Lecture 9: Class Libraries & Packages, GUI Applications

CSC 1214: Object-Oriented Programming

Outline

- Class Libraries & Packages
- GUI Applications

Outline

- Class Libraries & Packages
- GUI Applications

Class Libraries

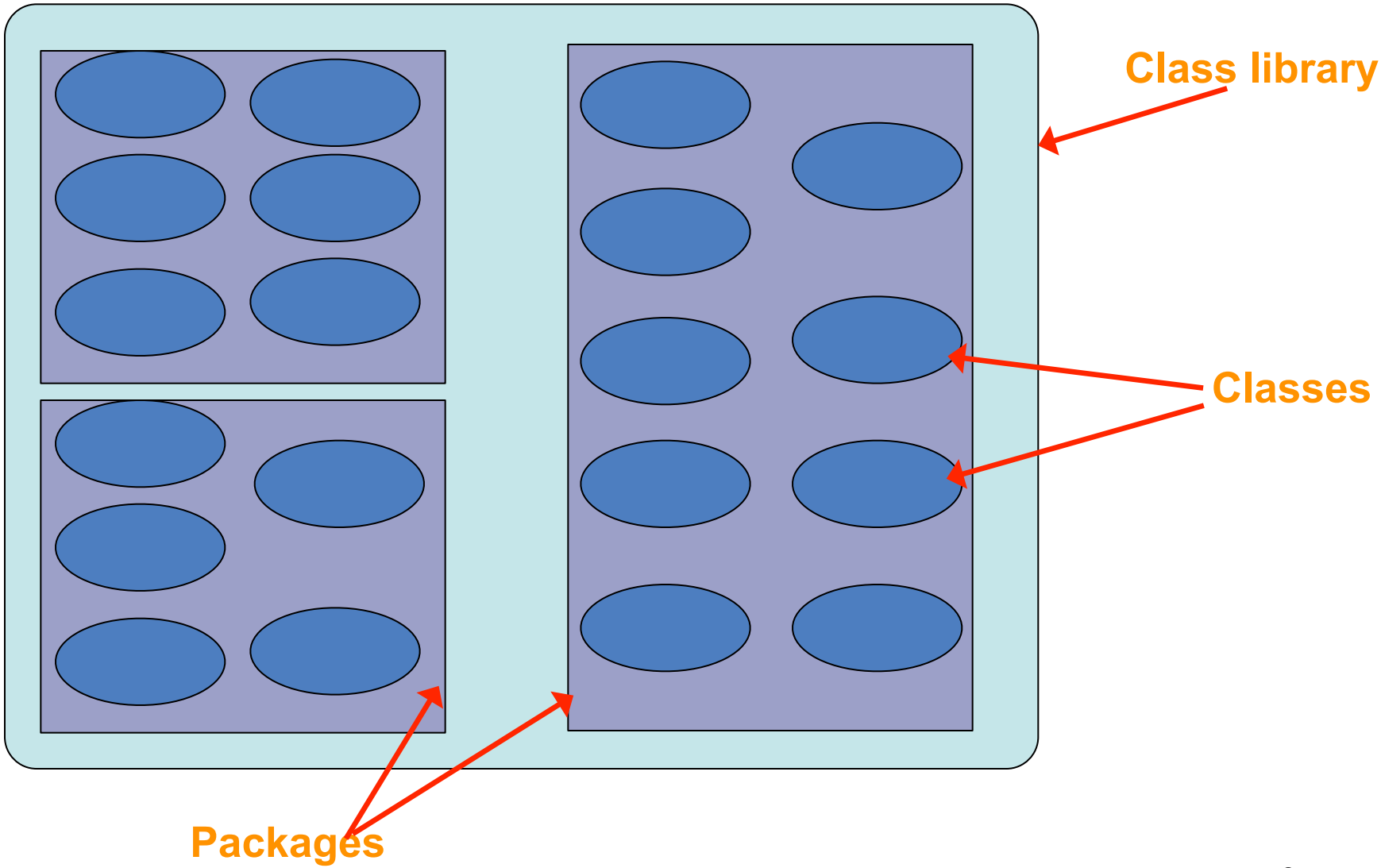
- A class library is a collection of classes that we can use when developing programs
- The Java standard class library is part of any Java development environment
- Its classes are not part of the Java language per se, but we rely on them heavily
- The **`System`** class and the **`String`** class are part of the Java standard class library
- Other class libraries can be obtained through third party vendors, or you can create them yourself

Packages

- The classes of the Java standard class library are organized into packages
- Some of the packages in the standard class library are:

Package	Purpose
java.lang	General support
java.applet	Creating applets for the web
java.awt	Graphics and graphical user interfaces
javax.swing	Additional graphics capabilities and components
java.net	Network communication
java.util	Utilities
javax.xml.parsers	XML document processing

Class Libraries & Packages



Working With Packages

- When you want to use a class from a package, you could use its fully qualified name

```
java.util.Random
```

- Or you can import the class, and then use just the class name

```
import java.util.Random;
```

- To import all classes in a particular package, you can use the * wildcard character

```
import java.util.*;
```

Working With Packages

- All classes of the **`java.lang`** package are imported automatically into all programs
- That's why we didn't have to import the **`System`** or **`String`** classes explicitly in earlier programs
- The **`Random`** class is part of the **`java.util`** package
- It provides methods that generate pseudorandom numbers

Creating Packages

- To create a package you use the `package` Java reserved keyword

- Put a `package` statement with the package name at the top of every source file that contains the types (classes, interfaces, enumerations, and annotation types) that you want to include in the package.

```
package pets;
public class Cat implements Animal
{
    public void makeSound ()
    {
        System.out.println("Meow");
    }
}
```

Naming Conventions

- Package names are usually written in all lower case to avoid conflict with the names of classes or interfaces.
- Companies use their reversed Internet domain name to begin their package names e.g., ug.ac.mak.cit.package_name for a programmer at CIT

```
package ug.ac.mak.cit.pets;
public class Cat implements Animal
{
    public void makeSound ()
    {
        System.out.println("Meow");
    }
}
```

Outline

- Class Libraries & Packages
- GUI Applications

GUI Applications

- Until now, the example programs we have explored have been text-based
- They are called command-line applications, which interact with the user using simple text prompts
- Let's examine some Java applications that have graphical components
- These components will serve as a foundation to programs that have true graphical user interfaces (GUIs)

GUI Components

- A GUI component is an object that represents a screen element such as a button or a text field
- GUI-related classes are defined primarily in the java.awt and the javax.swing packages
- The Abstract Windowing Toolkit (AWT) was the original Java GUI package
- The Swing package provides additional and more versatile components
- Both packages are needed to create a Java GUI-based program

GUI Containers

- A GUI container is a component that is used to hold and organize other components
- A frame is a container that is used to display a GUI-based Java application
- A frame is displayed as a separate window with a title bar – it can be repositioned and resized on the screen as needed
- A panel is a container that cannot be displayed on its own but is used to organize other components
- A panel must be added to another container to be displayed

Labels

- A label is a GUI component that displays a line of text
- Labels are usually used to display information or identify other components in the interface

Example

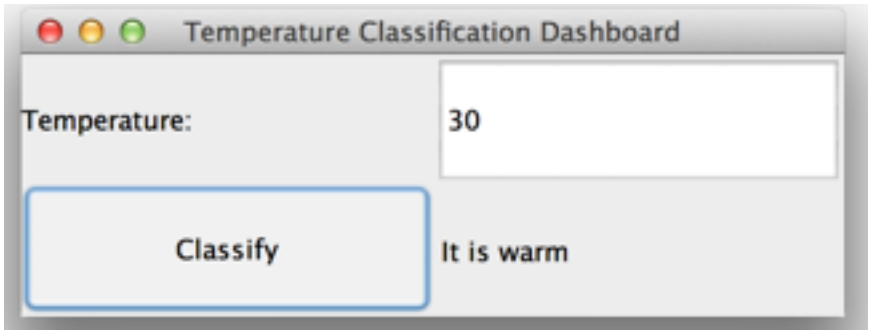
Remember our text-based thermometer example in Lecture 5

```
class Thermometer {
   public static void main(String args[]) {
      double currentTemp = 20.0;

      System.out.println("Current temperature is "+ currentTemp);
      if (currentTemp > 30.0)
         System.out.println(" It is too hot");
         else
           System.out.println(" It is warm or cold");

      }
}
```

Example

Let's create a GUI application for the same program

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
public class TemperatureDashboard implements ActionListener
{
    private Container pane;
    private JTextField textfield;
    private JButton classifyButton;
    private JLabel tempLabel, classificationLabel;
    private JFrame frame;

    public TemperatureDashboard()
    {
        frame = new JFrame("Temperature Classification Dashboard");
        frame.setSize(800,300);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        pane = frame.getContentPane();
        pane.setLayout(new GridLayout(2,2));

        tempLabel = new JLabel("Temperature: ");
        classifyButton = new JButton("Classify");
        textfield = new JTextField(5);
        classificationLabel = new JLabel("");

        pane.add(tempLabel);
        pane.add(textfield);
        pane.add(classifyButton);
        pane.add(classificationLabel);

        classifyButton.addActionListener(this);
        frame.setVisible(true);
    }

    public String classifyTemp(int currentTemp){
        if (currentTemp > 30)
            return " It is too hot";
        else if (currentTemp > 18)
            return " It is warm";
        else
            return " It is too cold";
    }
    public void actionPerformed (ActionEvent event)
    {
        int temp;
        String text = textfield.getText();
        temp = Integer.parseInt (text);
        String classification = classifyTemp(temp);
        classificationLabel.setText (classification);
    }

    public static void main(String args[]){
        TemperatureDashboard tempDashboard = new TemperatureDashboard();

       }

}
```



Example

Let's create a GUI application for the same program

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
public class TemperatureDashboard implements ActionListener
{
    private Container pane;
    private JTextField textfield;
    private JButton classifyButton;
    private JLabel tempLabel, classificationLabel;
    private JFrame frame;

    public TemperatureDashboard()
    {
        frame = new JFrame("Temperature Classification Dashboard");
        frame.setSize(800,300);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        pane = frame.getContentPane();
        pane.setLayout(new GridLayout(2,2));

        tempLabel = new JLabel("Temperature: ");
        classifyButton = new JButton("Classify");
        textfield = new JTextField(5);
        classificationLabel = new JLabel("");

        pane.add(tempLabel);
        pane.add(textfield);
        pane.add(classifyButton);
        pane.add(classificationLabel);

        classifyButton.addActionListener(this);

        frame.setVisible(true);
    }

    public String classifyTemp(int currentTemp){
        if (currentTemp > 30)
            return " It is too hot";
        else if (currentTemp > 18)
            return " It is warm";
        else
            return " It is too cold";
    }
    public void actionPerformed (ActionEvent event)
    {
        int temp;
        String text = textfield.getText();
        temp = Integer.parseInt (text);
        String classification = classifyTemp(temp);
        classificationLabel.setText (classification);
    }

    public static void main(String args[]){
        TemperatureDashboard tempDashboard = new TemperatureDashboard();

       }

}
```

Importing GUI-related classes

Defining GUI components

Initializing the GUI components, and adding them to the container

Specifying this object as a listener to the mouse click events of the button

Making the frame visible

The classifyTemp method implements the logic of the application

This method is invoked whenever the "Classify" button is clicked

Example

Let's create a GUI application for the same program

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
public class TemperatureDashboard implements ActionListener
{
    private Container pane;
    private JTextField textfield;
    private JButton classifyButton;
    private JLabel tempLabel, classificationLabel;
    private JFrame frame;

    public TemperatureDashboard()
    {
        frame = new JFrame("Temperature Classification Dashboard");
        frame.setSize(800,300);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        pane = frame.getContentPane();
        pane.setLayout(new GridLayout(2,2));

        tempLabel = new JLabel("Temperature: ");
        classifyButton = new JButton("Classify");
        textfield = new JTextField(5);
        classificationLabel = new JLabel("");

        pane.add(tempLabel);
        pane.add(textfield);
        pane.add(classifyButton);
        pane.add(classificationLabel);

        classifyButton.addActionListener(this);
        frame.setVisible(true);
    }

    public String classifyTemp(int currentTemp){
        if (currentTemp > 30)
            return " It is too hot";
        else if (currentTemp > 18)
            return " It is warm";
        else
            return " It is too cold";
    }
    public void actionPerformed (ActionEvent event)
    {
        int temp;
        String text = textfield.getText();
        temp = Integer.parseInt (text);
        String classification = classifyTemp(temp);
        classificationLabel.setText (classification);
    }

    public static void main(String args[]){
        TemperatureDashboard tempDashboard = new TemperatureDashboard();

       }

}
```