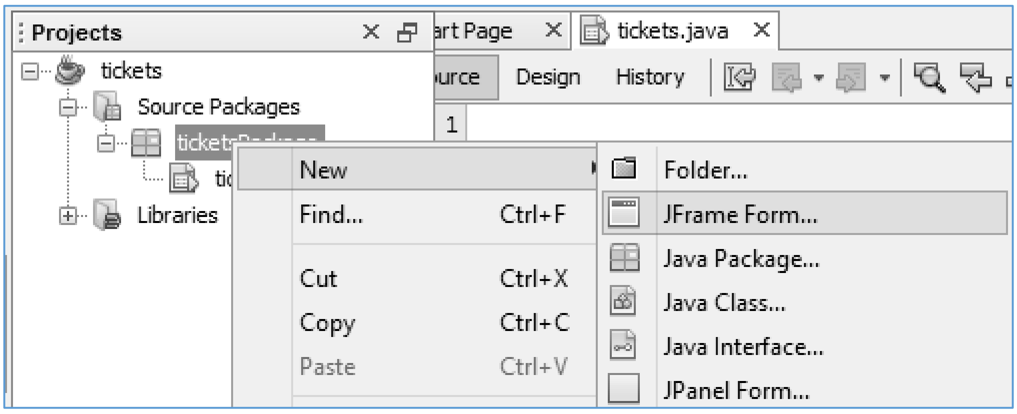
Run the program and test all types of ticket, with or without group discount:



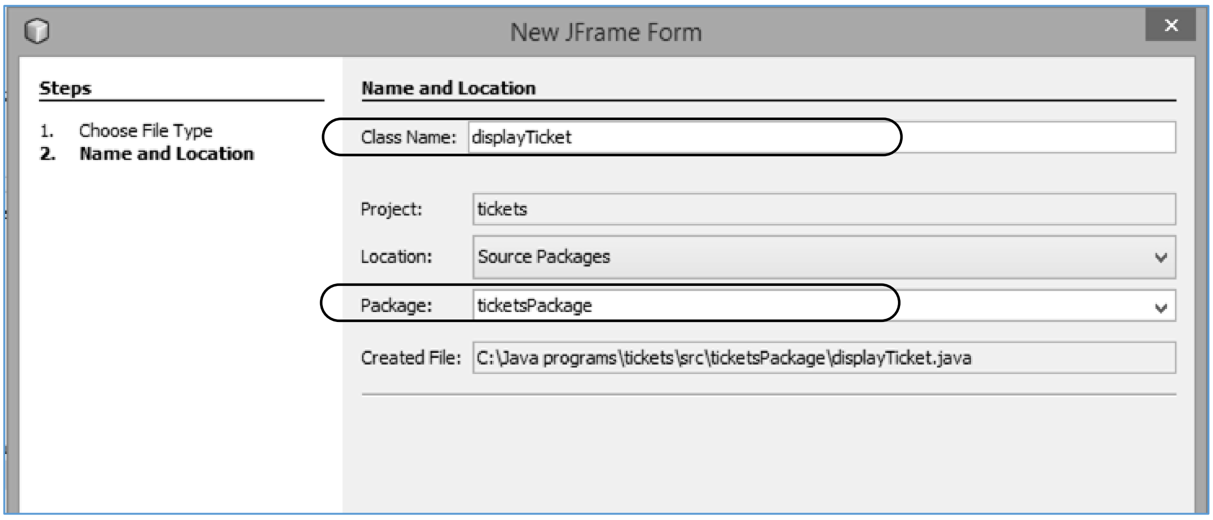
Close the program window and return to the NetBeans editing page.

The next stage of the program is to create a second window where the ticket can be displayed. In a real system, this would then be printed and given to the passengers.

Begin by locating ***ticketPackage*** in the ***Projects*** window. Right-click and select ***New*** / ***JFrame Form***:



Give the ***Class Name*** as ***displayTicket***, but leave the ***Package*** name as ***ticketsPackage***:


New JFrame Form

Steps
1. Choose File Type
2. **Name and Location**

Name and Location

Class Name: displayTicket

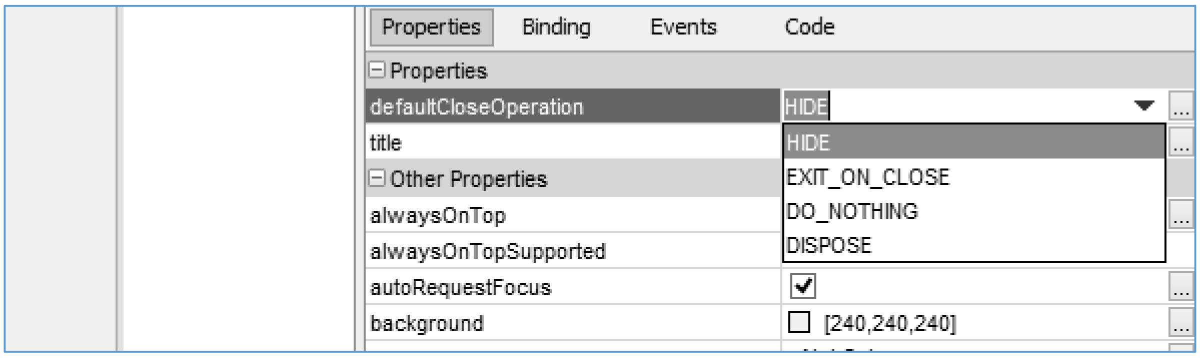
Project: tickets

Location: Source Packages

Package: ticketsPackage

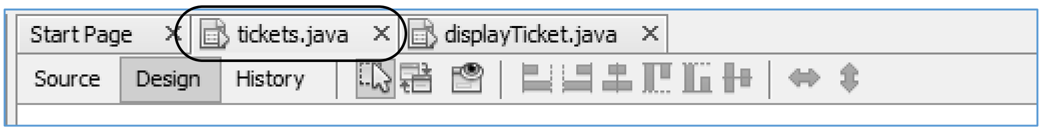
Created File: C:\Java programs\tickets\src\ticketsPackage\displayTicket.java

Return to the NetBeans ***Design*** page. Set the ***defaultCloseOperation*** property in the drop-down list to ***HIDE*** for the displayTicket form:



Properties	
defaultCloseOperation	HIDE
title	
Other Properties	
alwaysOnTop	
alwaysOnTopSupported	
autoRequestFocus	✓
background	[240,240,240]

Use the tab above the Design window to change to the ***tickets.java*** page:



Double click the "***Issue ticket***" button to open the button click method. Locate the line:

```
txtTotal.setText(String.format("%.2f",ticketcost));
```

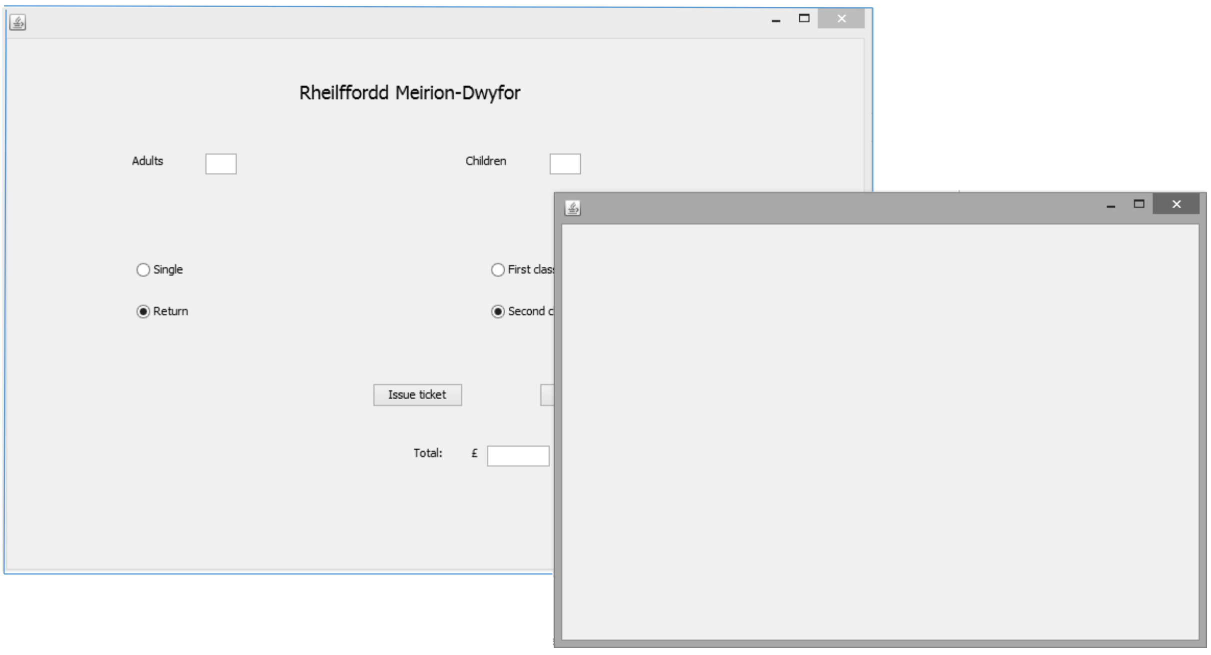
and replace it with:

```
new displayTicket().setVisible(true);
```

This line of code will open the ***displayTicket*** window:

```
        if ((adult + child) >=4)
        {
           ticketcost = ticketcost * 0.90;
        }

        new displayTicket().setVisible(true);

    }
```

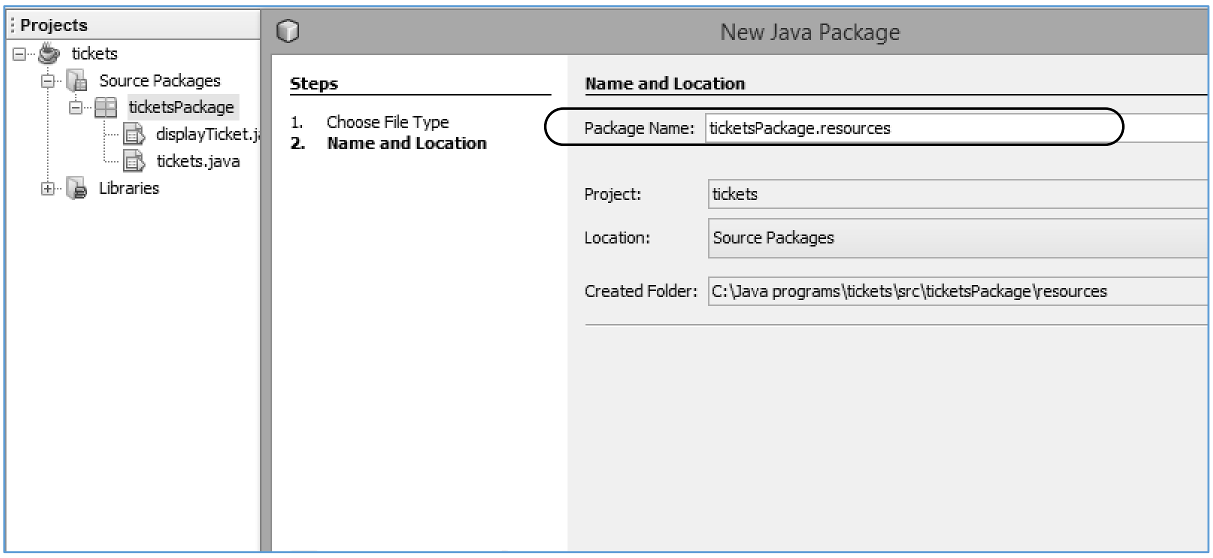
Run the program. Click the "***Issue ticket***" button and check that the second window opens, then close the program window.



If the size of the ***displayTicket*** form was not set correctly, go to the ***Code*** tab in the Properties window and select ***Form Size Policy*** / ***Generate pack()*** / ***Generate Resize Code***.

We will use a photograph image on the new form, so a ***resources*** folder will be needed.

Go to the Projects window and locate ***ticketsPackage***. Right-click and select ***New*** / ***Java Package***. Give this the name ***ticketsPackage.resources***:



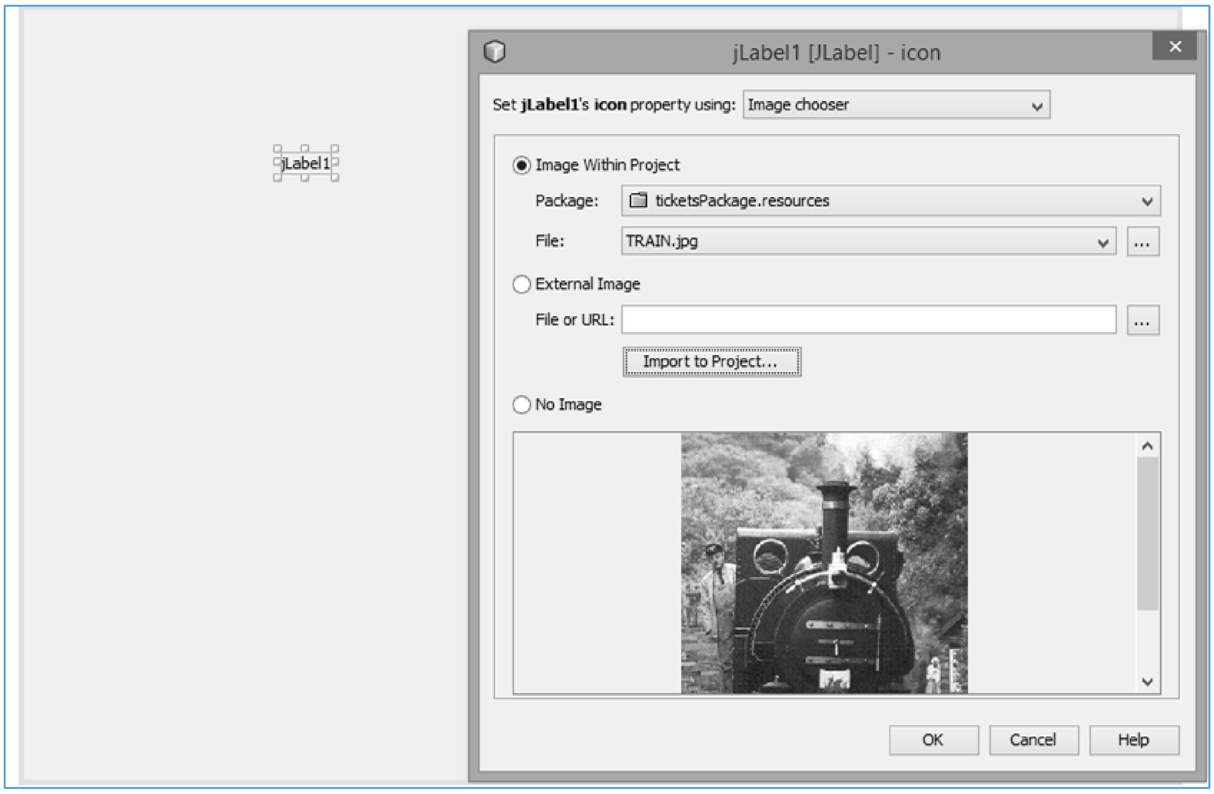
Package Name: ticketsPackage.resources
Project: tickets
Location: Source Packages
Created Folder: C:\Java programs\tickets\src\ticketsPackage\resources

Obtain a photograph of a narrow gauge railway train, and use a graphics application program to set the width to approximately 200 pixels.

Return to the ***ticketDisplay*** form and click the ***Design*** tab to move to the form layout view. Right-click on the form and select ***Layout*** / ***Absolute Layout***.

Place a ***label*** component on the form. Go to the ***icon*** property and click the ellipsis (" **…** ") symbol to open the image selection window.

Use the "***Import to Project***" button to upload the picture to the resources folder. Check that this image is the selected ***File*** and click the ***OK*** button.

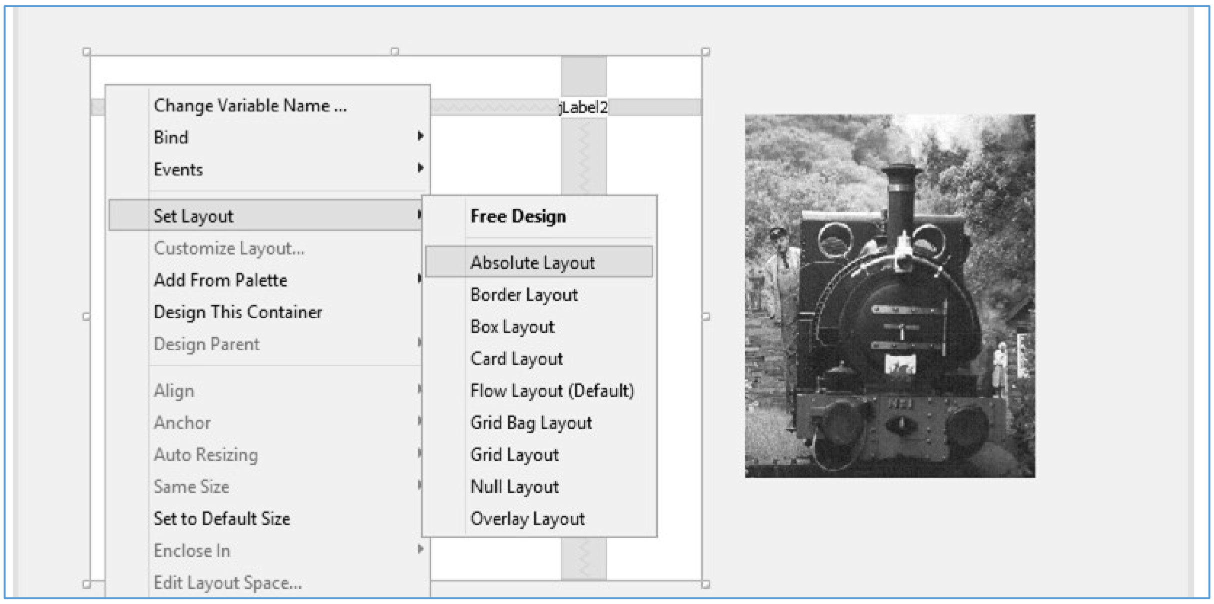


Set **jLabel1**'s **icon** property using: Image chooser
(●) Image Within Project
Package: ticketsPackage.resources
File: TRAIN.jpg
( ) External Image
File or URL:
( ) No Image

Return to the Design screen and check that the train photograph is displayed correctly. Right-click to delete the label caption.

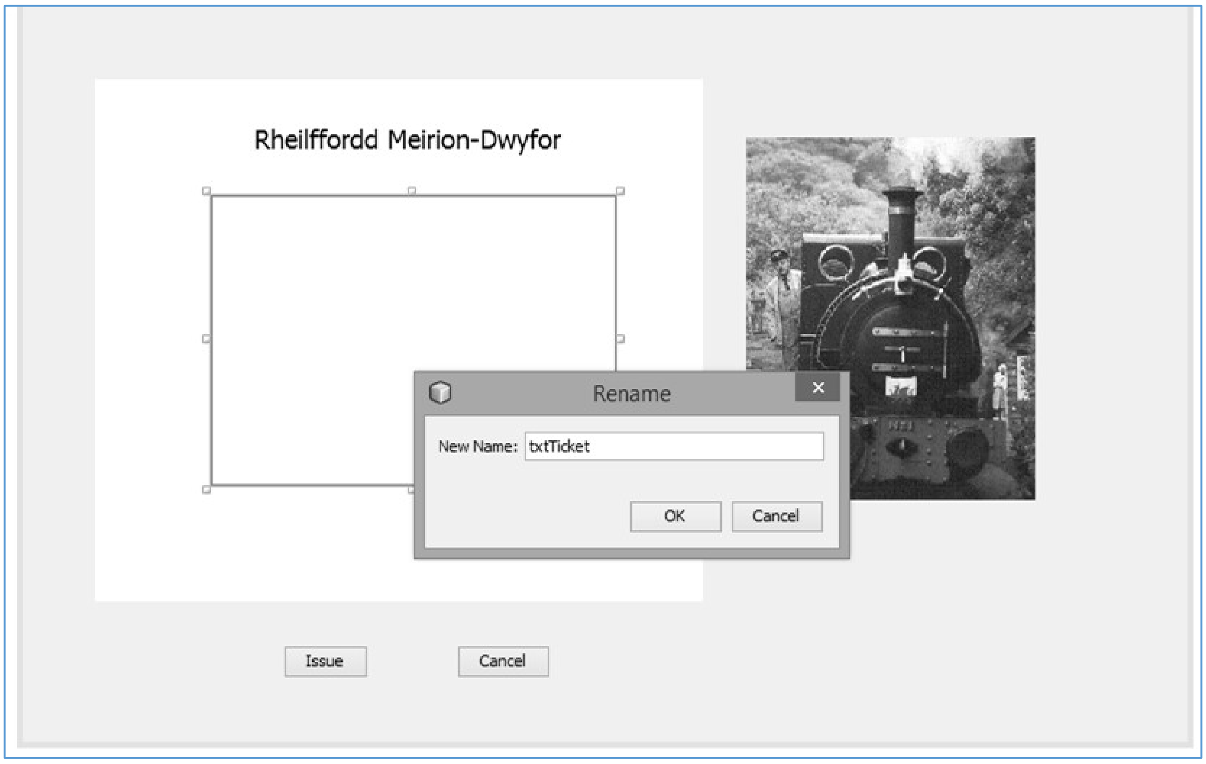
Go to the Palette and locate the ***Panel*** component. Drag and drop a Panel to the left of the picture. Use the ***background*** property to set the colour of the panel to white.

Drag and drop a label on the panel. You will find that the panel operates independently of the form, so it is necessary to right-click on the ***panel*** and again set an ***Absolute Layout***:



Set the caption of the label to display the name of the railway, using a suitable font size.

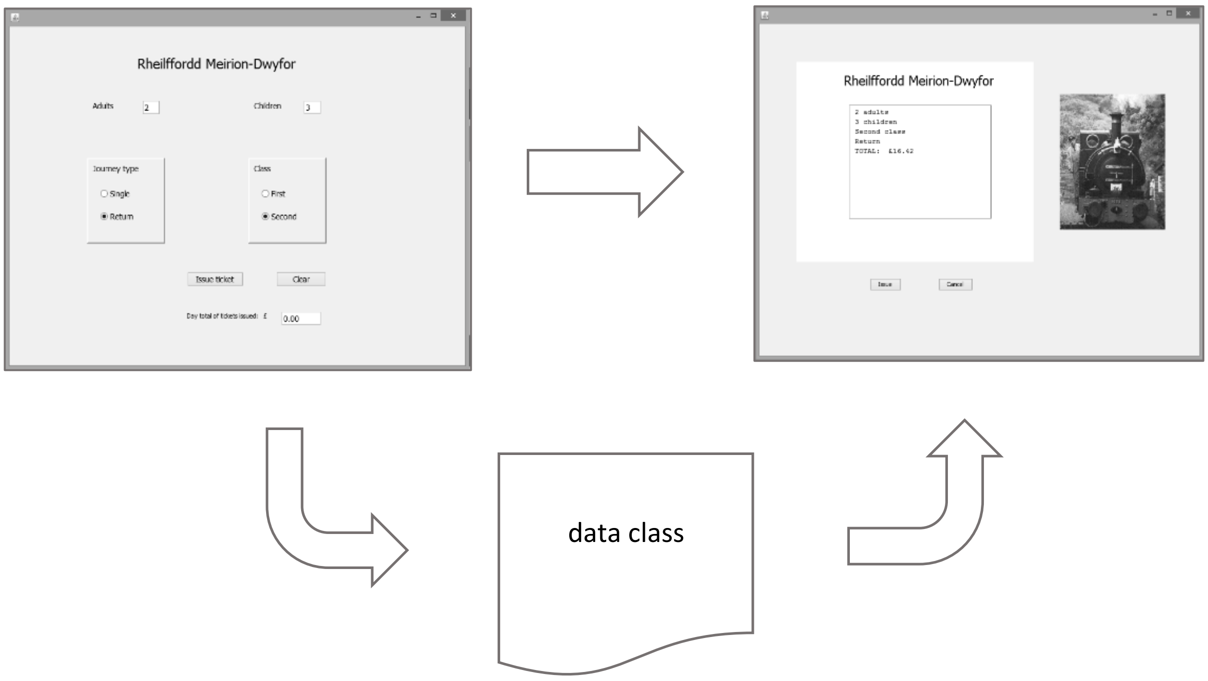
Locate the ***Text Area*** component in the palette. Drag and drop a ***text area*** on the ***panel***, and reset its variable name to ***txtTicket***. Also add ***buttons*** with the captions "***Issue***" and "***Clear***". Rename these as ***btnIssue*** and ***btnClear***:



The panel will be used to display the details of the tickets purchased. You will remember that the numbers of adults and children, the journey type and class were entered on the ***tickets*** form. This information needs to also be available on the ***displayTicket*** form.

Variables are not automatically carried forward from one form to another. The easiest way to transfer variables is by a two-stage process:

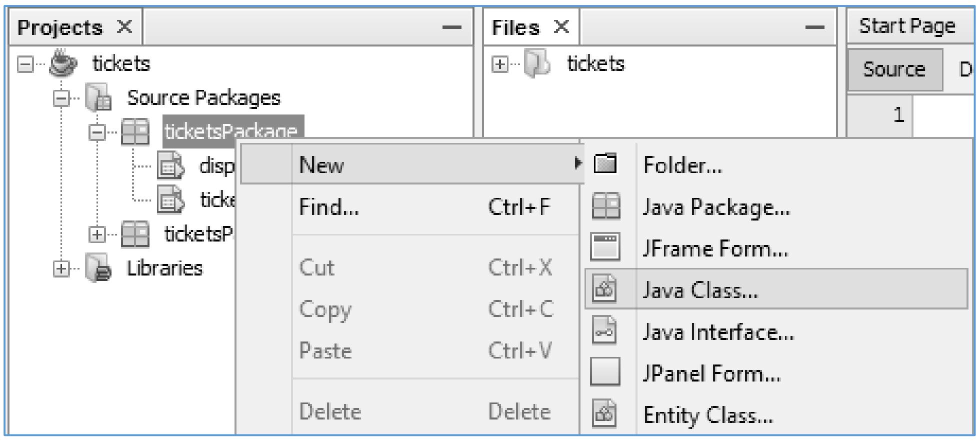
- Variable values from the first form are stored in a ***class*** file.
- When the second form is loaded, the variable values are retrieved from the class file:



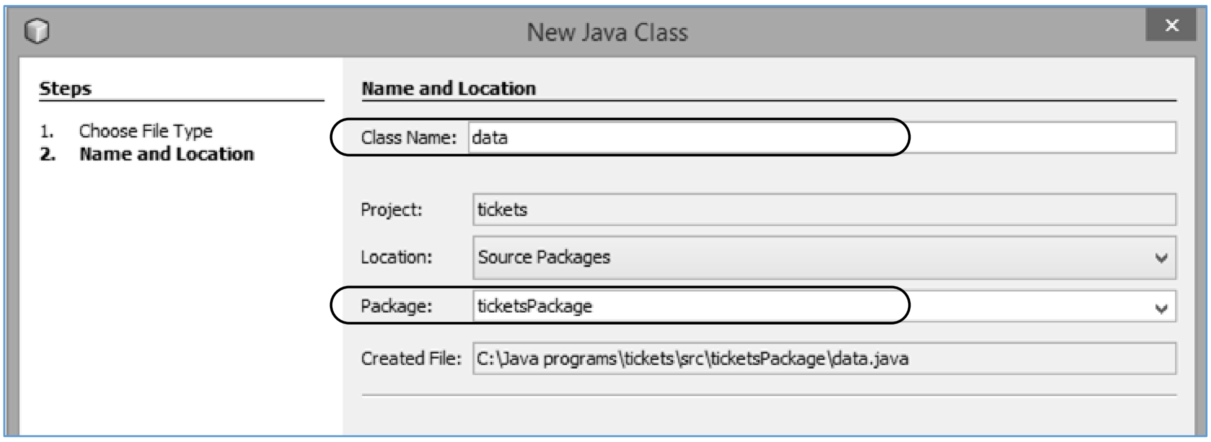
```
2 adults
3 children
Second class
Return
TOTAL:  £16.42
```

We will begin by setting up a ***class*** file for this purpose.

Right-click on ***ticketsPackage*** in the ***Projects*** window, then select ***New*** / ***Java Class***:



Give the ***class Name*** as ***data***, leaving the ***Package*** name as ***ticketsPackage***:



Class Name: data

Project: tickets

Location: Source Packages

Package: ticketsPackage

Created File: C:\Java programs\tickets\src\ticketsPackage\data.java

Open the ***Source*** code window for the ***data*** class. Add variable definitions:

```
package ticketsPackage;

public class data {

   public static int adult;
   public static int child;
   public static boolean returnWanted;
   public static boolean firstWanted;
   public static double ticketcost;
   public static double dayTotal=0;

}
```

The keyword ***public*** indicates that the variables should be accessible from other forms in the project, and the keyword ***static*** indicates that only one instance of each variable will exist.

Move now to the tickets form and locate the "***Issue ticket***" button click method. Add lines of code to store the variable values in the ***data*** class before the ***displayTicket*** form opens:

```
        if ((adult + child) >=4)
        {
           ticketcost = ticketcost * 0.90;
        }

        data.adult=adult;
        data.child=child;
        data.returnWanted=returnWanted;
        data.firstWanted=firstWanted;
        data.ticketcost=ticketcost;

        if (ticketcost>0)
        {
             new displayTicket().setVisible(true);
        }
  }
```

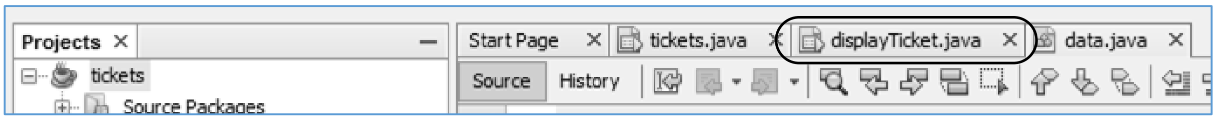
Notice the ***dot notation***:

data.adult

refers to the variable called ***adult*** in the class called ***data***.

Another error trapping feature has been also added. The ***displayTicket*** form will not be opened if the ***Issue ticket*** button is clicked with no passenger data entered.

Use the tab above the code window to move to the ***displayTicket.java*** page



Click the ***Source*** tab to display the program code screen. Locate the ***displayTicket( )*** method near the start of the program. This is the first method to run when the form opens.

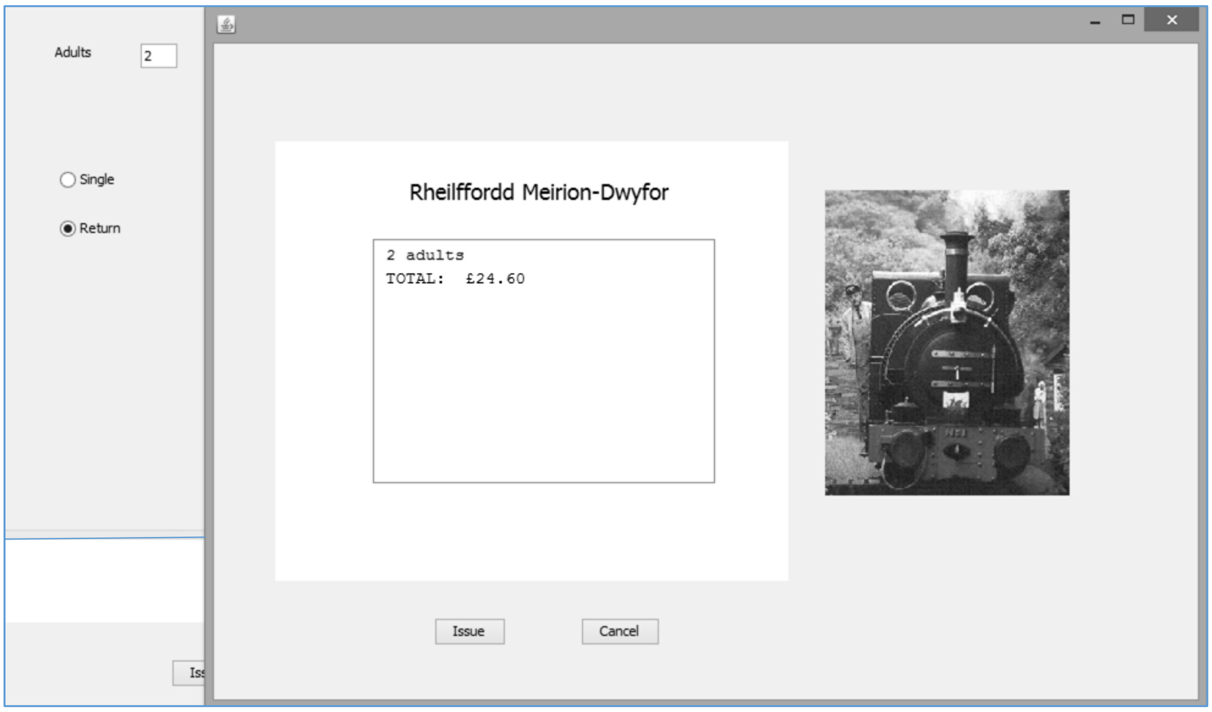
Add lines of code to collect the variable values form the ***data*** class:

```
package ticketPackage;

public class displayTicket extends javax.swing.JFrame {

    public displayTicket() {
        initComponents();

        int adult = data.adult;
        int child = data.child;
        boolean firstWanted=data.firstWanted;
        boolean returnWanted=data.returnWanted;
        double ticketcost=data.ticketcost;

    }
```

Still within the ***displayTicket( )*** method, we will set up a string of text which can be displayed on the panel. This begins by indicating the number of adults travelling. Notice how we have used the singular "***adult***" if there is one passenger, but the plural "***adults***" if there is more than one:

```
        boolean returnWanted=data.returnWanted;
        double ticketcost=data.ticketcost;

        String s="";
        if (adult>0)
        {
            if (adult==1)
            {
                s += " "+ Integer.toString(adult)+ " adult\n";
            }
            else
            {
                s += " "+ Integer.toString(adult)+ " adults\n";
            }
        }

        s += " TOTAL:  £"+ String.format("%.2f",ticketcost);
        txtTicket.setText(s);

    }
```

Run the program. Enter ticket details including one or more adults, then click the "***Issue ticket***" button. The second window should open, displaying the number of adults and the total ticket cost:



```
2 adults
TOTAL:  £24.60
```

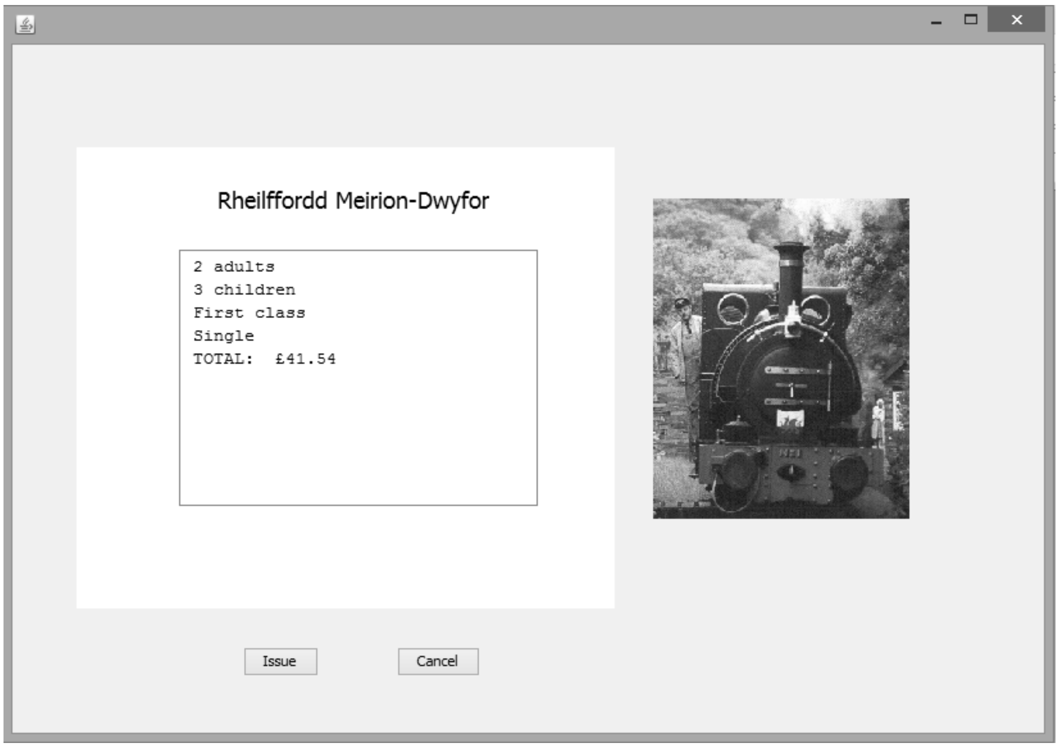
Close the program window and return to the Source code window. Add similar lines of program to display the number of children travelling. The code "***\n***" moves down onto a new line in the text area:

```
        if (adult>0)
        {
            if (adult==1)
            {
                s += " "+ Integer.toString(adult)+ " adult\n";
            }
            else
            {
                s += " "+ Integer.toString(adult)+ " adults\n";
            }
        }
        if (child>0)
        {
            if (child==1)
            {
                s += " "+ Integer.toString(child)+ " child\n";
            }
            else
            {
                s += " "+ Integer.toString(child)+ " children\n";
            }
        }
        s += " TOTAL:  £"+ String.format("%.2f",ticketcost);
        txtTicket.setText(s);
    }
```

Finally, add sections of code to output whether the ticket is ***First class*** or ***Second class***, ***Single*** or ***Return***:

```
        if (child>0)
        {
            if (child==1)
            {
                s += " "+ Integer.toString(child)+ " child\n";
            }
            else
            {
                s += " "+ Integer.toString(child)+ " children\n";
            }
        }

        if(firstWanted==true)
        {
            s += " First class\n";
        }
        else
        {
            s += " Second class\n";
        }

        if(returnWanted==true)
        {
            s += " Return\n";
        }
        else
        {
            s += " Single\n";
        }

        s += " TOTAL:  £"+ String.format("%.2f",ticketcost);
        txtTicket.setText(s);
    }
```

Run the program. Check the ticket display for a variety of groups of passengers with different journey requirements:



```
2 adults
3 children
First class
Single
TOTAL:  £41.54
```

Close the program window. Select the ***Design*** window for the ***displayTicket*** form.

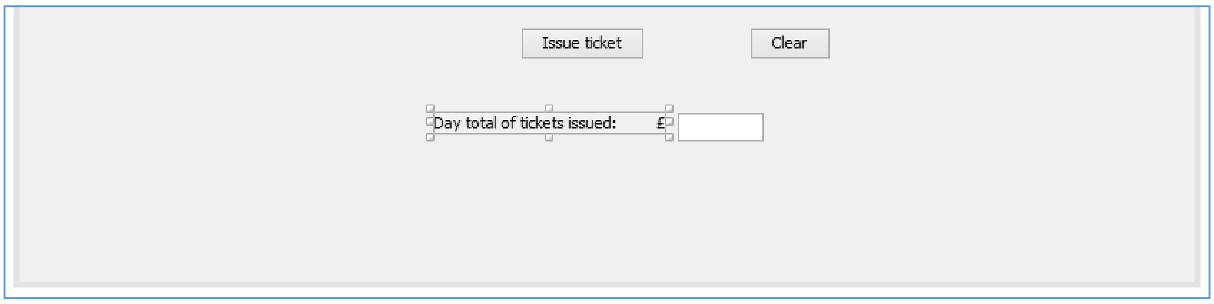
One further requirement in the specification was to keep a running total of the tickets issued. Double click the Issue button to create a button click method. Add line of codes to add the cost of the current ticket to the ***dayTotal*** variable in the ***data*** class, then close the ***displayTicket*** window:

```
private void btnIssueActionPerformed(java.awt.event.ActionEvent evt) {
    data.dayTotal += data.ticketcost;
    this.setVisible(false);
}
```

Return to the ***Design*** window and double click the ***Cancel*** button. In this case, we will not update the day total of tickets sold, but simply close the ***ticketDisplay*** window:

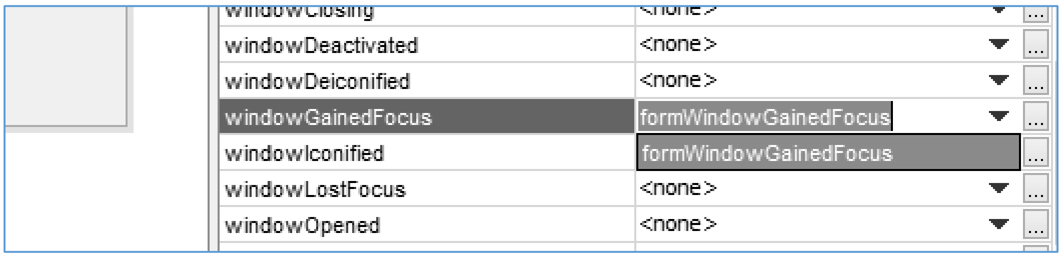
```
private void btnCancelActionPerformed(java.awt.event.ActionEvent evt) {
    this.setVisible(false);
}
```

Move now to the ***Design*** window for the ***tickets*** form. Edit the label text at the bottom of the form to read "***Day total of tickets issued: £***":



We will now create a method which will operate each time the user moves back from the ***ticketDisplay*** form to the ***tickets*** input form. Go to the ***Properties*** window for the ***tickets*** form and select the ***Events*** tab.

Locate the ***windowGainedFocus*** property, and click on ***formWindowGainedFocus*** in the drop-down list:

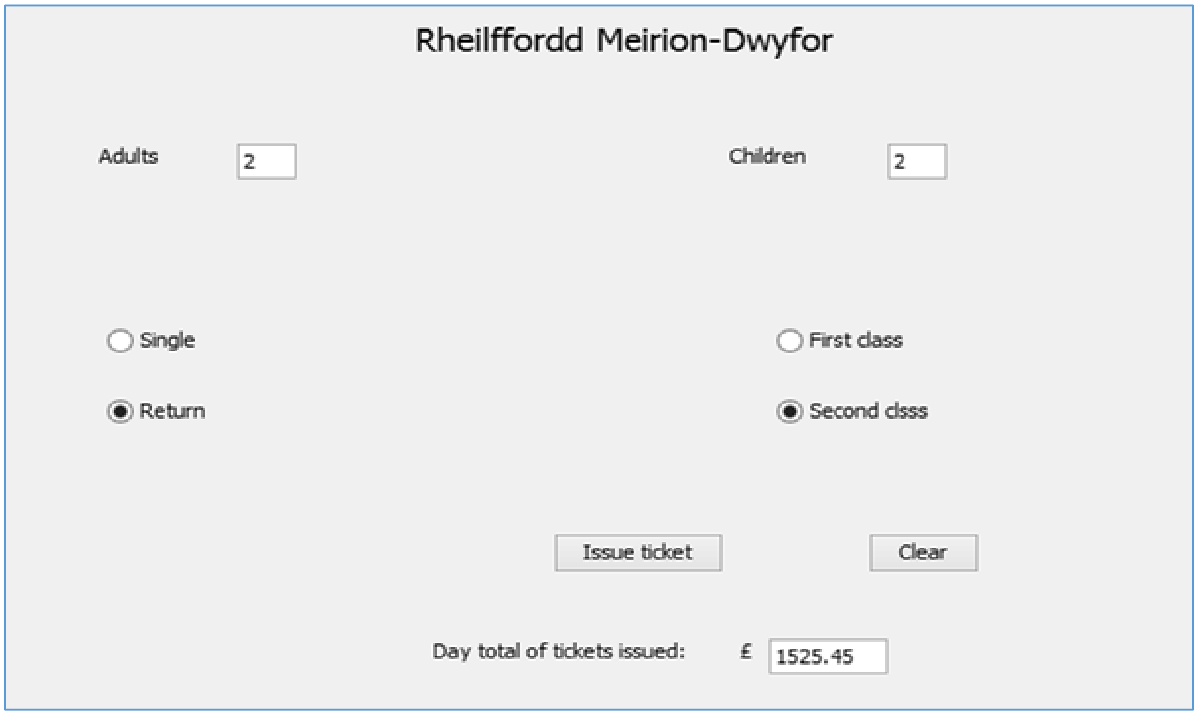


windowClosing	<none>
windowDeactivated	<none>
windowDeiconified	<none>
windowGainedFocus	formWindowGainedFocus
windowIconified	formWindowGainedFocus
windowLostFocus	<none>
windowOpened	<none>

Add lines of code to the ***formWindowGainedFocus*** method to display the day total. Notice that the dot convention is used to access the value of the variable ***dayTotal*** from the ***data*** class:

```
private void formWindowGainedFocus(java.awt.event.WindowEvent evt) {
    String s = String.format("%.2f",data.dayTotal);
    txtTotal.setText(s);
}
```

Run the program. Issue a number of tickets and check that the day total is maintained correctly as each ticket is added. You may spot one problem: if the ***Clear*** button is clicked, then the ***Day total*** box is cleared, which should not happen. We will correct this now.



Close the program window and return to the NetBeans editing screen. Click the tab to move to the ***tickets.java*** page, then click the ***Source*** tab to display the program code. Locate the ***btnClear( )*** method. Remove the line which clears the ***txtTotal*** text field.

```
private void btnClearActionPerformed(java.awt.event.ActionEvent evt) {
    txtAdult.setText("");
    txtChild.setText("");
    txtTotal.setText("");    DELETE THIS LINE
  }
```

Run the program. Issue several tickets, then click the ***Clear*** button. The ***Day total*** should not now be deleted.