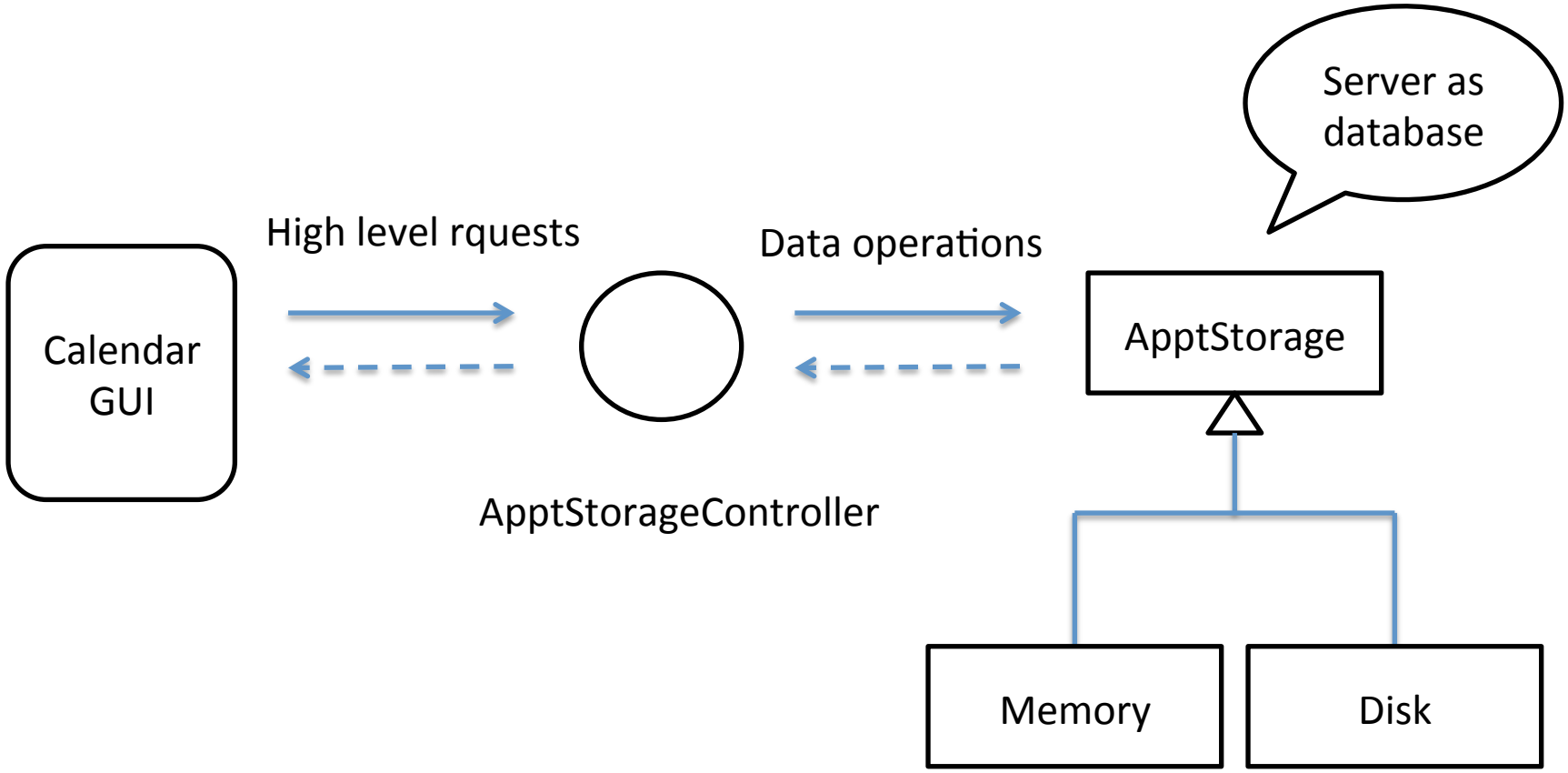
The overall calendar architecture



Important classes

- ApptStorageController and ApptStorage
- Partition of high level functions and low level data operations
- Only a suggested partition using the interface-based programming

ApptStorageController

- Communicate with GUI with high level functions
- Define “business functions”.
  - Return the earliest event from today
  - Return all events of the current week
- Feel free to add your own

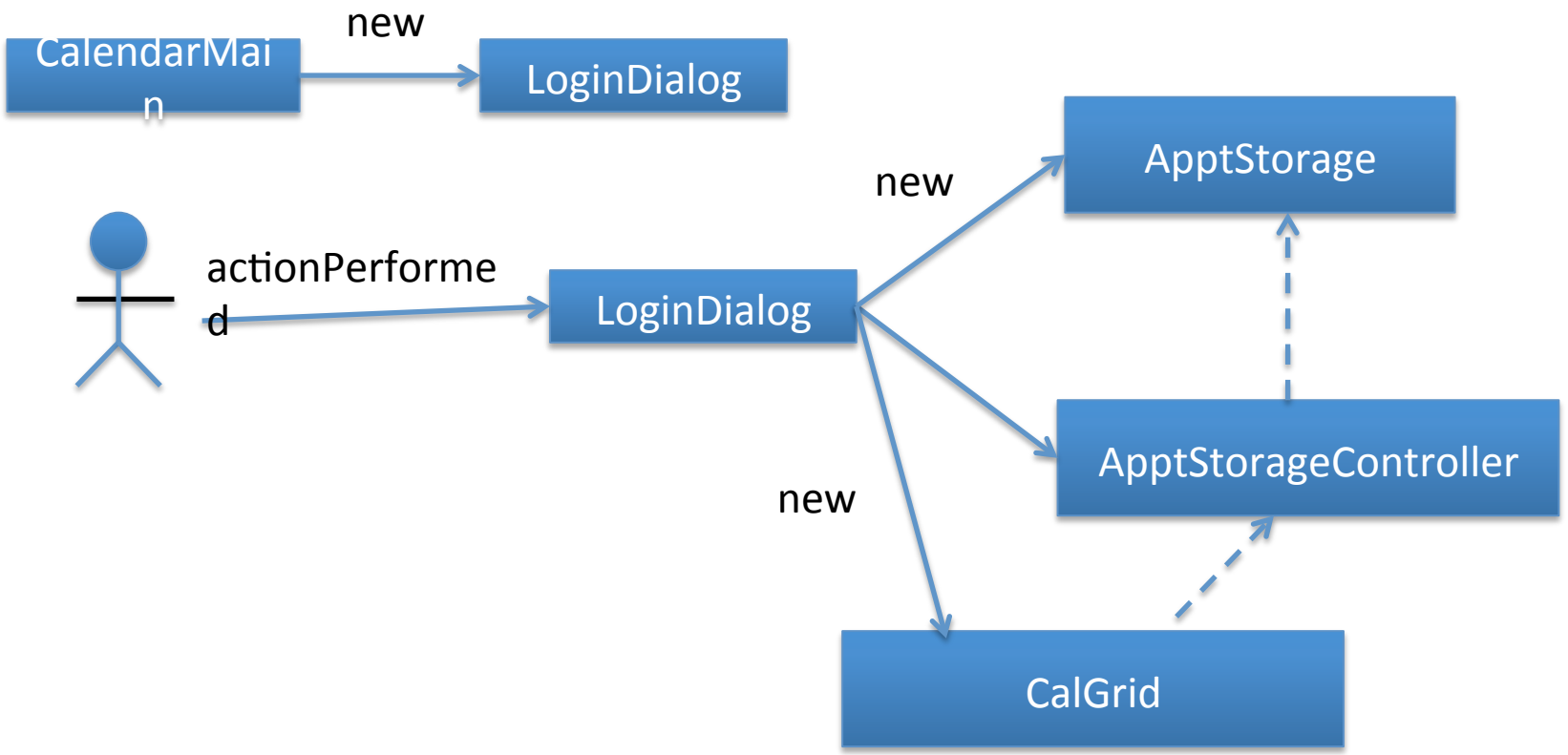
ApptStorage

- An abstract class
- Communicate with ApptStorageController on low-level data operations
- Define database like operations
  - Store event
  - Store user
  - find event
  - find user
- The implementation of the abstract class
  - Memory-based : Use Java collection classes
  - Disk-based: Use object storage classes (Phase II)
  - Database-based: Use JDBC interface (Bonus)

Partition of Tasks

- ApptStorageController
  - Return the earliest event of today
    - Ask ApptStorage to return all events
    - Sort the events according to time
    - Return the first item
  - Store event (Wait! This is a data operation)
    - Check if the event is a valid event, such as happens in the future
- Advantage
  - GUI code is simplified
  - Wrong event information → something wrong with ApptStorage
  - Wrong event → Sorting algorithm

How is everything connected?



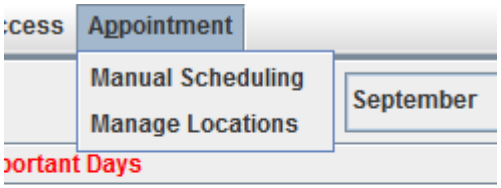
Requirement

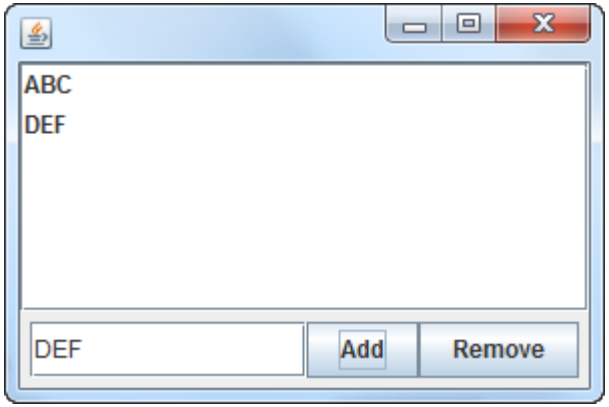
Location information:
The location must be uniquely identified by its name. The user must select from a set of predefined locations. The locations can be added through a separate interface (Note that this interface does not exist in the current skeleton).

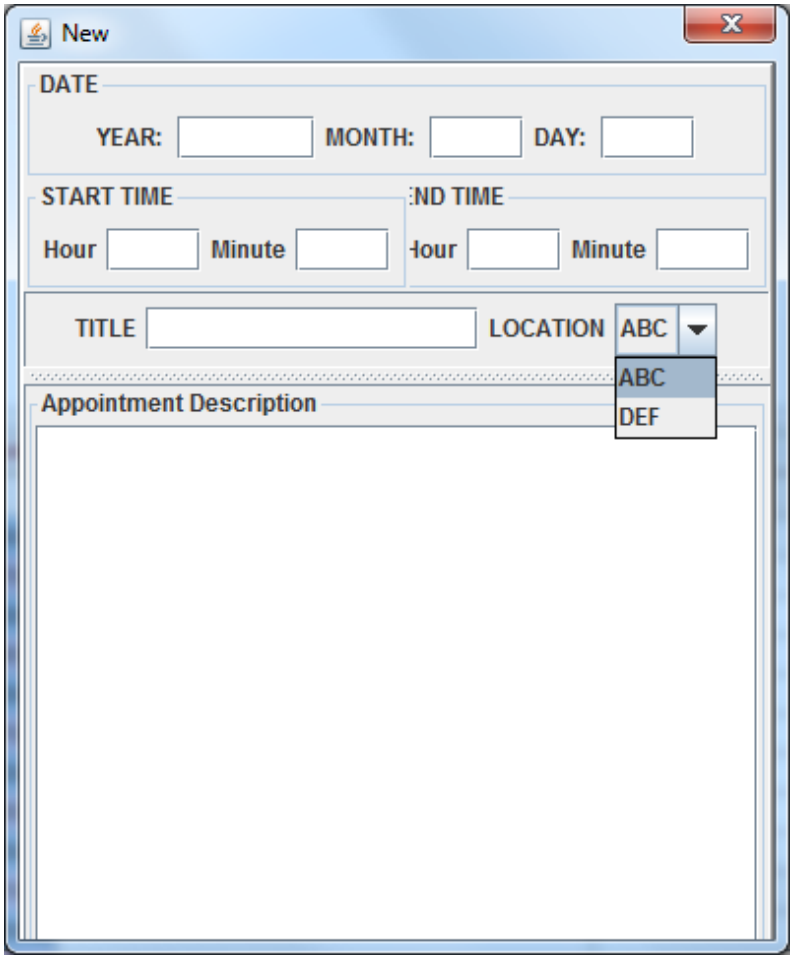
Tasks

- Need to define new classes to store location information
- Need to add a new window to input location information
- Need to add a menu item to start the new window
- Need to add a list on the event window for choosing locations.

Objective


ccess
Appointment
Manual Scheduling
Manage Locations
September
portant Days

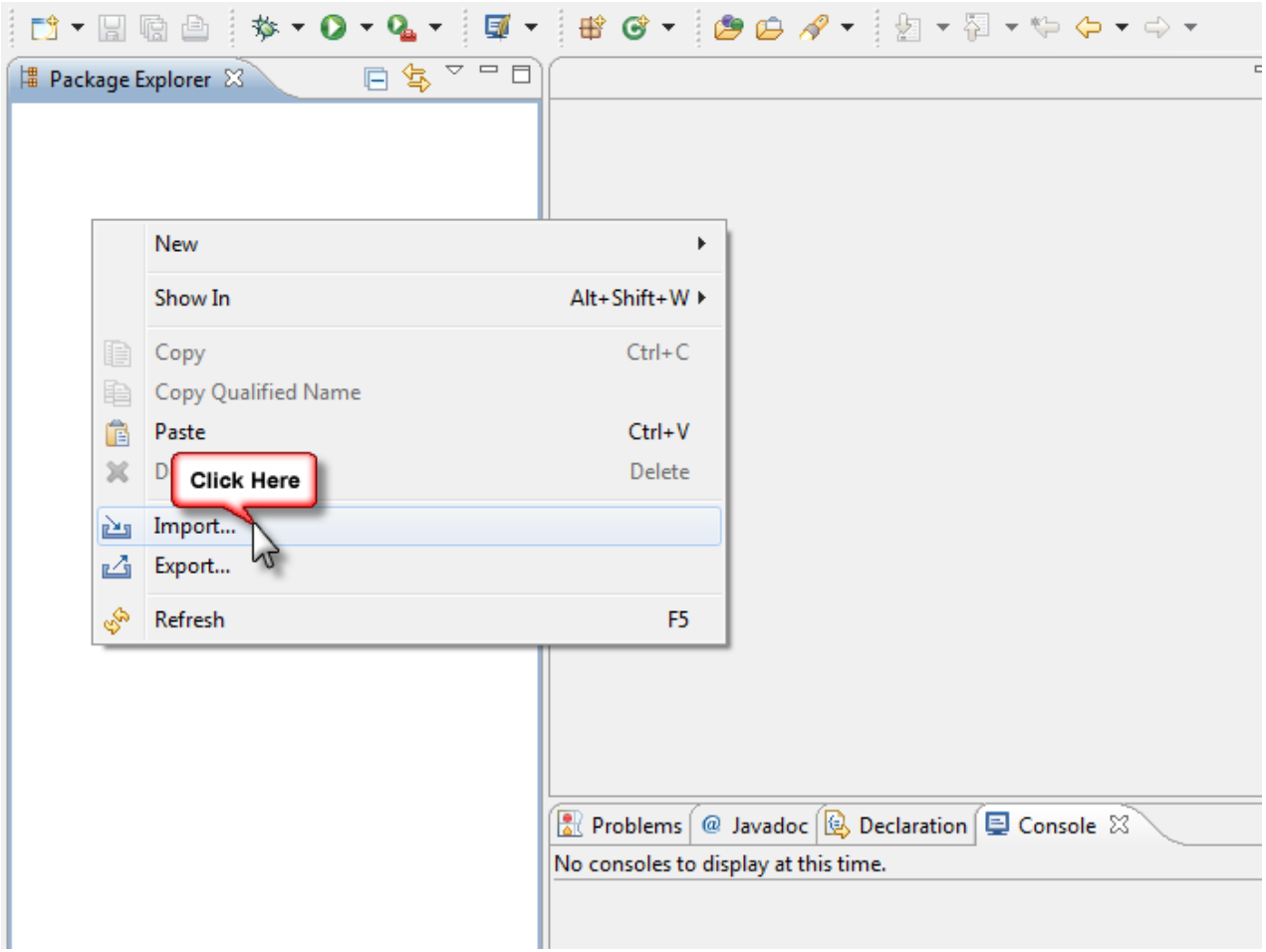

ABC
DEF
DEF
Add
Remove


New
DATE
YEAR:
MONTH:
DAY:
START TIME
Hour
Minute
:ND TIME
Hour
Minute
TITLE
LOCATION
ABC
ABC
DEF
Appointment Description

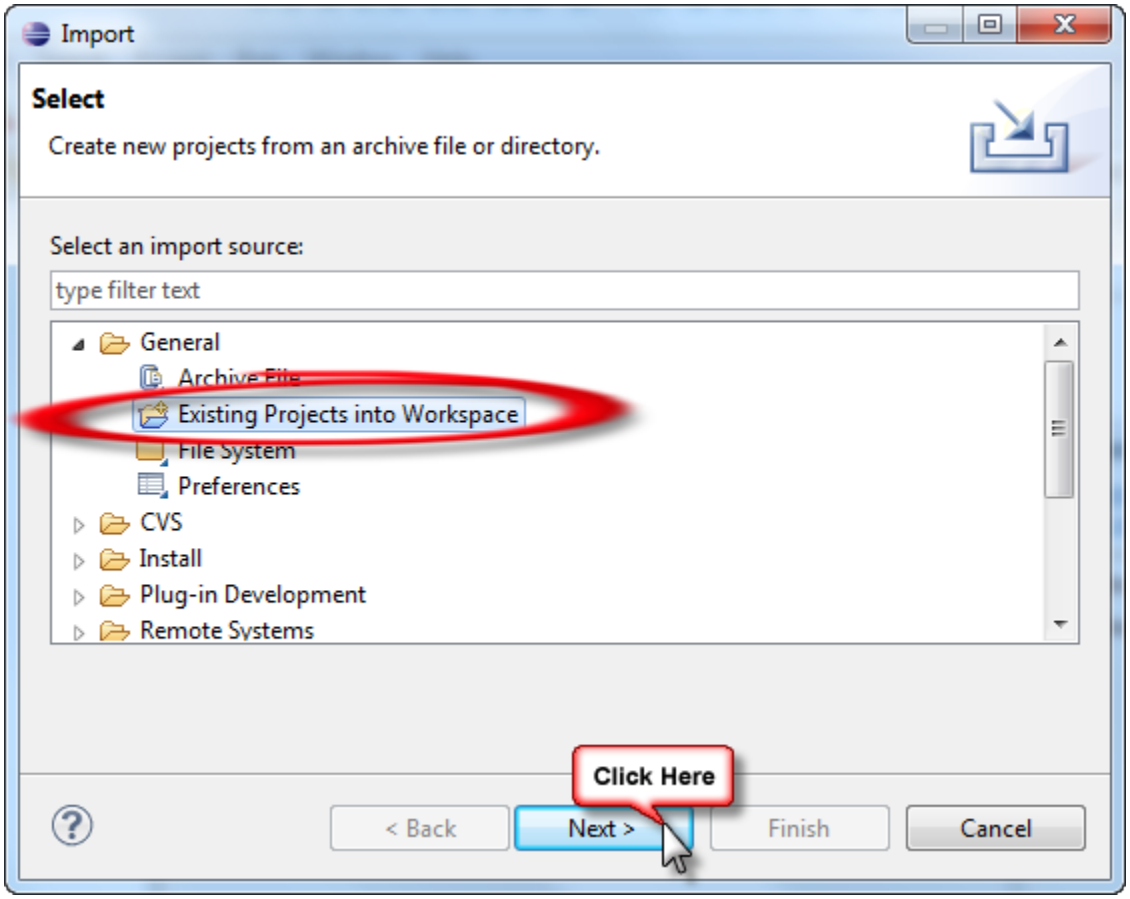
Getting Started

- Download and start Eclipse SDK
  - http://www.eclipse.org/downloads/
  - Should already installed on the lab computers
- Download the base code
  - http://course.cse.ust.hk/comp3111/project/base.jar
- Import the base code

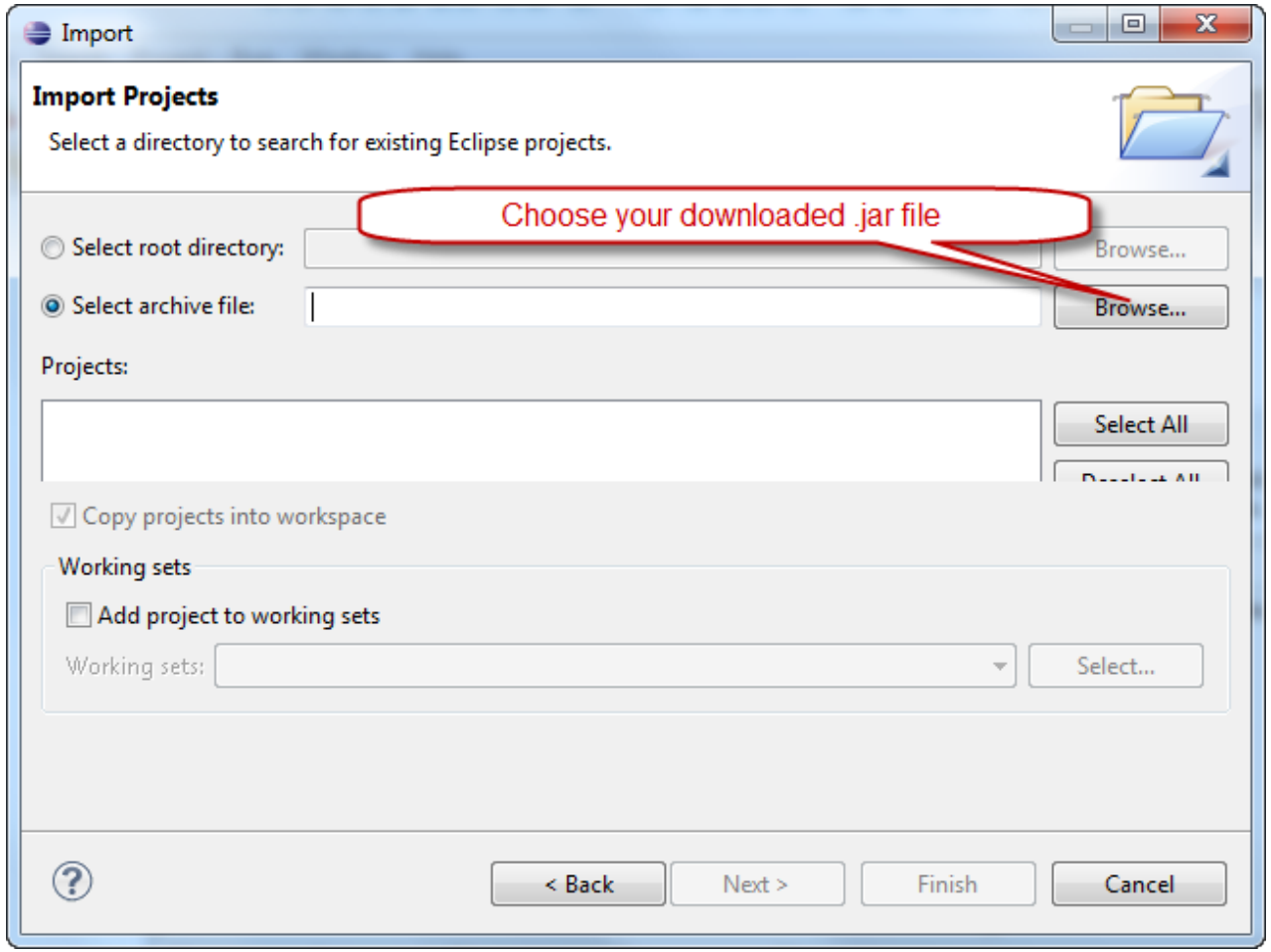
Import the base code - 1



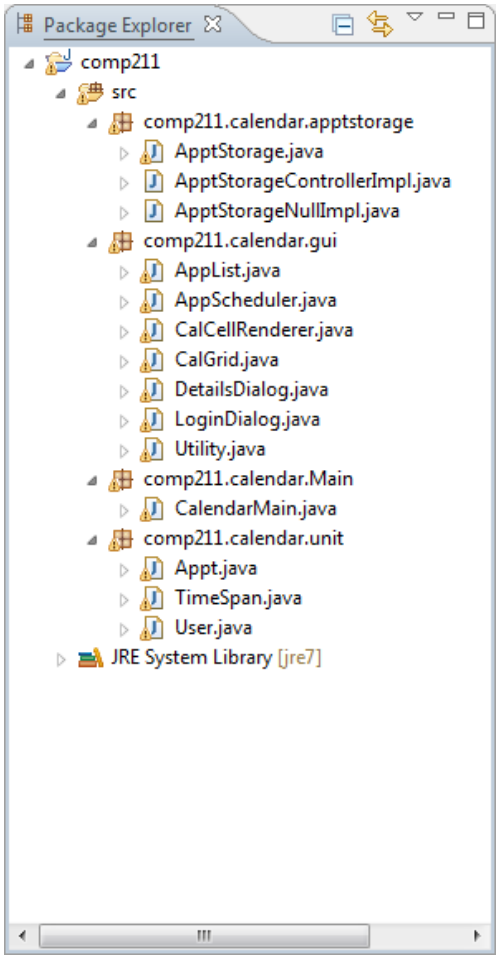
Import the base code - 2



Import the base code

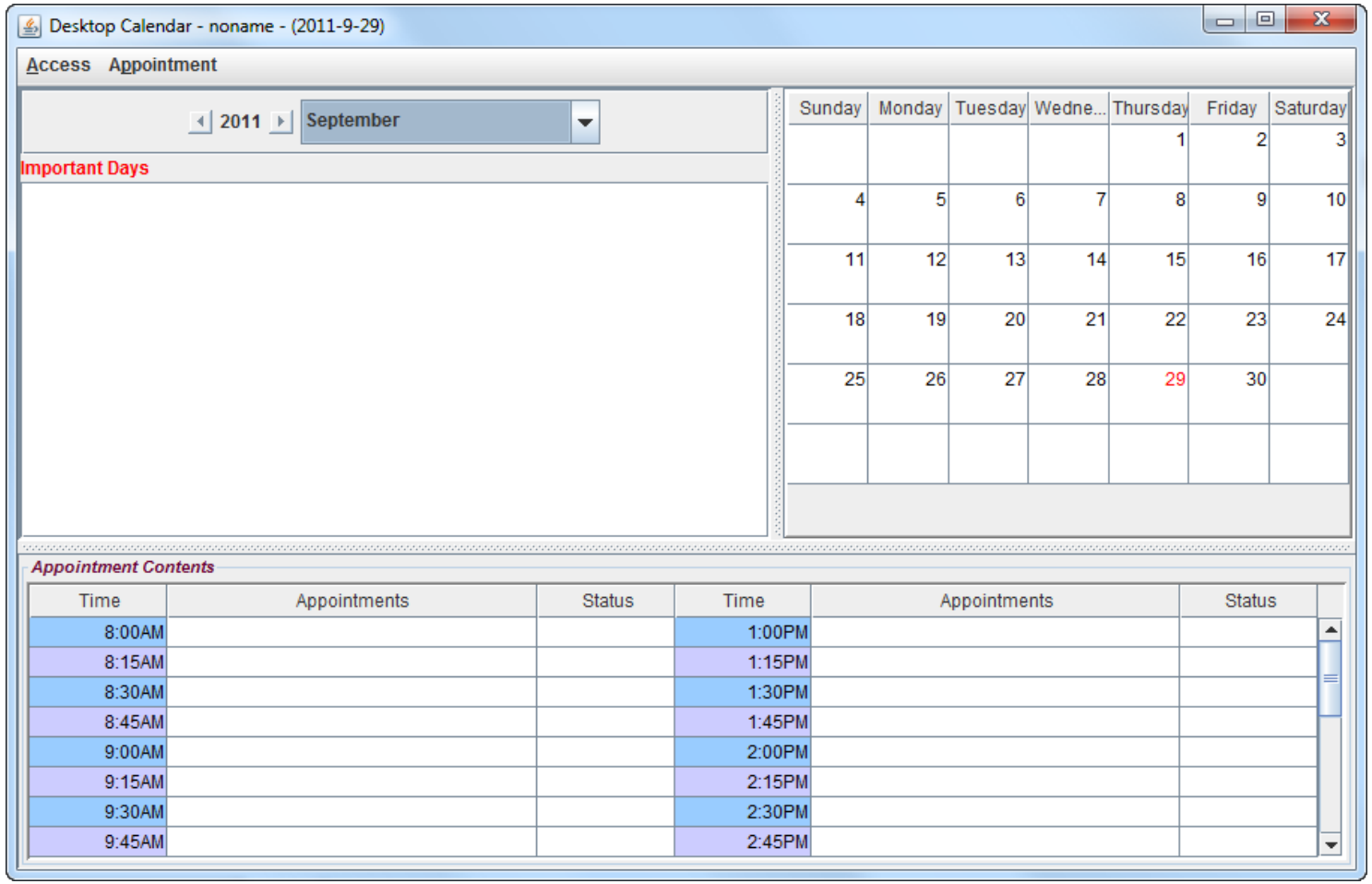


Code structure



- comp211.calendar.appstorage
  - Data storage implementation
- comp211.calendars.gui
  - GUI elements
- comp211.calendars.Main
  - Main class
- comp211.calendar.unit
  - Some data structures

Run the application

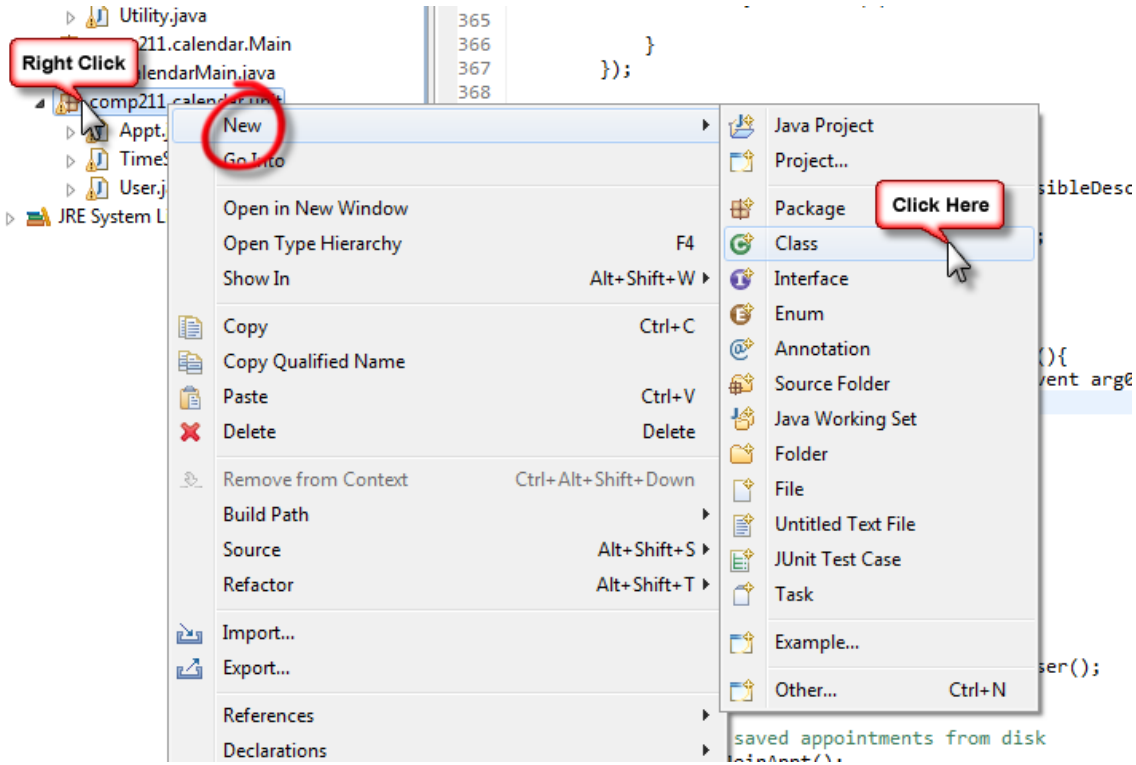


Implementing a Feature

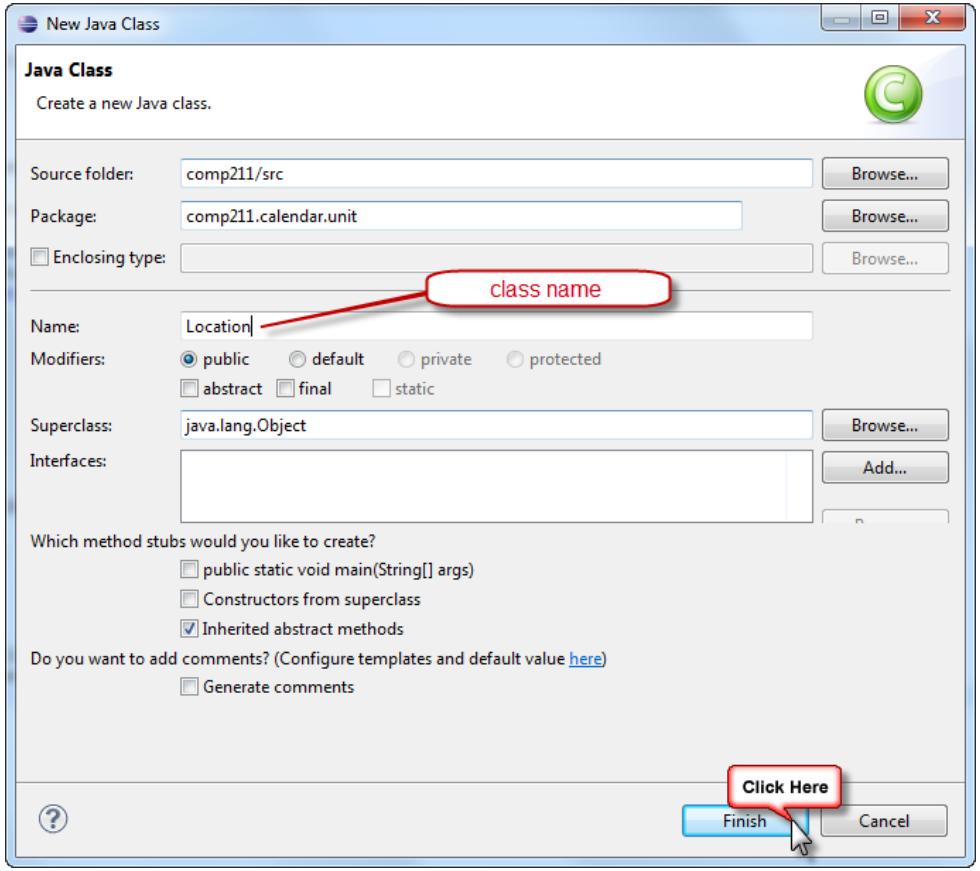
- Add location information for event scheduling
  - Add a data structure to store the location
  - Implement the data storage for saving location data
  - Cate a dialog for manage locations
  - Create a menu entry for displaying the dialog
  - In the event creating GUI, Create a GUI element for choosing the location

Implementing a Feature

- Add location information for event scheduling
  - Add a data structure to store the location
  - Implement the data storage for saving location data
  - Cate a dialog for manage locations
  - Create a menu entry for displaying the dialog
  - In the event creating GUI, Create a GUI element for choosing the location

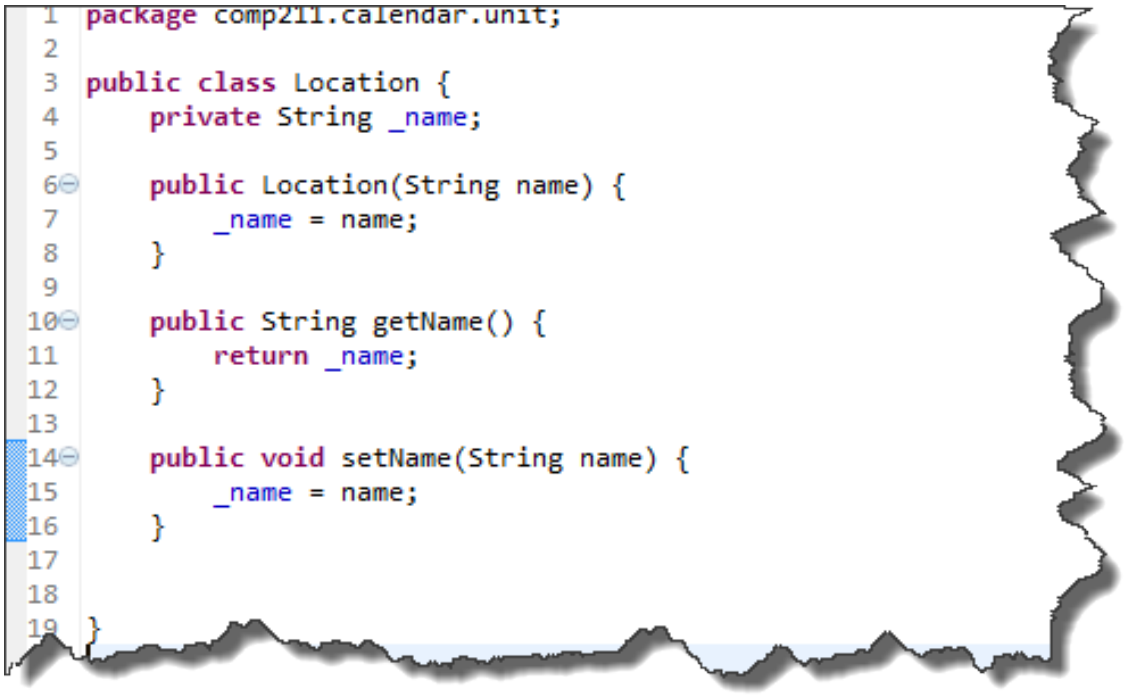


Create a data structure (new class)



New class

The name is **Location**

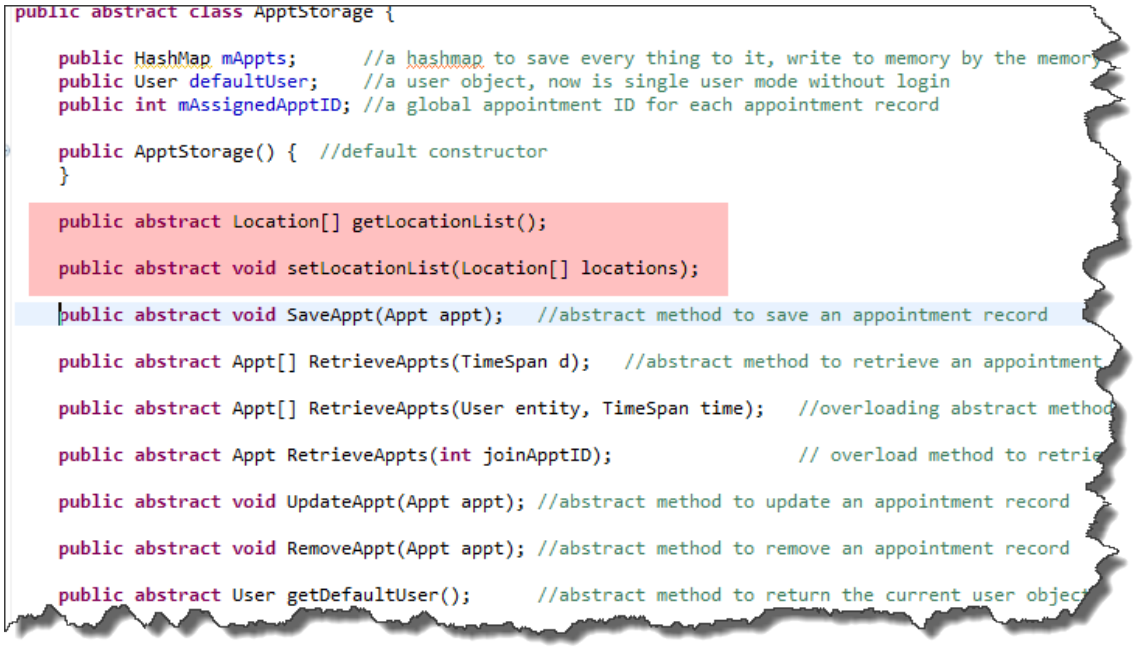


```
1  package comp211.calendar.unit;
2
3  public class Location {
4      private String _name;
5
6      public Location(String name) {
7          _name = name;
8      }
9
10     public String getName() {
11         return _name;
12     }
13
14     public void setName(String name) {
15         _name = name;
16     }
17
18
19 }
```

Add the code

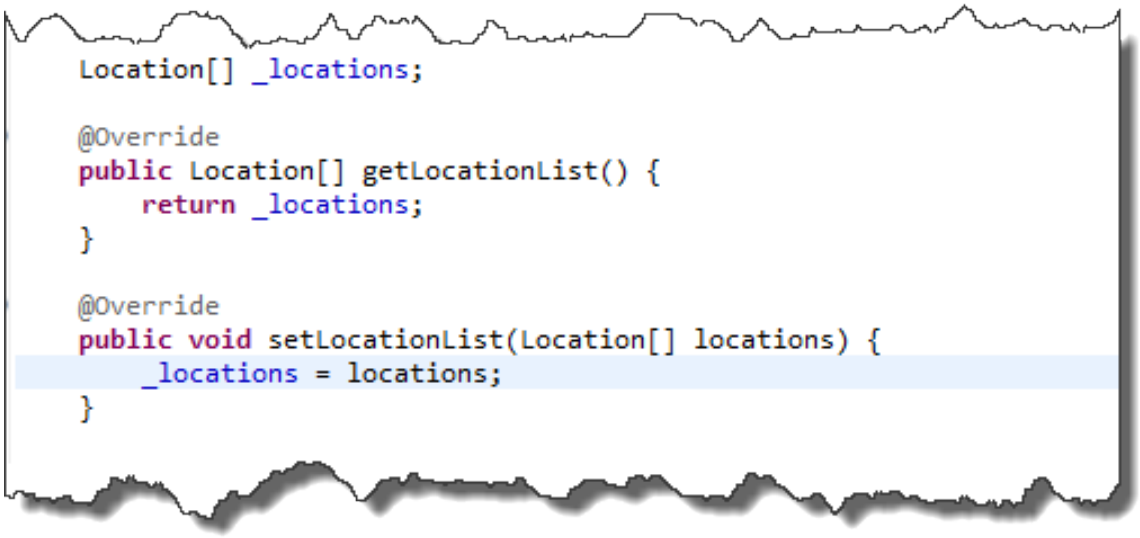
Implementing a Feature

- Add location information for event scheduling
  - Add a data structure to store the location
  - Implement the data storage for saving location data
  - Cate a dialog for manage locations
  - Create a menu entry for displaying the dialog
  - In the event creating GUI, Create a GUI element for choosing the location



```
public abstract class ApptStorage {

    public HashMap mAppts;      //a hashmap to save every thing to it, write to memory by the memory
    public User defaultUser;    //a user object, now is single user mode without login
    public int mAssignedApptID; //a global appointment ID for each appointment record

    public ApptStorage() {  //default constructor
    }

    public abstract Location[] getLocationList();

    public abstract void setLocationList(Location[] locations);

    public abstract void SaveAppt(Appt appt);   //abstract method to save an appointment record

    public abstract Appt[] RetrieveAppts(TimeSpan d);   //abstract method to retrieve an appointment

    public abstract Appt[] RetrieveAppts(User entity, TimeSpan time);   //overloading abstract method

    public abstract Appt RetrieveAppts(int joinApptID);                 // overload method to retrie

    public abstract void UpdateAppt(Appt appt); //abstract method to update an appointment record

    public abstract void RemoveAppt(Appt appt); //abstract method to remove an appointment record

    public abstract User getDefaultUser();      //abstract method to return the current user objec
```

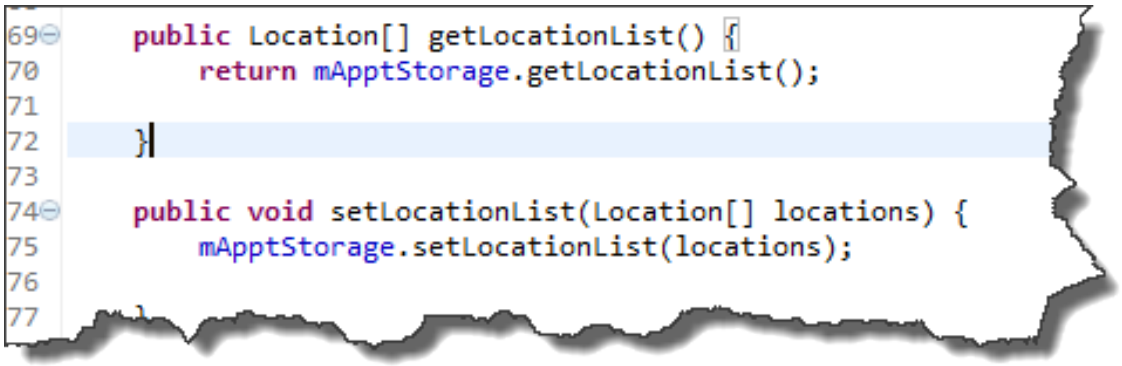
Add method to the ApptStorage class



```
Location[] _locations;

@Override
public Location[] getLocationList() {
    return _locations;
}

@Override
public void setLocationList(Location[] locations) {
    _locations = locations;
}
```

Add the implement the ApptStoragNullImpl

Change the class name if you don’t like the word “null”



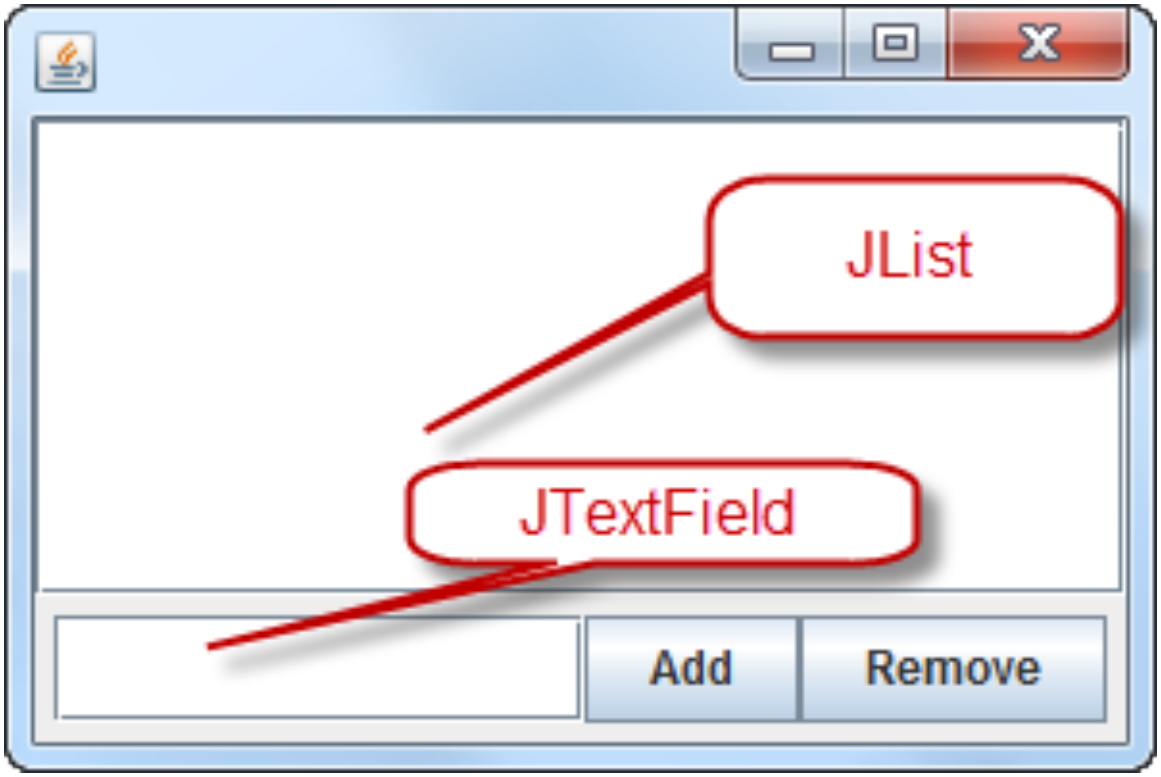
```
69      public Location[] getLocationList() {
70          return mApptStorage.getLocationList();
71
72      }
73
74      public void setLocationList(Location[] locations) {
75          mApptStorage.setLocationList(locations);
76
77
```

Also add new methods to the controller class

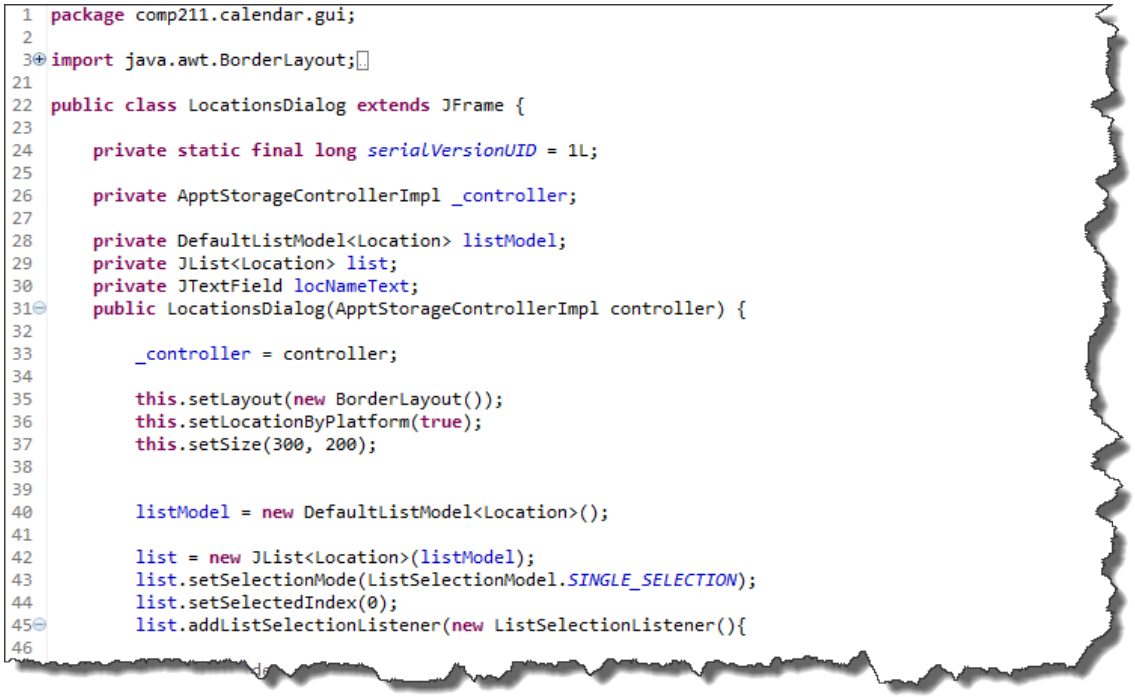
Route to the newly implemented methods

Implementing a Feature

- Add location information for event scheduling
  - Add a data structure to store the location
  - Implement the data storage for saving location data
  - Cate a dialog for manage locations
  - Create a menu entry for displaying the dialog
  - In the event creating GUI, Create a GUI element for choosing the location



We need a dialog looks like this



```
1  package comp211.calendar.gui;
2
3  import java.awt.BorderLayout;
21
22 public class LocationsDialog extends JFrame {
23
24     private static final long serialVersionUID = 1L;
25
26     private ApptStorageControllerImpl _controller;
27
28     private DefaultListModel<Location> listModel;
29     private JList<Location> list;
30     private JTextField locNameText;
31     public LocationsDialog(ApptStorageControllerImpl controller) {
32
33         _controller = controller;
34
35         this.setLayout(new BorderLayout());
36         this.setLocationByPlatform(true);
37         this.setSize(300, 200);
38
39
40         listModel = new DefaultListModel<Location>();
41
42         list = new JList<Location>(listModel);
43         list.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
44         list.setSelectedIndex(0);
45         list.addListSelectionListener(new ListSelectionListener(){
46
```

Create a new class LocationsDialog extending JFrame

Create the GUI

Manipulate the location data via calling the controller methods

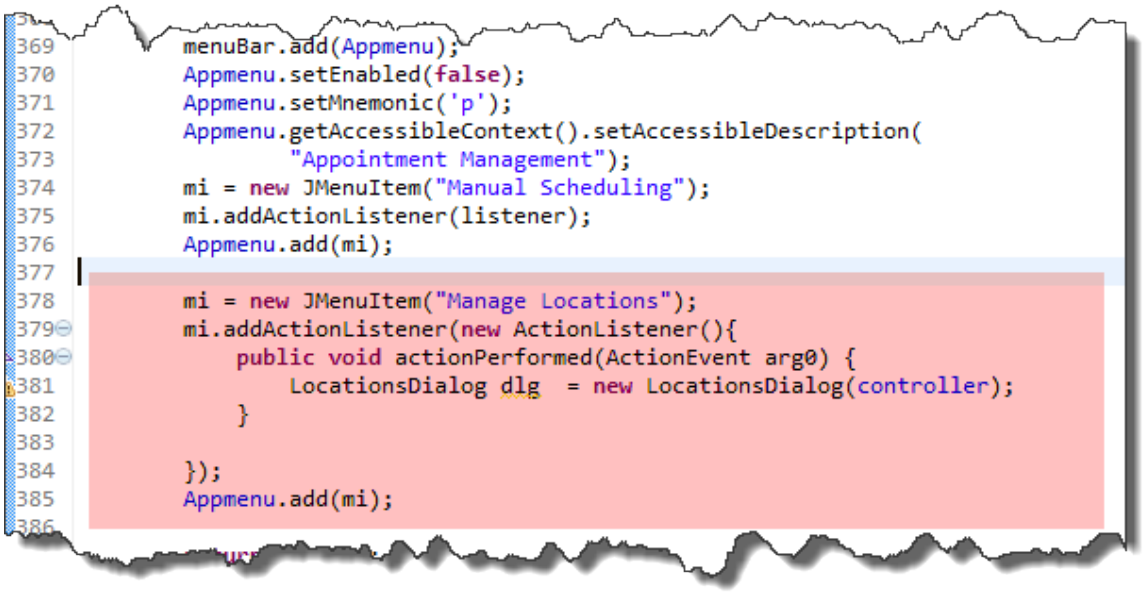
Implementing a Feature

- Add location information for event scheduling
  - Add a data structure to store the location
  - Implement the data storage for saving location data
  - Cate a dialog for manage locations
  - Create a menu entry for displaying the dialog
  - In the event creating GUI, Create a GUI element for choosing the location

```
JMenuBar createMenuBar() {

    ActionListener listener = new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            if (e.getActionCommand().equals("Manual Scheduling")) {
                AppScheduler a = new AppScheduler("New", CalGrid.this);
                a.updateSetApp(comp211.calendar.gui.Utility
                        .createDefaultAppt(currentY, currentM, currentD,
                                mCurrUser));
                a.setLocationRelativeTo(null);
                a.show();
                TableModel t = prepareTableModel();
                tableView.setModel(t);
                tableView.repaint();
            }

        }
    };
    JMenuBar menuBar = new JMenuBar();
    menuBar.getAccessibleContext().setAccessibleName("Calendar Choices");
```

Find the code for initializing the menu bar

- Open **CalGrid.java** in **comp211.calendar.gui**
- Navigate to **createMenuBar()** method.



```
369         menuBar.add(Appmenu);
370         Appmenu.setEnabled(false);
371         Appmenu.setMnemonic('p');
372         Appmenu.getAccessibleContext().setAccessibleDescription(
373                 "Appointment Management");
374         mi = new JMenuItem("Manual Scheduling");
375         mi.addActionListener(listener);
376         Appmenu.add(mi);
377
378         mi = new JMenuItem("Manage Locations");
379         mi.addActionListener(new ActionListener(){
380             public void actionPerformed(ActionEvent arg0) {
381                 LocationsDialog dlg  = new LocationsDialog(controller);
382             }
383
384         });
385         Appmenu.add(mi);
386
```

Add a new menu item under “Appointment Management”

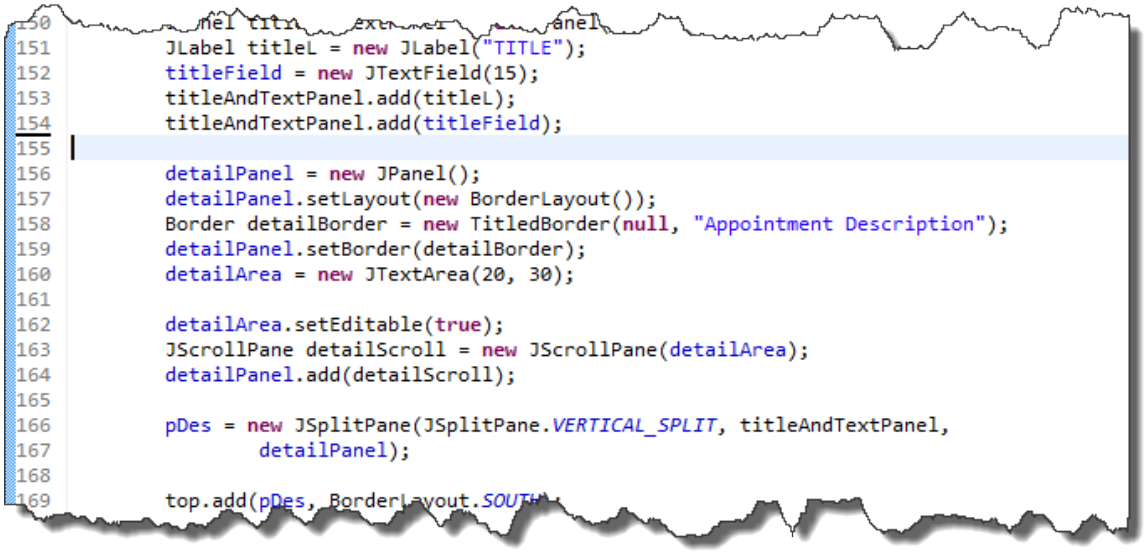
Related code is at the end of the slides

For creating menu entry

```
public class CalGrid extends JFrame implements ActionListener {
    …
    JMenuBar createMenuBar() {
        …
        mi = new JMenuItem("Manual Scheduling");
        mi.addActionListener(listener);
        Appmenu.add(mi);
        mi = new JMenuItem("Manage Locations");
        mi.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent arg0) {
                LocationsDialog dlg  = new LocationsDialog(controller);
            }
        });
        Appmenu.add(mi);
        return menuBar;
    }
    …
}
```

Implementing a Feature

- Add location information for event scheduling
  - Add a data structure to store the location
  - Implement the data storage for saving location data
  - Cate a dialog for manage locations
  - Create a menu entry for displaying the dialog
  - In the event creating GUI, Create a GUI element for choosing the location



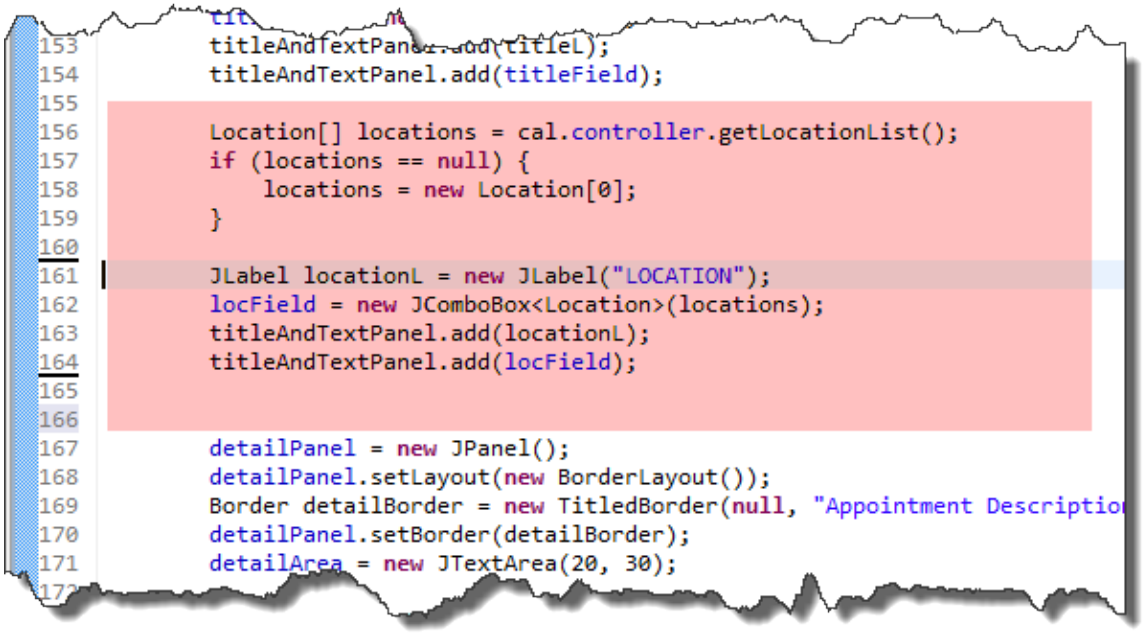
```
150         panel titlePanel = new JPanel();
151         JLabel titleL = new JLabel("TITLE");
152         titleField = new JTextField(15);
153         titleAndTextPanel.add(titleL);
154         titleAndTextPanel.add(titleField);
155
156         detailPanel = new JPanel();
157         detailPanel.setLayout(new BorderLayout());
158         Border detailBorder = new TitledBorder(null, "Appointment Description");
159         detailPanel.setBorder(detailBorder);
160         detailArea = new JTextArea(20, 30);
161
162         detailArea.setEditable(true);
163         JScrollPane detailScroll = new JScrollPane(detailArea);
164         detailPanel.add(detailScroll);
165
166         pDes = new JSplitPane(JSplitPane.VERTICAL_SPLIT, titleAndTextPanel,
167                 detailPanel);
168
169         top.add(pDes, BorderLayout.SOUTH);
```

Find the code for creating the GUI of event scheduling

In the **DetailsDialog** class

For creating the combo box

```
public class AppScheduler extends JDialog implements ActionListener, ComponentListener {
    …
    private JComboBox locField;
    private void commonConstructor(String title, CalGrid cal) {
        …
        titleAndTextPanel.add(titleField);
        Location[] locations = cal.controller.getLocationList();
        if (locations == null) {
            locations = new Location[0];
        }
        JLabel locationL = new JLabel("LOCATION");
        locField = new JComboBox(locations);
        titleAndTextPanel.add(locationL);
        titleAndTextPanel.add(locField);
        detailPanel = new JPanel();
        …
    }
    …
}
```



```
153         titleAndTextPanel.add(titleL);
154         titleAndTextPanel.add(titleField);
155
156         Location[] locations = cal.controller.getLocationList();
157         if (locations == null) {
158             locations = new Location[0];
159         }
160
161         JLabel locationL = new JLabel("LOCATION");
162         locField = new JComboBox<Location>(locations);
163         titleAndTextPanel.add(locationL);
164         titleAndTextPanel.add(locField);
165
166
167         detailPanel = new JPanel();
168         detailPanel.setLayout(new BorderLayout());
169         Border detailBorder = new TitledBorder(null, "Appointment Descriptio
170         detailPanel.setBorder(detailBorder);
171         detailArea = new JTextArea(20, 30);
```

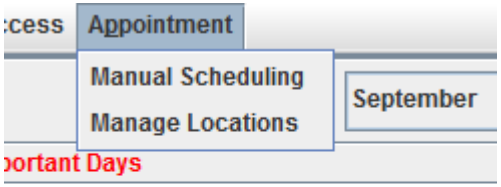
Create a combo box to select the location

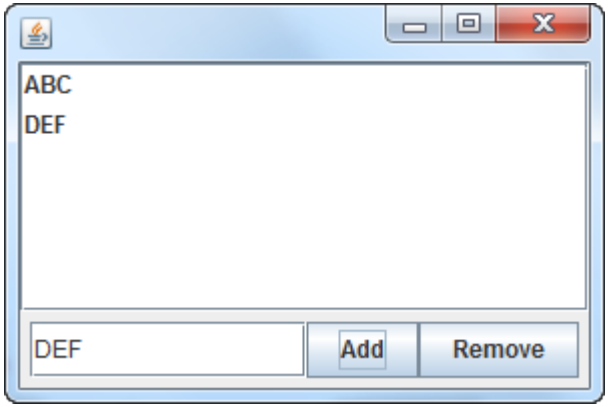
On the **AppScheduler** class

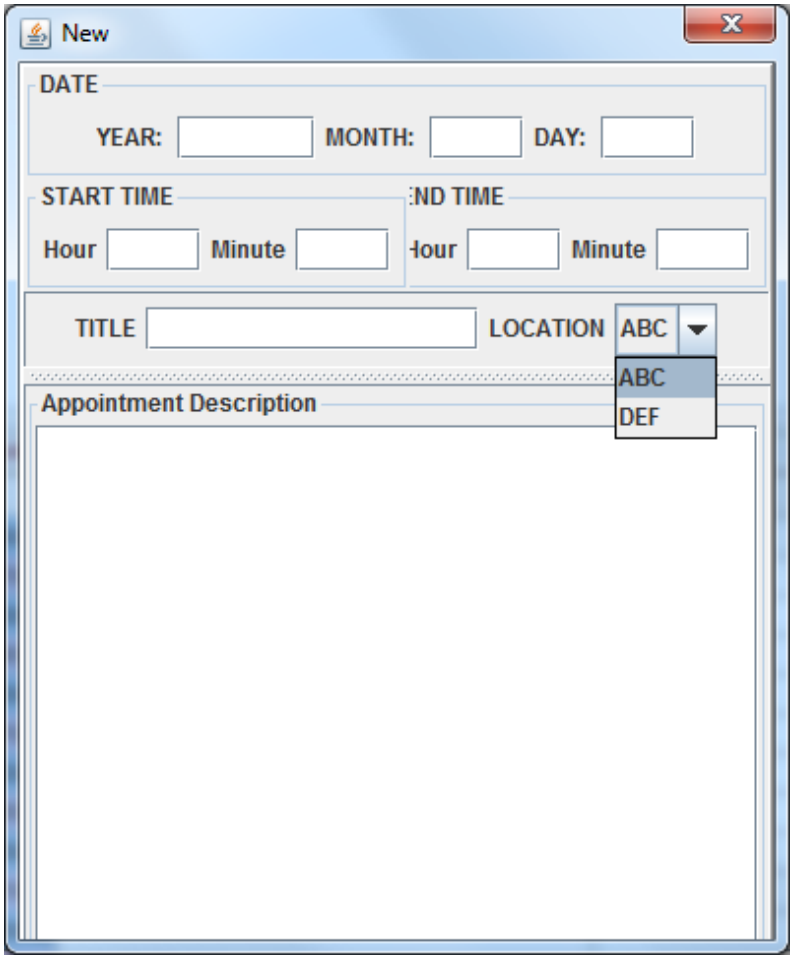
Here we add it next to the TITLE field, in the same panel.

Related code is at the end of the slides

We're Done


ccess
Appointment
Manual Scheduling
Manage Locations
September
portant Days


ABC
DEF
DEF
Add
Remove


New
DATE
YEAR:
MONTH:
DAY:
START TIME
Hour
Minute
END TIME
Hour
Minute
TITLE
LOCATION
ABC
ABC
DEF
Appointment Description

Some related Code