CSE2016 Programming Methodology

Week 4: Input, Output, and State

Instructor: Jinyoung Han (jinyounghan@hanyang.ac.kr)

HANYANG UNIVERSITY

Contents

HANYANG UNIVERSITY

Today's Schedule

01. Interactive Input

Celsius To Fahrenheit

```
import java.text.*;

public class CelsiusToFahrenheit
{
    public static void main(String[] args)
    {
        int c = new Integer(args[0]).intValue();
        double f = ((9.0/5.0)*c) + 32;
        System.out.println("For Celsius degrees " + c + ",");
        DecimalFormat formatter = new DecimalFormat("0.0");
        System.out.println("Degrees Fahrenheit = " +
        formatter.format(f));
    }
}
```

01. Interactive Input

Celsius To Fahrenheit

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

public class CelsiusToFahrenheit
{
    public static void main(String[] args)
    {
        Scanner in = new Scanner(System.in);
        System.out.print("Input: ");
        int c = in.nextInt();
        double f = ((9.0/5.0)*c) + 32;
        DecimalFormat formatter = new DecimalFormat("0.0");
        System.out.println("For Celsius degrees " + c + ",");
        System.out.println("Degrees Fahrenheit = " +
        formatter.format(f));
    }
}
```

01. Interactive Input

Input Window

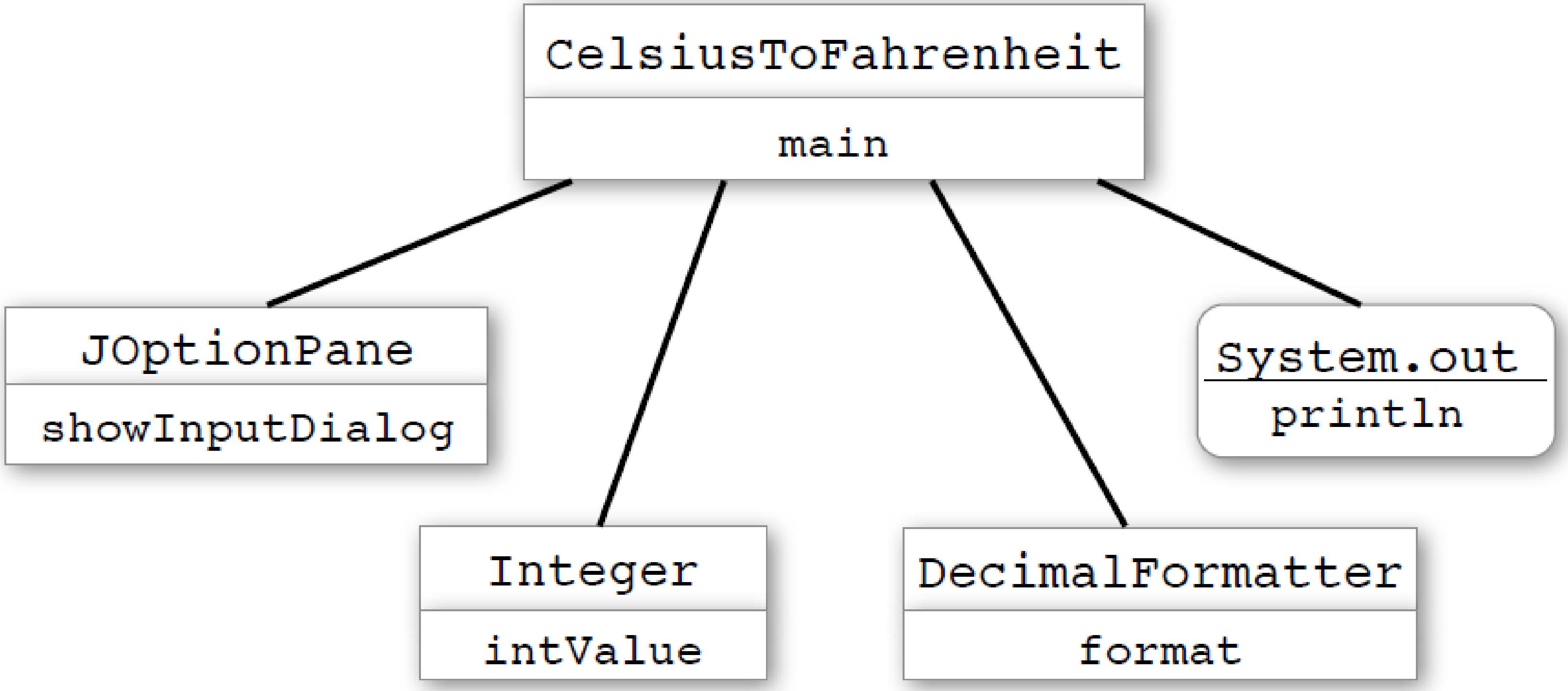
- Input Window Class
  - javax.swing.JOptionPane

```
import javax.swing.*;

String s = JOptionPane.showInputDialog("message");
```

  - We don’t need to use “new” for object creation
  - JOptionPane object can do:
    - Creating a message window
    - Getting input
    - Passing the input

01. Interactive Input

Class Diagram



01. Interactive Input

CelsiusToFahrenheit.java

```
import java.text.*;
import javax.swing.*;

public class CelsiusToFahrenheit
{
    public static void main(String[] args)
    {
        String input = JOptionPane.showInputDialog("Type an integer
        Celsius temperature");
        int c = new Integer(input).intValue();
        double f = ((9.0/5.0)*c) + 32;
        System.out.println("For Celsius degrees " + c + ",");
        DecimalFormat formatter = new DecimalFormat("0.0");
        System.out.println("Degrees Fahrenheit = " +
        formatter.format(f));
    }
}
```

01. Interactive Input

Output Window

- Output Window Class
  - javax.swing.JOptionPane

```
import javax.swing.*;

JOptionPane.showMessageDialog(null, "message");
```

  - We don’t need to use “new” for object creation
  - JOptionPane object can do:
    - Creating a message window
    - Making a closing button

01. Interactive Input

CelsiusToFahrenheit.java

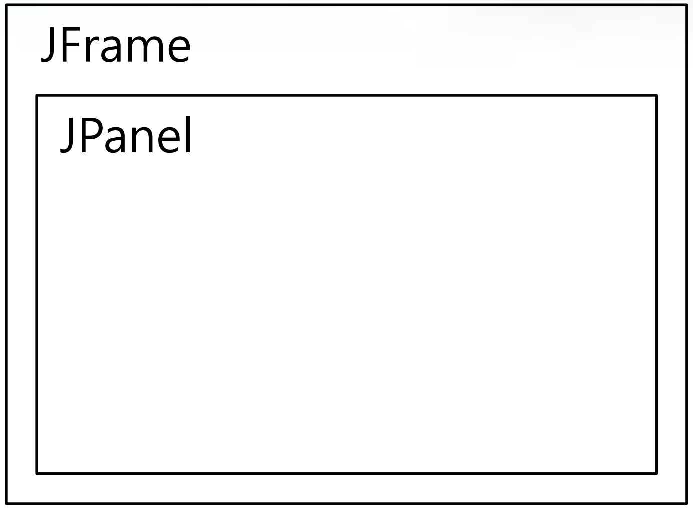
```
import java.text.*;
import javax.swing.*;

public class CelsiusToFahrenheit
{
    public static void main(String[] args)
    {
        String input = JOptionPane.showInputDialog("Type an
        integer Celsius temperature");
        int c = new Integer(input).intValue();
        double f = ((9.0/5.0)*c) + 32;
        DecimalFormat formatter = new DecimalFormat("0.0");
        JOptionPane.showMessageDialog(null, c + " Celsius is " +
        formatter.format(f) + " Fahrenheit");
    }
}
```

02. Graphical Output

Beautiful Window?

- Let’s use library!
    - javax.swing package

- Frame and panel
    - JFrame
    - JPanel



02. Graphical Output

Frame

- Frame creation
  - new JFrame()

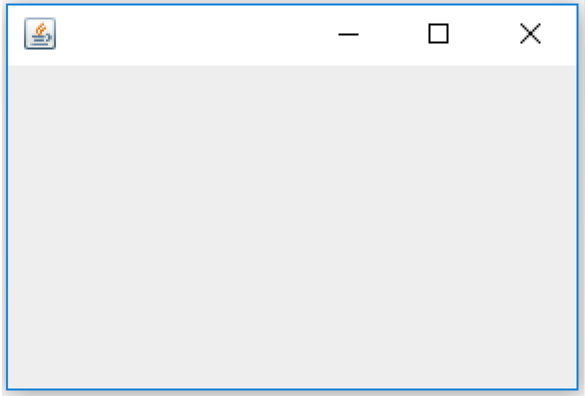
- To show..
  - <frame>.setVisible(true);

- Adjusting size
  - <frame>.setsize(<width>, <height>)
  - size = number of pixels

02. Graphical Output

FrameTest.java

```
import javax.swing.*;

public class FrameTest
{
    public static void main(String[] args)
    {
        JFrame f = new JFrame();
        f.setSize(300, 200);
        f.setVisible(true);
    }
}
```



[Empty Window]

02. Graphical Output

Panel

- For drawing
  - new JPanel()

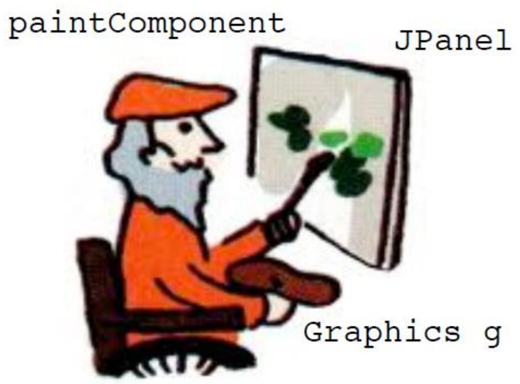
- A panel should be attached to a frame
  - <frame>.getContentPane().add(<panel>);

02. Graphical Output

Panel and Painter

- Panel has Painter
    - JPanel's method: paintComponent
- Painter can draw again
    - Calling paintComponent
- Painter needs a drawing tool
    - Drawing tool: Graphics object

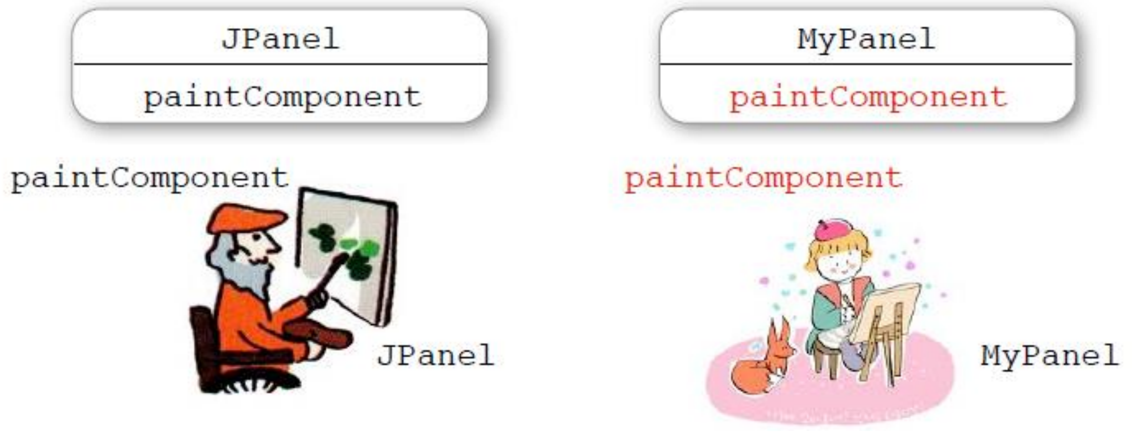
JPanel
paintComponent(Graphics g)



02. Graphical Output

MyPanel

- What if you want to create your own panel?
  - Changing the Painter
  - That is, paintComponent method needs to be replaced
- How?
  - "**inheritance**" and "**overriding**"



02. Graphical Output

Inheritance

- Extending by “**inheritance**”
  - public class A extends B { ... }
    - **Based on B, define “A”**
    - Including all the fields and methods of B

- **Overriding**
  - A method can be “re-defined”
  - Super-class is not used -> sub-class is used

02. Graphical Output

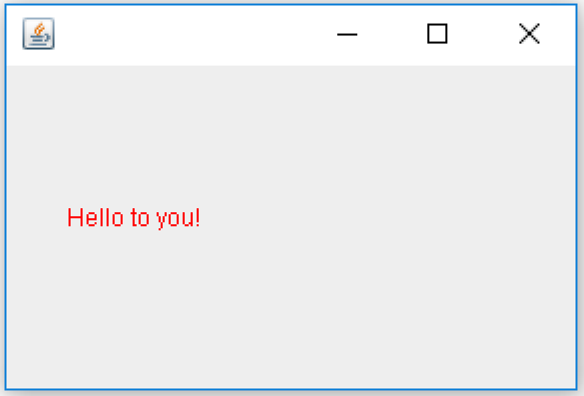
MyPanel

```
import java.awt.*;
import javax.swing.*;
                                        inheritance
public class MyPanel extends JPanel
{
    public void paintComponent(Graphics g)   overriding
    {
        painting code…
    }
}
```

- Only painter is changed!
    - paintComponent, by **overriding**

02. Graphical Output

New Panel



```
Hello to you!
```

02. Graphical Output

HANYANG UNIVERSITY

New Panel: MyPanel.java

```
import java.awt.*;     For graphics
import javax.swing.*; For JPanel

public class MyPanel extends JPanel
{
    public void paintComponent(Graphics g)
    {                                          "Graphics" provides various
        g.setColor(Color.red);                 drawing tools
        g.drawString("Hello to you!", 30, 80);
    }
}
```

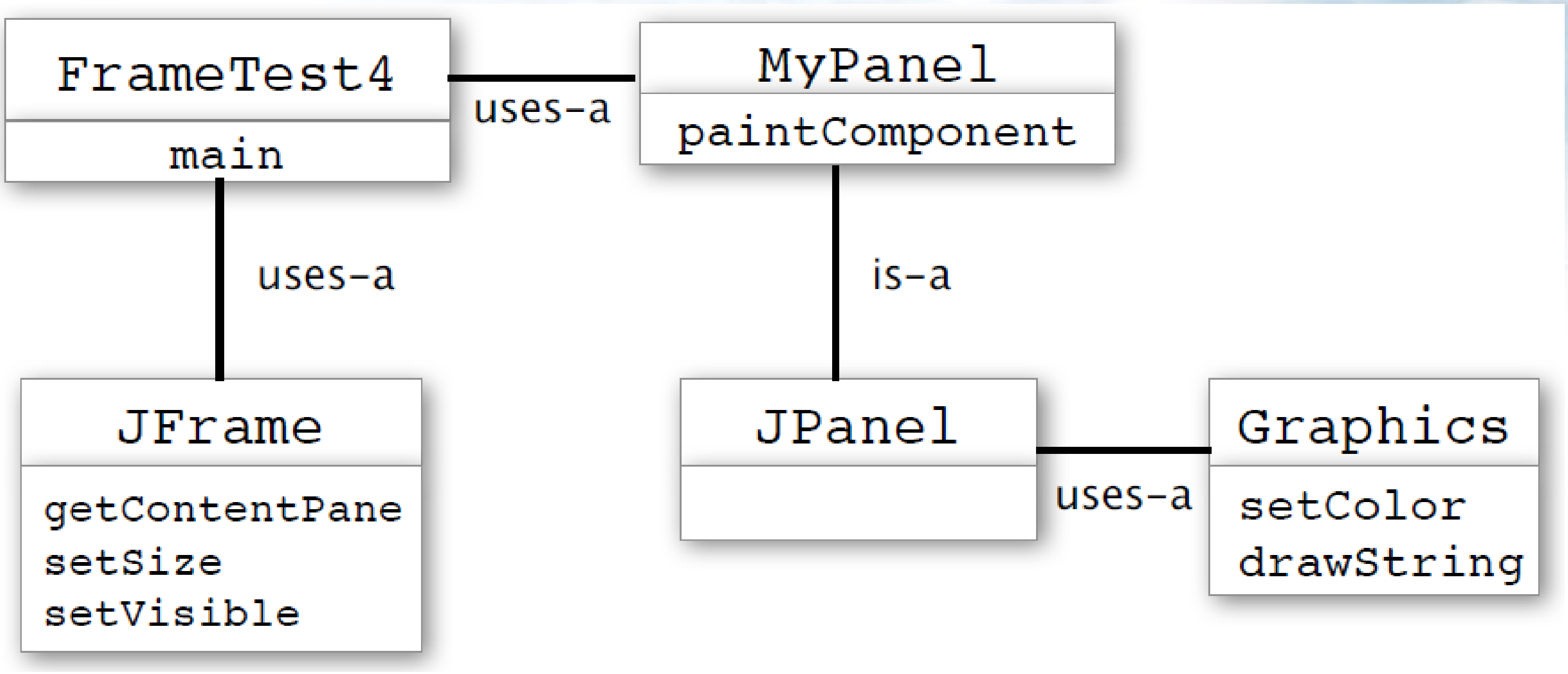
02. Graphical Output

Running: FrameTest4.java

```
import javax.swing.*;

public class FrameTest4
{
    public static void main(String[] args)
    {
        MyPanel p = new MyPanel();      Creating my panel
        JFrame f = new JFrame();        Creating a frame
        f.getContentPane().add(p);      Adding the panel to the frame
        f.setSize(300, 200);
        f.setVisible(true);
    }
}
```

02. Graphical Output

Diagram



* is-a relationship: if class C2 extends class C1, we say that C2 is a C1
* uses-a relationship: if class C2 sends messages to objects constructed from class C1, we say that C2 uses a C1.

03. Graphics

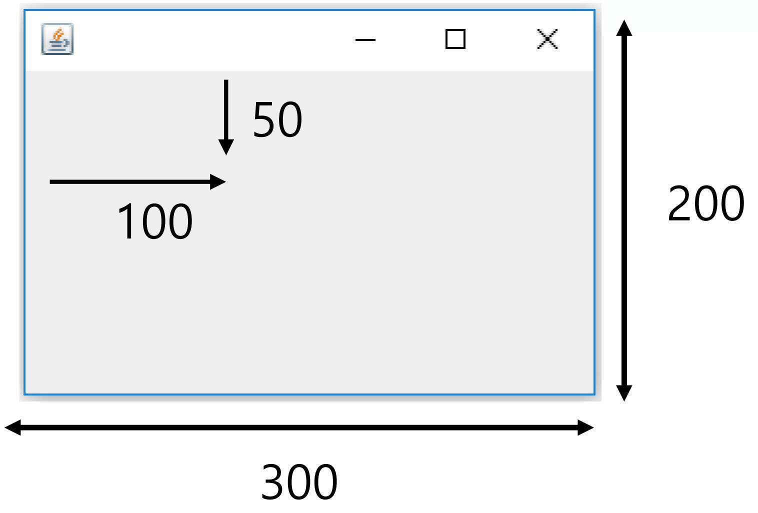
Graphics

- Graphics provides various drawing tools
  - setColor(c)
  - drawLine(x1,y1,x2,y2)
  - drawString(s,x,y)
  - drawRect/fillRect/drawOval/fillOval(x,y,dx,dy)
  - drawArc/fillArc(x,y,dx,dy,a,da)
  - paintImage(i,x,y,ob)
  - Etc.
- Please refer to documents!

03. Graphics

Locating a Pixel

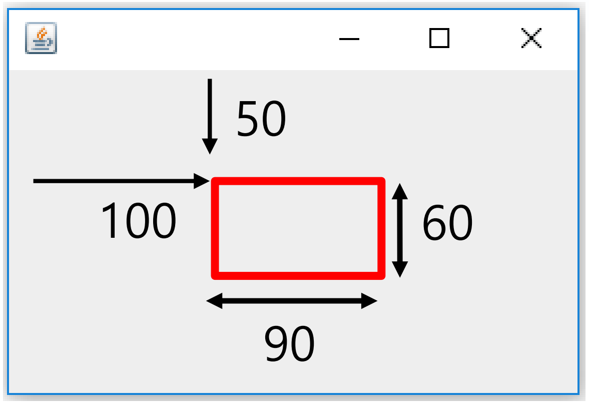
- If (width, height) = (300,200)
  - Where is the point (100, 50)?



03. Graphics

Drawing Rectangle

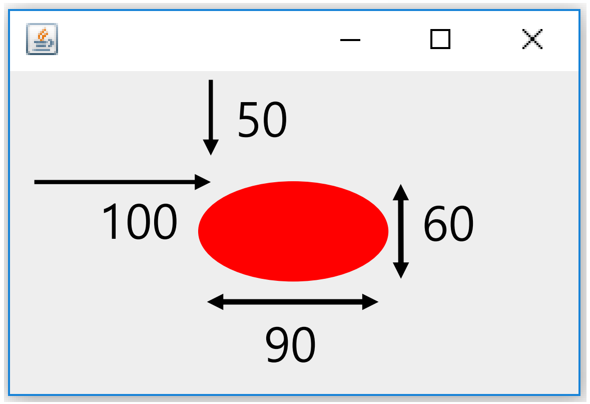
- g.drawRect(100, 50, 90, 60)



03. Graphics

Drawing Oval

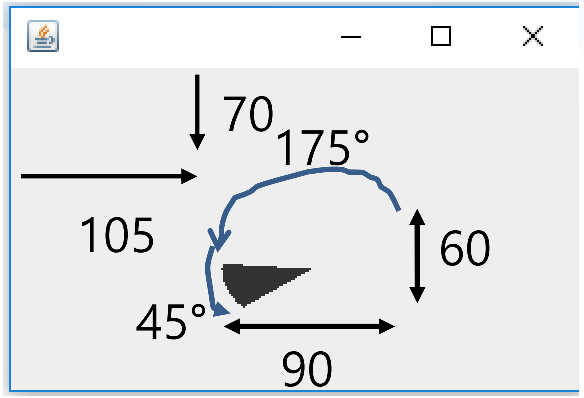
- g.fillOval(100, 50, 90, 60)



03. Graphics

Drawing Arc

- g.fillArc(105, 70, 90, 60, 175, 45)



03. Graphics

Constructor and this

- A panel should be attached to the “frame”
- How about (1) creating a frame and (2) attaching a panel **when the panel is created**?
- We need to know…
  - **Constructor** method
  - “**this**” object

03. Graphics

HANYANG UNIVERSITY

Constructor

- Called method when an object creates
- Initializing the object
- Constructor name = Class name

```
public class <class name>
{
        public <class name>()
        {
                Initialization code…
        }
}
```

03. Graphics

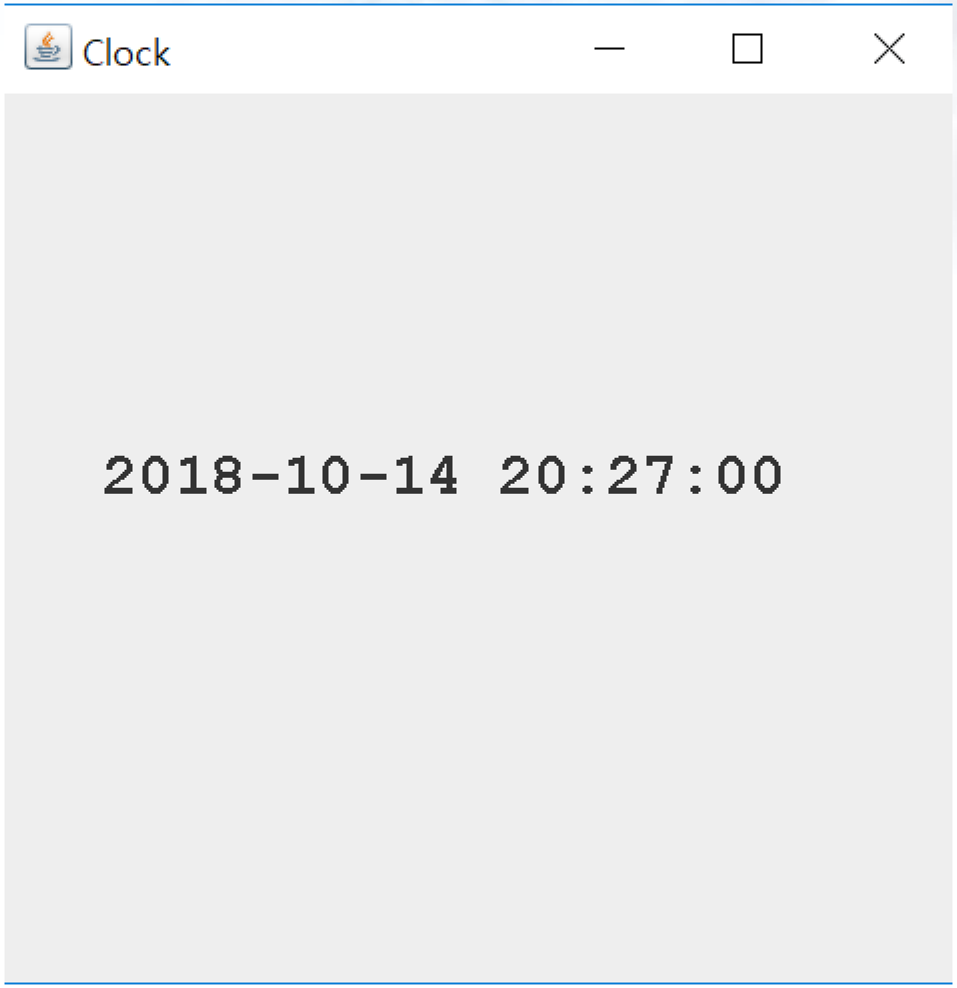
this

- For referring other objects
  - Create an object, and assign a name
    - String **s** = “HONG Gildong";
  - Getting the object as a parameter
    - public void paintComponent(Graphics **g**) { … }
- For referring self?
  - **this**

03. Graphics

Example: Clock

- Let's draw a simple clock



```
2018-10-14 20:27:00
```

03. Graphics

Constructor

```
public class ClockBasic extends JPanel
{
	public ClockBasic()
	{
	int width = 500;
	JFrame f = new JFrame();
	f.getContentPane().add(this);
	f.setTitle("Clock");
	f.setSize(width, width);
	f.setVisible(true);
	}

	public void paintComponent(Graphics g)
	{
		…
	}

	public static void main(String[] args)
	{
		new ClockBasic();
	}
}
```

03. Graphics

Drawing Time

```
public void paintComponent(Graphics g)
{
    GregorianCalendar time = new GregorianCalendar();
    DateFormat formatter = new SimpleDateFormat("yyyy-MM-dd
    HH:mm:ss");
    String result = formatter.format(time.getTime());

    Font myfont = new Font ("Courier New", 1, 30);
    g.setFont(myfont);
    g.drawString(result, 50, 200);
}
```

04. Objects with State: Field Variables

HANYANG UNIVERSITY

Field Variable

- An Object
  - is distinguishable (with memory address)
  - has information (through field variables)
  - calculates something (by methods)

- Field variables are shared/accessed by all the methods in the given object

04. Objects with State: Field Variables

HANYANG UNIVERSITY

Possible field variables

```
public class ClockBasic extends JPanel
{
        public ClockBasic()
        {
                int width = 200;
                …
        }

        public void paintComponent
        (Graphics g)
        {
                int width = 200;
                …
        }
}
```

```
public class ClockBasic extends JPanel
{
         int width = 200;
        public ClockBasic()
        {
                …
        }

        public void paintComponent
        (Graphics g)
        {
                …
        }
}
```

04. Objects with State: Field Variables

HANYANG UNIVERSITY

Field Variable

- A field variable exists only in the object
  - Accessible from the constructor of the object

- A field variable maintains a value
  - Even if when methods are not executed

04. Objects with State: Field Variables

Initialization value

- If any initial value is not assigned to a field variable, it is initialized
  - 0 for int, char
  - 0.0 for float, double
  - false for boolean
  - null for object

04. Objects with State: Field Variables

Local vs. Field variables

	Local variable	Field variable
Creation	When declared	When object is created
Destroy	When block is finished	When object is destroyed
Scope	Only in a block	Object

05. Summary

Summary

- Inheritance
  - public class <class name> extends <super class> { ... }
  - method overriding
- Constructor
- Self reference
  - **this**
- Field variable

Thanks

Week 4: Input, Output, and State
Instructor: Jinyoung Han (jinyounghan@hanyang.ac.kr)