ECE 462
Object-Oriented Programming using C++ and Java

Static Members and Sharing in Java

Yung-Hsiang Lu
yunglu@purdue.edu
```java
class User {
    public String name;
    public int age;
    public User(String s, int a) { name = s; age = a; }
    public User(final User orig) {
        // required to create u3
        name = orig.name; age = orig.age;
    }
    public String toString() {
        return name + " " + age;
    }
}

public class CreateObject {
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        User u1 = new User("Robert", 21);
        System.out.println(u1); // Robert 21
        User u2 = u1; // does not actually create an object
        u2.name = "Tom";
        u2.age = 33;
        System.out.println(u1); // Tom 33
        User u3 = new User(u1);
        System.out.println(u3); // Tom 33
        u1.name = "John";
        u1.age = 19;
        System.out.println(u1); // John 19
        System.out.println(u3); // Tom 33
    }
}
```
Object Creation in Java

• Java does not automatically create copy constructors.
• Objects must be created using \texttt{new}; therefore, \texttt{u2} is not a separate object.
• Objects, once created, do not share attributes. Changing one does not affect another.
• Java does not allow operator overloading by programmers. Hence, it is not possible to redefine operator \texttt{=}.
• In Java, operator \texttt{=} does not perform copy. It creates "alias". The previous object becomes garbage.
Static Member and Memory Sharing
```java
class ShareMemory {

    private static int[] u_data;

    // no need to keep track of object count
    // no need to keep a separate size for the data array

    public ShareMemory(int sz) {
        if (u_data == null) {
            u_data = new int[sz];
        }

        if (sz > u_data.length) {
            int[] newData = new int[sz];
            for (int index = 0; index < u_data.length; index++) {
                newData[index] = u_data[index];
            }

            u_data = newData; // old u_data becomes garbage
        }
    }

    private boolean checkIndex(int index) {
        if ((index < 0) || (index >= u_data.length)) {
            System.out.println("index out of range");
            return false;
        }

        return true;
    }

    public int read(int index) {
        if (checkIndex(index)) {
            return u_data[index];
        }
    }
}
```
public int read(int index) {
    if (checkIndex(index)) {
        return u_data[index];
    }
    return -1;
}

public void write(int index, int value) {
    if (checkIndex(index)) {
        u_data[index] = value;
    }
}

public int getSize() {
    if (u_data == null) {
        return 0;
    }
    return u_data.length;
}

public class ShareMemoryMain {
    public static void main(String[] args) {
        ShareMemory sm1 = new ShareMemory(10);
        for (int index = 0; index < 10; index++) {
            sm1.write(index, index);
        }
    }
}
```java
public int getSize() {
    if (u_data == null) {
        return 0;
    }
    return u_data.length;
}

public class ShareMemoryMain {
    public static void main(String[] args) {
        ShareMemory sm1 = new ShareMemory(10);
        for (int index = 0; index < 10; index++) {
            sm1.write(index, index);
        }
        ShareMemory sm2 = new ShareMemory(6);
        sm2.write(4, -9);
        System.out.println(sm1.read(4));
        System.out.println(sm2.getSize());
        ShareMemory sm3 = new ShareMemory(20);
        System.out.println(sm1.getSize());
        sm1.write(16, 77);
        System.out.println(sm3.read(16));
    }
}
```
Self Test
ECE 462
Object-Oriented Programming using C++ and Java

Game Programs

Yung-Hsiang Lu
yunglu@purdue.edu
Developing Complex Programs
(Using Games as Examples)
Java Games

• Applet games: running through web browser
  + no installation needed, easy upgrade (the web version is always the latest)
  – security restrictions, cannot save game status

• Window games:
  + no restrictions like applets
  – players may be distracted by other windows

• Full-screen games:
  + no other program can appear to distract players
  – do not allow players to engage in other activities (such as instant messaging)
Applet Game
Window Game
Full Screen Game
Developing Games in Java

- Get the book
- Play the example game
- Download the source
- View the notes & errata
- What's happened since the book was released?
- View source code license

Get the book
Download source code

These packages include source code and any resources (graphics, sounds, scripts, etc) needed to run the examples.

Requires Apache Ant 1.5 to compile. Ant is either directly integrated or available as a plugin for several free/open source editors and IDEs, including JEdit, NetBeans, and Eclipse.

If you use Apache Ant, everything compiles error-free! See the errata below for any issues.

<table>
<thead>
<tr>
<th>Description</th>
<th>Download</th>
</tr>
</thead>
<tbody>
<tr>
<td>All source code</td>
<td>allsrc.zip (4.3MB)</td>
</tr>
<tr>
<td></td>
<td>or allsrc.tar.bz2 (3.2MB)</td>
</tr>
<tr>
<td>Chapter 1, &quot;Java Threads&quot;</td>
<td>ch01src.zip</td>
</tr>
<tr>
<td>Chapter 2, &quot;2D Graphics and Animation&quot;</td>
<td>ch02src.zip</td>
</tr>
<tr>
<td>Chapter 3, &quot;Interactivity and User Interfaces&quot;</td>
<td>ch03src.zip</td>
</tr>
</tbody>
</table>
Create a Java Project from an Ant Buildfile

Create a new Java project based on the specification of a javac task in the Ant buildfile. This does not copy the source contents to the workspace.

**Project Name:** chap02 - 2D Graphics and Animation

**Ant Buildfile:** C:\yungle\ece462\JavaGame\chap02\src\build.xml

Select javac declaration to use to define project:

```
<javac taskname="compile" srcdir="chap02\src" destdir="chap02\classes">
```

- Link to the buildfile in the file system

[Next] Finish Cancel
C:\yunglu\ece462\JavaGame\ch02src>java AnimationTest1
ECE 462
Object-Oriented Programming using C++ and Java

Full-Screen Games

Yung-Hsiang Lu
yunglu@purdue.edu
Create a Java Project in Netbeans using Existing Code
Imports an existing Java application into a free-form IDE project. A free-form project uses your existing Ant build script to run, compile, and debug your project.
New Java Project with Existing Ant Script

Steps
1. Choose Project
2. **Name and Location**
3. Build and Run Actions
4. Source Package Folders
5. Java Sources Classpath

Name and Location

Select the folder that contains the project's files and specify the location of the build script.

**Location:**

```asf
C:\lyunglu\ece462\JavaGame\ch02src
```

**Build Script:**

```asf
C:\lyunglu\ece462\JavaGame\ch02src\build.xml
```

Specify a name and location for the new project.

**Project Name:**

```asf
chap02
```

**Project Folder:**

```asf
C:\lyunglu\ece462\JavaGame\ch02src
```

- [ ] Set as Main Project

< Back   Next >   Finish   Cancel   Help
### New Java Project with Existing Ant Script

#### Steps

1. Choose Project
2. Name and Location
3. **Build and Run Actions**
4. Source Package Folders
5. Java Sources Classpath

#### Build and Run Actions

<table>
<thead>
<tr>
<th>Action</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Build Project:</td>
<td>compile</td>
</tr>
<tr>
<td>Clean Project:</td>
<td>clean</td>
</tr>
<tr>
<td>Generate Javadoc:</td>
<td>javadoc</td>
</tr>
<tr>
<td>Run Project:</td>
<td></td>
</tr>
<tr>
<td>Test Project:</td>
<td></td>
</tr>
</tbody>
</table>

### YHL Full-Screen Games

6
Steps

1. Choose Project
2. Name and Location
3. Build and Run Actions
4. Source Package Folders
5. Java Sources Classpath

Source Package Folders

Specify the folders containing the Java source packages and JUnit test packages.

Source Package Folders:

Package Folder

Test Package Folders:

Package Folder
Full Screen Test
Hello World!
import java.awt.*;

public class FullScreenTest extends JFrame {

    public static void main(String[] args) {

        DisplayMode displayMode;

        if (args.length == 3) {
            displayMode = new DisplayMode(
                Integer.parseInt(args[0]),
                Integer.parseInt(args[1]),
                Integer.parseInt(args[2]),
                DisplayMode.REFRESH_RATE_UNKNOWN);
        } else {
            displayMode = new DisplayMode(800, 600, 16,
                DisplayMode.REFRESH_RATE_UNKNOWN);
        }

        FullScreenTest test = new FullScreenTest();
        test.runTest(displayMode);
    }

    private static final long DEMO_TIME = 5000;

    public void runTest(DisplayMode displayMode) {
    }
on some machines, the background appears black

```java
private static final long DEMO_TIME = 5000;

public void runTest(DisplayMode displayMode) {
    setBackground(Color.blue);
    setForeground(Color.white);
    setFont(new Font("Dialog", 0, 24));

    SimpleScreenManager screen = new SimpleScreenManager();
    try {
        screen.setFullScreen(displayMode, this);
        try {
            Thread.sleep(DEMO_TIME);
        }
        catch (InterruptedException ex) { }
    }
    finally {
        screen.restoreScreen();
    }
}

public void paint(Graphics g) {
    g.drawString("Hello World!", 20, 50);
}
```
public final class DisplayMode
extends Object

The DisplayMode class encapsulates the bit depth, height, width, and refresh rate of a GraphicsDevice. The ability to change graphics device's display mode is platform- and configuration-dependent and may not always be available (see GraphicsDevice.isDisplayChangeSupported()).

For more information on full-screen exclusive mode API, see the Full-Screen Exclusive Mode API Tutorial.

Since:
1.4

See Also:
public abstract class GraphicsDevice
extends Object

The GraphicsDevice class describes the graphics devices that might be available in a particular graphics environment. These include screen and printer devices. Note that there can be many screens and many printers in an instance of GraphicsEnvironment. Each graphics device has one or more GraphicsConfiguration objects associated with it. These objects specify the different configurations in which the GraphicsDevice can be used.

In a multi-screen environment, the GraphicsConfiguration objects can be used to render components on multiple screens. The following code sample demonstrates how to create a JFrame object for each GraphicsConfiguration on each screen device in the GraphicsEnvironment.
java.awt

Class GraphicsEnvironment

```
java.lang.Object
    |-- java.awt.GraphicsEnvironment
```

```java
public abstract class GraphicsEnvironment
    extends Object

The GraphicsEnvironment class describes the collection of GraphicsDevice objects and Font objects available to a Java(tm) application on a particular platform. The resources in this GraphicsEnvironment might be local or on a remote machine. GraphicsDevice objects can be screens, printers or image buffers and are the destination of graphicsID drawing methods. Each GraphicsDevice has a number of GraphicsConfiguration objects associated with it. These objects specify the different configurations in which the GraphicsDevice can be used.

See Also:

GraphicsDevice, GraphicsConfiguration
```
import java.awt.*;

/**
 * The SimpleScreenManager class manages initializing and displaying full screen graphics modes.
 */
public class SimpleScreenManager {

    private GraphicsDevice device;

    /**
     * Creates a new SimpleScreenManager object.
     */
    public SimpleScreenManager() {
        GraphicsEnvironment environment =
            GraphicsEnvironment.getLocalGraphicsEnvironment();
        device = environment.getDefaultScreenDevice();
    }

    /**
     * Enters full screen mode and changes the display mode.
     */
    public void setFullScreen(DisplayMode displayMode,
                                JFrame window)
    {
        window.setUndecorated(true);
        window.setResizable(false);
        device.setFullScreenWindow(window);
        if (displayMode != null &&
            device.isDisplayChangeSupported())
        {
            try {
            } catch (Exception e) {
            }
        }
    }
}
device.isDisplayChangeSupported()
{
	try {
	
device.setDisplayMode(displayMode);
	}
	catch (IllegalArgumentException ex) {
		// ignore - illegal mode for this device
	}

/**
 * Returns the window currently used in full screen mode.
 */
public Window getFullScreenWindow()
{
	return device.getFullScreenWindow();
}

/**
 * Restores the screen's display mode.
 */
public void restoreScreen()
{
	Window window = device.getFullScreenWindow();
	if (window != null) {
		window.dispose();
	}
	device.setFullScreenWindow(null);
}
Anti Alias
Hello World!

↑ notice the stairs in W

↓

WA

Hello World! (anti-alias)
ignore the errors reported by Netbeans. This program is built using ant.
java.awt

Class RenderingHints

java.lang.Object
   - java.awt.RenderingHints

All Implemented Interfaces:
       Cloneable, Map<Object, Object>

public class RenderingHints
extends Object
implements Map<Object, Object>, Cloneable

The RenderingHints class defines and manages collections of keys and associated values which allow an application to provide input into the choice of algorithms used by other classes which perform rendering and image manipulation services. The Graphics2D class, and classes that implement BufferedImageOp and RasterOp all provide methods to get and possibly to set individual or groups of RenderingHints keys and their associated values. When those implementations perform any rendering or image manipulation operations they should examine the values of any RenderingHints that were requested by the
Display Images
Image Types

- **Opaque**
- **Transparent**
- **Translucent**
Load and Display Images
import java.awt.*;
import javax.swing.ImageIcon;
import javax.swing.JFrame;

public class ImageTest extends JFrame {
    public static void main(String[] args) {
        DisplayMode displayMode;
        if (args.length == 3) {
            displayMode = new DisplayMode(
                Integer.parseInt(args[0]),
                Integer.parseInt(args[1]),
                Integer.parseInt(args[2]),
                DisplayMode.REFRESH_RATE_UNKNOWN);
        } else {
            displayMode = new DisplayMode(800, 600, 16,
                DisplayMode.REFRESH_RATE_UNKNOWN);
        }

        ImageTest test = new ImageTest();
        test.runTest(displayMode);
    }

    private static final int FONT_SIZE = 24;
    private static final long DEMO_TIME = 10000;
    private SimpleScreenManager screen;
    private Image bgImage;
    private Image opaqueImage;
    private Image transparentImage;
private SimpleScreenManager screen;
private Image bgImage;
private Image opaqueImage;
private Image transparentImage;
private Image translucentImage;
private Image antiAliasedImage;
private boolean imagesLoaded;

public void runTest(DisplayMode displayMode) {
    setBackground(Color.blue);
    setForeground(Color.white);
   setFont(new Font("Dialog", Font.PLAIN, FONT_SIZE));
    imagesLoaded = false;

    screen = new SimpleScreenManager();
    try {
        screen.setFullScreen(displayMode, this);
        loadImages();
        try {
            Thread.sleep(DEMO_TIME);
        }
        catch (InterruptedException ex) {
        }
    }
    finally {
        screen.restoreScreen();
    }
}
```java
    screen.restoreScreen();
}

public void loadImages()
    bgImage = loadImage("images/background.jpg");
    opaqueImage = loadImage("images/opaque.png");
    transparentImage = loadImage("images/transparent.png");
    translucentImage = loadImage("images/translucent.png");
    antiAliasedImage = loadImage("images/antialiased.png");
    imagesLoaded = true;
    // signal to AWT to repaint this window
    repaint();
}

private Image loadImage(String fileName)
    return new ImageIcon(fileName).getImage();
}

public void paint(Graphics g) {
    // set text anti-aliasing
    if (g instanceof Graphics2D) {
        Graphics2D g2 = (Graphics2D)g;
        g2.setRenderingHint(
            RenderingHints.KEY_TEXT_ANTIALIASING,
            RenderingHints.VALUE_TEXT_ANTIALIAS_ON);
    }

    // draw images
    if (imagesLoaded) {
```
g2.setRenderingHint(
    RenderingHints.KEY_TEXT_ANTIALIASING,
    RenderingHints.VALUE_TEXT_ANTIALIAS_ON);

// draw images
if (imagesLoaded) {
    g.drawImage(bgImage, 0, 0, null);
    drawImage(g, opaqueImage, 0, 0, "Opaque");
    drawImage(g, transparentImage, 320, 0, "Transparent");
    drawImage(g, translucentImage, 0, 300, "Translucent");
    drawImage(g, antiAliasedImage, 320, 300,
        "Translucent (Anti-Aliased)";
}
else {
    g.drawString("Loading Images...", x, FONT_SIZE);
}

public void drawImage(Graphics g, Image image, int x, int y, String caption) {
    g.drawImage(image, x, y, null);
    g.drawString(caption, x + 5, y + FONT_SIZE + image.getHeight(null));
}
java.awt

Class MediaTracker

java.lang.Object
   ^-- java.awt.MediaTracker

All Implemented Interfaces:
   Serializable

public class MediaTracker
   extends Object
   implements Serializable

The MediaTracker class is a utility class to track the status of a number of media objects. Media objects could include audio clips as well as images, though currently only images are supported.

To use a media tracker, create an instance of MediaTracker and call its addImage method for each image to be tracked. In addition, each image can be assigned a unique identifier. This identifier controls the priority order in which the images are
Animation
Animation

- Change frames with small differences quickly.
- The time in each frame may be different.
import java.awt.Image;
import java.util.ArrayList;

/**
 * The Animation class manages a series of images (frames) and
 * the amount of time to display each frame.
 */
public class Animation {
    private ArrayList<AnimFrame> frames;
    private int currFrameIndex;
    private long animTime;
    private long totalDuration;
    /**
     * Creates a new, empty Animation.
     */
    public Animation() {
        frames = new ArrayList<AnimFrame>();
        totalDuration = 0;
        start();
    }

    /**
     * Adds an image to the animation with the specified
     * duration (time to display the image).
     */
    public synchronized void addFrame(Image image, int duration) {
        frames.add(new AnimFrame(image, duration, 0));
    }
}
> javac -Xlint:unchecked Animation.java
Animation.java:34: warning: [unchecked] unchecked call to add(E) as a member of the raw type
java.util.ArrayList
frames.add(new AnimFrame(image, totalDuration));
duration (time to display the image).
* /
  public synchronized void addFrame(Image image,
      long duration)
  {
      totalDuration += duration;
      frames.add(new AnimFrame(image, totalDuration));
  }

  /**
   * Starts this animation over from the beginning.
   * /
  public synchronized void start() {
      animTime = 0;
      currFrameIndex = 0;
  }

  /**
   * Updates this animation's current image (frame), if
   * neccessary.
   * /
  public synchronized void update(long elapsedTime) {
      if (frames.size() > 1) {
          animTime += elapsedTime;
          if (animTime >= totalDuration) {
              animTime = animTime & totalDuration;
          }
      }
  }
```java
if (animTime >= totalDuration) {
    animTime = animTime % totalDuration;
    currFrameIndex = 0;
}

while (animTime > getFrame(currFrameIndex).endTime) {
    currFrameIndex++;
}

/**
 * Gets this Animation's current image. Returns null if this animation has no images.
 */
public synchronized Image getImage() {
    if (frames.size() == 0) {
        return null;
    } else {
        return getFrame(currFrameIndex).image;
    }
}

private AnimFrame getFrame(int i) {
    return (AnimFrame)frames.get(i);
}
```
```java
public synchronized Image getImage() {
    if (frames.size() == 0) {
        return null;
    } else {
        return getFrame(currFrameIndex).image;
    }
}
```

```java
private AnimFrame getFrame(int i) {
    return (AnimFrame)frames.get(i);
}
```

```java
private class AnimFrame {
    Image image;
    long endTime;

    public AnimFrame(Image image, long endTime) {
        this.image = image;
        this.endTime = endTime;
    }
}
```
<table>
<thead>
<tr>
<th>frame</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>duration (ms)</td>
<td>150</td>
<td>250</td>
<td>300</td>
<td>150</td>
<td>200</td>
</tr>
<tr>
<td>endTime</td>
<td>150</td>
<td>400</td>
<td>700</td>
<td>850</td>
<td>1050</td>
</tr>
</tbody>
</table>

- **animTime = 600 ⇒**
  - animTime > 150? Yes, index ++ ⇒ index = 1
  - animTime > 400? Yes, index ++ ⇒ index = 2
  - animTime > 700? No, index = 2

- **animTime = 2900 ⇒ 2 iterations = 2100**
  - animTime = 2900 % 1050 = 800, index = 0
  - animTime > 150? Yes, index ++ ⇒ index = 1
  - animTime > 400? Yes, index ++ ⇒ index = 2
  - animTime > 700? Yes, index ++ ⇒ index = 3
  - animTime > 850? No, index = 3
Synchronized Methods

• At any moment, only one of each object's synchronized methods can execute.

   Animation anim = new Amination();
   anim.addFrame(...);
   anim.addFrame(...);
   ...
   anim.update(...);

   the execution time of
   synchronized methods
cannot overlap

• Since the program is not multi-thread, it is unnecessary to make methods synchronized.
```java
import java.awt.*;
import javax.swing.ImageIcon;
import javax.swing.JFrame;

public class AnimationTest1 {
    public static void main(String args[]) {
        DisplayMode displayMode;
        if (args.length == 3) {
            displayMode = new DisplayMode(
                Integer.parseInt(args[0]),
                Integer.parseInt(args[1]),
                Integer.parseInt(args[2]),
                DisplayMode.REFRESH_RATE_UNKNOWN);
        } else {
            displayMode = new DisplayMode(800, 600, 16,
                DisplayMode.REFRESH_RATE_UNKNOWN);
        }

        AnimationTest1 test = new AnimationTest1();
        test.run(displayMode);
    }

    private static final long DEMO_TIME = 5000;
    private SimpleScreenManager screen;
    private Image bgImage;
    private Animation anim;
}
```
```java
public void loadImages() {
    // load images
    bgImage = loadImage("images/background.jpg");
    Image player1 = loadImage("images/player1.png");
    Image player2 = loadImage("images/player2.png");
    Image player3 = loadImage("images/player3.png");

    // create animation
    anim = new Animation();
    anim.addFrame(player1, 250);
    anim.addFrame(player2, 150);
    anim.addFrame(player1, 150);
    anim.addFrame(player2, 150);
    anim.addFrame(player3, 200);
    anim.addFrame(player2, 150);
}

private Image loadImage(String fileName) {
    return new ImageIcon(fileName).getImage();
}

public void run(DisplayMode displayMode) {
    screen = new SimpleScreenManager();
    try {
        screen.setFullScreen(displayMode, new JFrame());
        loadImages();
        animationLoop();
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```
public void run(DisplayMode displayMode) {
    screen = new SimpleScreenManager();
    try {
        screen.setFullScreen(displayMode, new JFrame());
        loadImages();
        animationLoop();
    }
    finally {
        screen.restoreScreen();
    }
}

public void animationLoop() {
    long startTime = System.currentTimeMillis();
    long currTime = startTime;
    
    while (currTime - startTime < DEMO_TIME) {
        long elapsedTime =
            System.currentTimeMillis() - currTime;
        currTime += elapsedTime;
        
        // update animation
        anim.update(elapsedTime);
        
        // draw to screen
        Graphics g =
            screen.getFullScreenWindow().getGraphics();
    }
// update animation
anim.update(elapsedTime);

// draw to screen
Graphics g =
        screen.getFullScreenWindow().getGraphics();
draw(g);
g.dispose();

// take a nap
try {
    Thread.sleep(20);
}
catch (InterruptedException ex) {
}

public void draw(Graphics g) {
    // draw background
    g.drawImage(bgImage, 0, 0, null);

    // draw image
    g.drawImage(anim.getImage(), 0, 0, null);
}