Object-Oriented Programming: Revision / Graphics / Subversion

Ewan Klein

Inf1 :: 2008/09

Breaking out of loops, 1

Task: Implement the method `public void contains2(int[] nums)`. Given an array of `int`s and a boolean instance variable `hasATwo`, the method sets variable `hasATwo` to `true` if the array contains a 2.

contains2, Version 1

```
public void contains2(int[] nums) {
    for (int n : nums) {
        if (n == 2) {
            hasATwo = true;
        }
    }
}
```

- Suppose input is `{ 2, 3, 0, 1, 2 }`.
- We don't care about the second 2.

Breaking out of loops, 2

contains2, Version 2

```
public void contains2(int[] nums) {
    for (int n : nums) {
        if (n == 2) {
            hasATwo = true;
            break;
        }
    }
}
```

- As soon as we get to the `break` statement, we exit the loop.
- Avoids needlessly processing the rest of the array.

Breaking out of loops, 3

Task: Implement the method `int[] doubleANumber(int[] nums)`. Given an array of `int`s, this modifies the array so that the first element greater than 2 is doubled. Return the modified array.

`doubleANumber`, Version 1

```
public int[] doubleANumber(int[] nums) {
    for (int i = 0; i < nums.length; i++) {
        if (nums[i] > 2) {
            nums[i] = nums[i] * 2;
        }
    }
    return nums;
}
```

- Suppose input is `{ 1, 3, 3, 3, 3 }`.
- This will return `{ 1, 6, 6, 6, 6 }` but we want `{ 1, 6, 3, 3, 3 }` instead.

Breaking out of loops, 4

Inserting a `break` statement gives the correct behaviour:

`doubleANumber`, Version 2

```
public int[] doubleANumber(int[] nums) {
    for (int i = 0; i < nums.length; i++) {
        if (nums[i] > 2) {
            nums[i] = nums[i] * 2;
            break;
        }
    }
    return nums;
}
```


Real Programmers Use the Command Line

- Or do they?
- In any case, this is less than 1% of population of computer users!
- So building a Graphical User Interface (GUI) is probably on your agenda at some point.
- Java Swing is a good way to get started.

Swing Overview

- Swing is a GUI toolkit, released 1998.
- Developed as successor to Java AWT (Abstract Windowing Toolkit).
- GUI components: buttons, text boxes, menus, panes, etc.
- Pluggable 'look-and-feel'.
- 2D graphics for display and printing.
- Main package: `javax.swing`
- Example classes:
  `JDialog`, `JEditorPane`, `JFileChooser`, `JFormattedTextField`, `JFrame`, `JMenu`, `JPanel`, `JPasswordField`, `JPopupMenu`, `JRadioButton`, `JScrollBar`, `JSlider`

See `http://java.sun.com/javase/6/docs/api/javax/swing/package-summary.html`

Using JFrame, 1

- JFrame is a component for making windows.
- Has two containers: ContentPane and GlassPane
- We will only look at the first of these — using getContentPane() method.

Gui1

```
import java.awt.*;
import javax.swing.*;
public class Gui1 extends JFrame {
    JButton button = new JButton("click me");
    public Gui1() { // Constructor for this class
        super("Inf1 GUI Demo");
        Container cp = getContentPane();
        cp.add(button);
    }
}
```


Using JFrame, 2

- Gui1 is a slight variant on the SimpleGui1 in *HFJ* p. 355.
- We need to import java.awt.* to get access to the Container class.
- super("Inf1 GUI Demo") calls the one-argument constructor of JFrame — the string sets the title for the frame.
- Final line adds a button to the JFrame's ContentPane container.

Running Gui1

The GuiLauncher

```
import javax.swing.*;
public class GuiLauncher {
    public static void main(String[] args) {
        JFrame gui = new Gui1();
        gui.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        gui.setSize(300,300); // Set size of frame, pixels
        gui.setVisible(true);
    }
}
```

- EXIT_ON_CLOSE — program should terminate when frame is closed.
- setVisible(true) — make the frame appear onscreen.

Events

- User clicks a button: a kind of event.
- Event-handling involves noticing and dealing with this.
- GUI components are sources of events:
  - mouse click, keystroke, drag window
  - represented as objects in an event class
- GUI code will typically 'listen' for events from event sources.
- How does an event source know that a listener is able to receive information?
- How does a listener tell an event source that it is interested in the events?

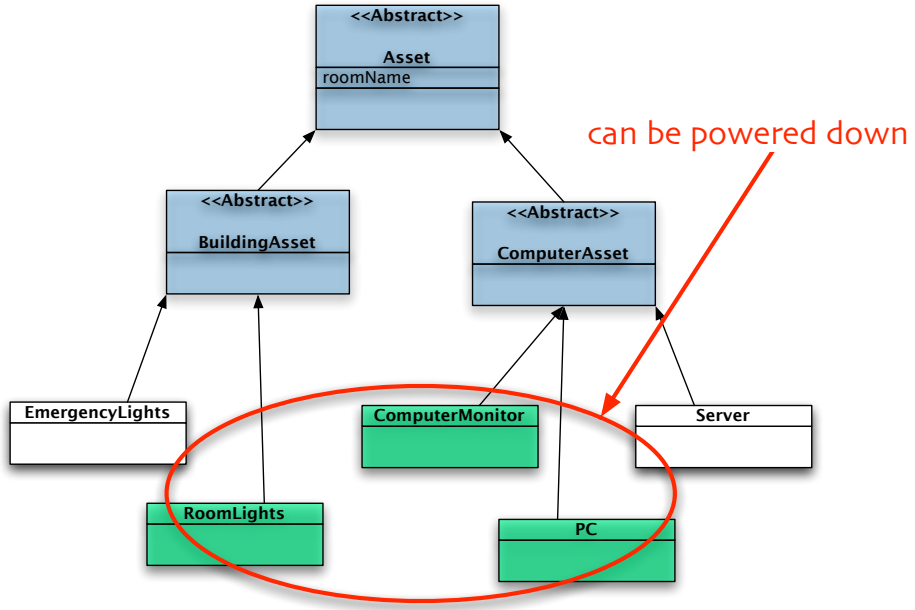
Interfaces Recap

- Interfaces are a special kind of abstract class — also for imposing 'contracts'.
- Can cut across class hierarchies — gives the effect of multiple inheritance.
- All methods in an interface are abstract — so these must be implemented in the subclasses that conform.

The `PowerSwitchable` Interface

```
public interface PowerSwitchable {
    public void powerDown();
    public void powerUp();
}
```


Powersaving for Appleton Tower Devices



Implementing an Interface

Implementing the `PowerSwitchable` Interface

```
public class Monitor extends ComputerAsset implements PowerSwitchable {
    ComputerMonitor(String roomName) {
        super(roomName);
    }
    public void powerDown() {
        System.out.println("Dousing monitor in " + roomName);
    }
    public void powerUp() {
        System.out.println("Warming up monitor in " + roomName);
    }
    public String toString() {
        return "Monitor in " + roomName;
   }
}
```


The `ActionListener` Interface

`ActionListener`

```
public interface ActionListener {
    public void actionPerformed(ActionEvent e)
    // Invoked when an action occurs
}
```

- If a class implements `ActionListener` then it guarantees to implement the `actionPerformed()` method.
- That is, it can respond to an event generated by an event source.

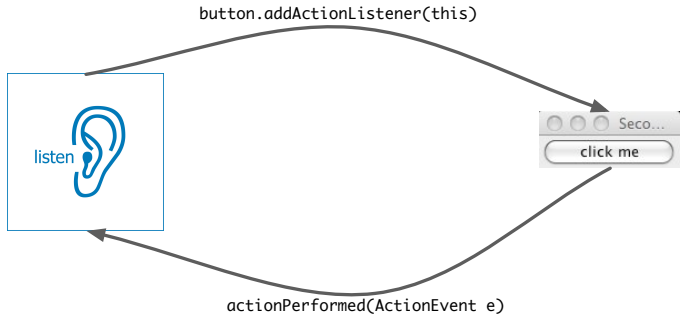
Implementing actionPerformed()

Gui2

```
import java.awt.*;
import java.awt.event.*;  // import ActionListener
import javax.swing.*;
public class Gui2 extends JFrame implements ActionListener {
    JButton button = new JButton("click me");
    public Gui2() {
        super("Second GUI Demo");
        Container cp = getContentPane();
        cp.add(button);
        button.addActionListener(this);
    }
    public void actionPerformed(ActionEvent e) {
        button.setText("I've been clicked");
    }
}
```


Register with a Button, 1

- How does a listener tell an event source that it is interested in the events?
- An instance o of the class that implements `ActionListener` needs to 'register' with event source; e.g. a button.
- That is, call the button's `addActionListener()` method.
- Instance o passes a reference to itself, using `this`.



Register with a Button, 2

Gui2

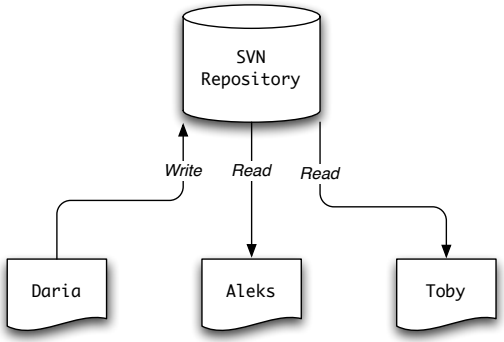
```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
public class Gui2 extends JFrame implements ActionListener {
    JButton button = new JButton("click me");
    public Gui2() {
        super("Second GUI Demo");
        Container cp = getContentPane();
        cp.add(button);
        button.addActionListener(this);
    }
    public void actionPerformed(ActionEvent e) {
        button.setText("I've been clicked");
    }
}
```


Onwards with Swing

- That's all for now on Swing.
- Try to play around with some of the code in Chapters 12 and 13 yourself.

Subversion (SVN)

- Centralized system for sharing information.
- Data is stored in a repository, in the form of files and directories.
- Clients connect to the repository and write or read files.
- The repository remembers every change to every file and directory.

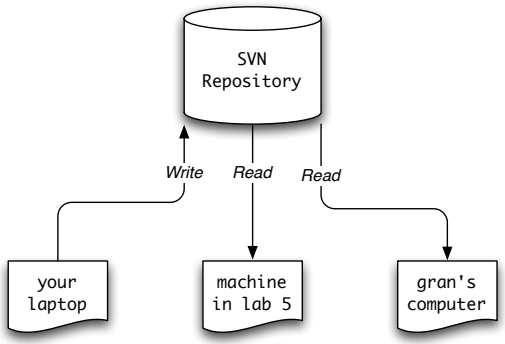


Typical Subversion Workflow

1. `svn co <url>`: Check Out a project (a directory path) from a repository.
2. In that project directory, create or edit files and subdirectories.
3. `svn up(date)`: Update your local copy from the repository, picking up changes your team members may have made since your last update.
4. Go to step 2. If you're ready to commit your changes, go to step 5.
5. `svn ci -m "your message here"`: Check In (commit) your changes to the repository. Go to step 2.

Keeping your computers in sync

- You can have as many checked-out copies of the repository as you like (usually on different machines!)
- Write from one location and read from another.
- Repeat.



Subversion Walkthrough, 1

Checkout a Project

```
 2% svn co svn+ssh://svn.inf.ed.ac.uk/svn/oopsvn
Warning: Permanently added the RSA host key for IP
address '129.215.33.98' to the list of known hosts.
Checked out revision 0.
[svn]
 3% ls
masws           oreilly         rte4            spnlp
nltktrunk       oopsvn          publications    shome
[svn]
 4% cd oopsvn/
[oopsvn]
 5% mkdir test
[oopsvn]
 6% cp ~/svn/shome/.../week10/Square.java test/
```

Subversion Walkthrough, 2

Add some stuff

```
[oopsvn]
 7% svn status
?      test
[oops vn]
 8% svn add test
A         test
A         test/Square.java
[oopsvn]
 9% svn ci -m "just testing the repository" test/
Adding         test
Adding         test/Square.java
Transmitting file data .
Committed revision 1.
```


Subversion Walkthrough, 3

Get info

```
[oopsvn]
 10% svn info test/Square.java
Path: test/Square.java
Name: Square.java
URL: svn+ssh://svn.inf.ed.ac.uk/svn/oopsvn/test/Square.java
Repository Root: svn+ssh://svn.inf.ed.ac.uk/svn/oopsvn
Repository UUID: 9fe6c084-4a2c-489d-8394-c9b9cce425bf
Revision: 1
Node Kind: file
Schedule: normal
Last Changed Author: ewan
Last Changed Rev: 1
Last Changed Date: 2009-03-19 19:38:30 +0000 (Thu, 19 Mar 2009)
Text Last Updated: 2009-03-19 19:38:39 +0000 (Thu, 19 Mar 2009)
Checksum: 316f6cc72adc608a5d5266e5c5a1306a
```


Subversion Walkthrough, 4

Check the log and edit the file

```
[oopsvn]
 11% svn log test/Square.java
------------------------
r1 | ewan | 2009-03-19 19:38:30 +0000 (Thu, 19 Mar 2009) | 1 line

just testing the repository
------------------------
[oopsvn]
 12% aquamacs test/Square.java
```

Do some editing ...

Subversion Walkthrough, 5

Inspect a Revision

```
[oopsvn]
 13% svn diff test/Square.java
Index: test/Square.java
===================================================================
- test/Square.java    (revision 1)
+++ test/Square.java    (working copy)
@@ -1,5 +1,3 @@
-package week10;
-
 public class Square extends Rectangle {

     public Square(int side) {
[oopsvn]
 14% svn status
M      test/Square.java
```

Subversion Walkthrough, 6

Checking in a Revision

```
[oopsvn]
 15% svn ci -m "Deleted package declaration" test/Square.java
Sending        test/Square.java
Transmitting file data .
Committed revision 2.
[oopsvn]
 16% svn up
At revision 2.
```


Experimenting with Subversion, 1

- Everyone registered for Inf1-OOP has read/write access to the oopsvn repository.
- To start off, go to a directory where you want your checked out project to live.
- `svn co svn+ssh://svn.inf.ed.ac.uk/svn/oopsvn`
  - creates the subdirectory oopsvn;
  - checks out a copy of the complete repo in that directory.
- `svn add <file>` — add a file for the first time.
- `svn ci -m "<msg>" <file>` — commit a new file or modification of a file already under SVN control.
- `svn up` — make sure your copy of the repo is up-to-date. You should always do this before committing changes.
- Clashing revisions: SVN will try to merge, or else ask you to resolve conflicts manually.

Experimenting with Subversion, 2

- Try this with a friend: one of you creates and commits a file, the other modifies it; then first person updates their copy from the repo to see the modification.
- Your own space: create (and add and commit) a directory for yourself in oopsvn, using either your name (if unambiguous) or matriculation number.
- Etiquette: don't go uninvited into other people's directories in oopsvn.

Subversion Tools and Resources

- The Subversion Book is available online: `http://svnbook.red-bean.com/`
- Google for 'subversion tutorial'
- Clients:
  - command line; easy to install client on most platforms, download from `http://subversion.tigris.org/`;
  - svn mode in emacs;
  - TortoiseSVN for Windows;
  - Subclipse plugin for Eclipse
  - built into NetBeans.

Reading, etc

- I just skimmed Chapter 12 — try to work through the rest of it yourself.
- Next week: there will be a lab exercise and tutorials.
- Results of mock exam should be available by time of tutorials next week.
- An announcement about extra pre-exam revision classes in April will be made nearer the time.