CSE2016 Programming Methodology

Component Structure

Week 5: Class and Method

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

HANYANG UNIVERSITY

Contents

HANYANG UNIVERSITY

Today's Schedule

01. Methods

Methods

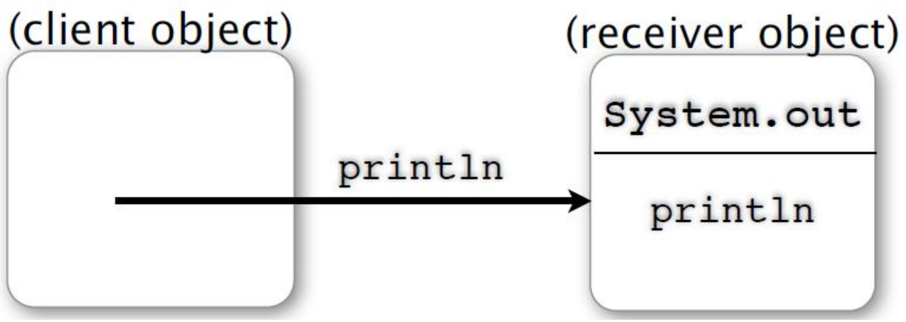
- In OOP,
  - an object owns a collection of methods for accomplishing work

- Public method
  - Method anyone can call
- Private method
  - Method only I can call
- Constructor
  - Method when an object is created
  - Public

02. Public Methods

Public Method

- Public method
  - Anyone can send a message for execution
  - A client sends a message for executing the method
  - A receiver receives the message and executes the method
    - E.g., System.out.println(“Hello!”);



02. Public Methods

An Example: ASCII Art

```
 ,-.
 \_/
>{|||}-
 / \
 `-^ hjw
```

[Bee]

```
    "
 _     _
(_\|/_)
 (\|/) ejm97
```

[Butterfly]

```
'm'
(|) sahr
```

[Ladybug]

```java
public void printBee()
{
    System.out.println(" ,-.");
    System.out.println(" \\_/"); // \ -> \\
    System.out.println(">{|||}-");
    System.out.println(" / \\");
    System.out.println(" `-^ hjw");
    System.out.println();
}
```

02. Public Methods

AsciiArtWriter

```java
public class AsciiArtWriter
{
    public AsciiArtWriter()
    {
        System.out.println();
    }

    public void printBee()
    {
        System.out.println(" ,-.");
        System.out.println(" \\_/"); // \ -> \\
        System.out.println(">{|||}-");
        System.out.println(" / \\");
        System.out.println(" `-^ hjw");
        System.out.println();
    }
```

```
 ,-.
 \_/
>{|||}-
 / \
 `-^ hjw
```

Continued..

02. Public Methods

AsciiArtWriter

```
    public void printButterfly()
    {
        System.out.println(" _ \""); // " -> \"
        System.out.println(" (_\\|/_)");
        System.out.println(" (/|\\) ejm97");
        System.out.println();
    }

    public void printLadybug()
    {
        System.out.println(" `m\'"); // ' -> \'
        System.out.println(" (|) sahr");
        System.out.println();
    }
}
```

```
 _ " _
(_\|/_)
 (\|/) ejm97
```

```
'm'
(|) sahr
```

02. Public Methods

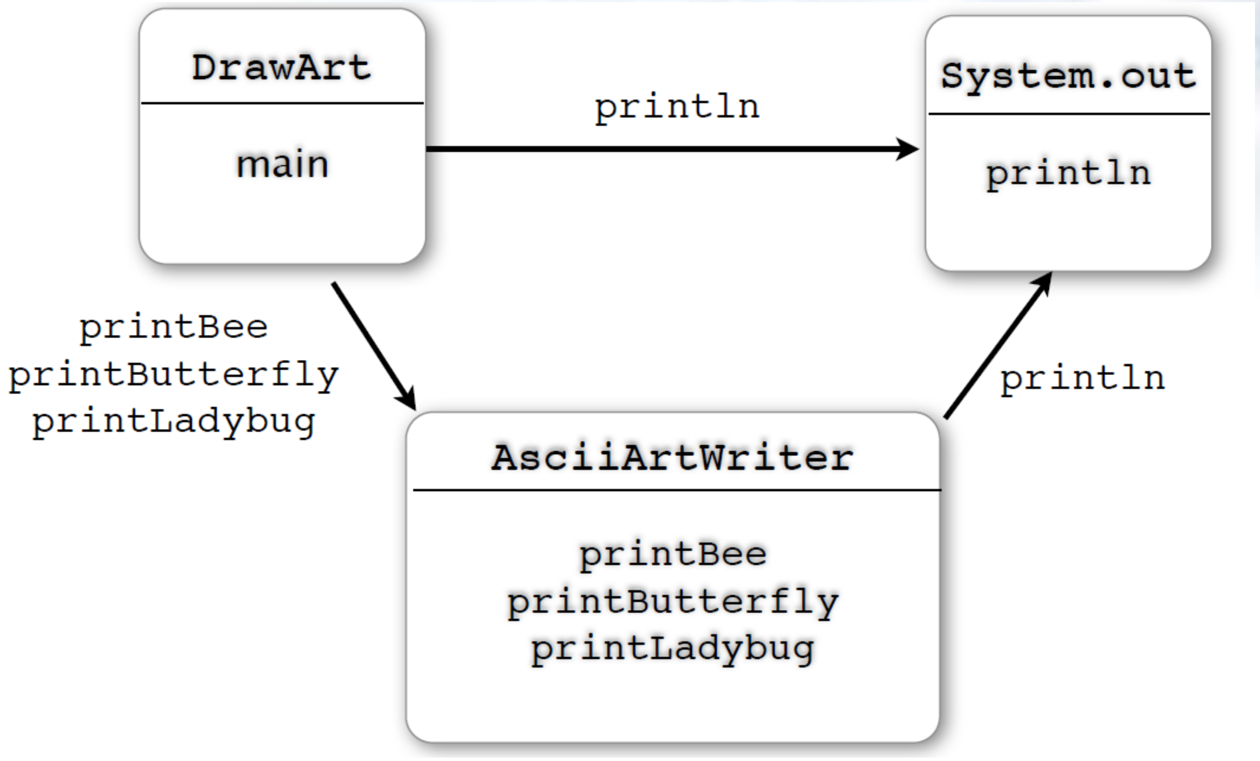
DrawArt

```
public class DrawArt
{
    public static void main(String[] args)
    {
        AsciiArtWriter writer = new AsciiArtWriter();
        writer.printBee();
        System.out.println("This is a test.");
        writer.printButterfly();
        writer.printLadybug();
    }
}
```

Since **printBee**, **printButterfly**, **printLadybug** are public methods, anyone can call them!

02. Public Methods

Class Diagram



03. Parameters to Methods

Parameters

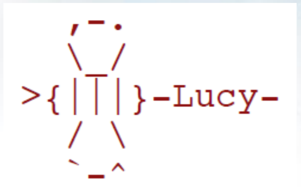
- When an object sends a message, the message often contains additional information in parentheses, called arguments
- A method receives values as “formal parameters”
  - Formal parameters are
    - Similar to local variables in terms of usage and scope
    - Initialized as the received values
  - The number and type of actual parameters should be same as the number and type of formal parameters

03. Parameters to Methods

An Example

```
public void printBeeWithName(String name)
{
    System.out.println("  ,-.");
    System.out.println("  \\_/");
    System.out.println(">{|||}-" + name + "-");
    System.out.println("  / \\");
    System.out.println("  `-^ hjw");
    System.out.println();
}
```

formal parameters



```
  ,-.
  \_/
>{|||}-Lucy-
  / \
  `-^
```

binding

```
AsciiArtWriter writer = new AsciiArtWriter();
writer.printBeeWithName("Lucy");
```

arguments or actual parameters

03. Parameters to Methods

HANYANG UNIVERSITY

printInverse

Parameter i: an integer value for calculating its inverse value

```
public void printInverse(int i)
{
    DecimalFormat formatter = new DecimalFormat("0.000");
    double d = 1.0 / i;
    String s = formatter.format(d);
    System.out.println(s);
}
```

```
MathOperations calculator = new MathOperations();
calculator.printInverse(3);
```

03. Parameters to Methods

printInverse

Parameter pattern: a pattern string for output

```
public void printInverse(int i, String pattern)
{
    DecimalFormat formatter = new DecimalFormat(pattern);
    double d = 1.0 / i;
    String s = formatter.format(d);
    System.out.println(s);
}
```

```
calculator.printInverse(3, "0.000000000");
```

03. Parameters to Methods

printInverse

Parameter pattern: a pattern format for output

```
public void printInverse(int i, DecimalFormat f)
{
    double d = 1.0 / i;
    String s = f.format(d);
    System.out.println(s);
}
```

```
DecimalFormat five_places = new DecimalFormat("0.00000");
calculator.printInverse(3, five_places);
```

03. Parameters to Methods

Overloading

- We can define methods, if type is different
  - printInverse(int i)
  - printInverse(int i, String pattern)
  - printInverse(int i, DecimalFormat f)

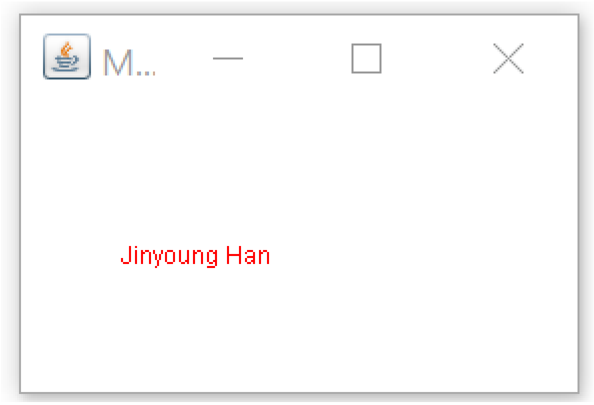
- When a method is called, we can differentiate with parameter types
  - printInverse(5)
  - printInverse(5, "0.00000")

- Method **overloading**!

04. Case Study: General-Purpose Output Frame

A Case Study

- A general-purpose output frame
  - A message is not pre-defined, later accepted
    - <output>.writeSentence(“Jinyoung Han”)
  - Output location is an input
    - <output>.repositionSentence(50,80)



04. Case Study: General-Purpose Output Frame

Specification

class MyWriter	
constructor:	
MyWriter(int w, int h)	Create a (w,h) window
private attributes:	
sentence	Output sentence
width, height	Window size
x_position, y_position	Location of output
methods:	
writeSentence(String s)	Print s
repositionSentence(int x, int y)	Locate at (x,y) and print

04. Case Study: General-Purpose Output Frame

Class MyWriter

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

public class MyWriter extends JPanel
{
    private int width;
    private int height;
    private String sentence = "";
    private int x_position;
    private int y_position;

    public MyWriter(int w, int h) { ... }
    public void paintComponent(Graphics g) { ... }
    public void writeSentence(String s) { ... }
    public void repositionSentence(int x, int y) { ... }
}
```

04. Case Study: General-Purpose Output Frame

Constructor

```
public MyWriter(int w, int h)
{
    width = w;
    height = h;
    x_position = width / 5;
    y_position = height / 2;

    JFrame f = new JFrame();
    f.getContentPane().add(this);
    f.setTitle("MyWriter");
    f.setSize(width, height);
    f.setVisible(true);
}
```

04. Case Study: General-Purpose Output Frame

Painter

```java
public void paintComponent(Graphics g)
{
    g.setColor(Color.red);
    g.drawString(sentence, x_position, y_position);
}
```

04. Case Study: General-Purpose Output Frame

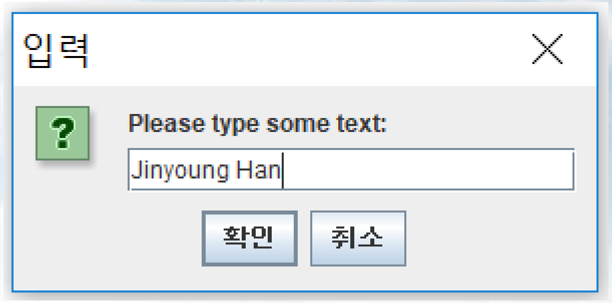
Methods

```
public void writeSentence(String s)
{
    sentence = s;
    this.repaint();
}

public void repositionSentence(int x, int y)
{
    x_position = x;
    y_position = y;
    writeSentence(sentence);
}
```

04. Case Study: General-Purpose Output Frame

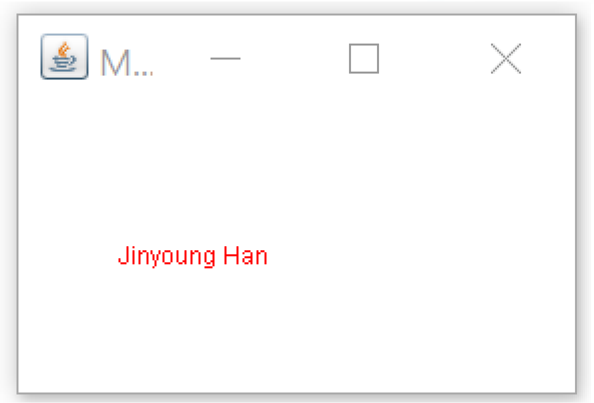
MyExample

```
import javax.swing.*;

public class MyExample
{
    public static void main(String[] args)
    {
        int x = 50;
        int y = 80;
        MyWriter w = new MyWriter(300, 200);
        w.repositionSentence(x, y);
        String s =
        JOptionPane.showInputDialog("Please type
        some text:");
        w.writeSentence(s);
    }
}
```


입력

Please type some text:

Jinyoung Han



Jinyoung Han

05. Results from Methods: Functions

Function

- A method that returns “results”
  - E.g., int c = new Integer(intput).intvalue();

- Return values can be:
  - basic types: int, char, double, boolean, etc.
  - object types: String, GregorianCalendar, Integer, etc.

05. Results from Methods: Functions

An Example

```
public double celsiusIntoFahrenheit(int c)
{
        double f = ((9.0/5.0)*c) + 32;
        return f;
}
```

```
public static void main(String[] args)
{
    String input = JOptionPane.showInputDialog("Type an
    integer Celsius temperature:");
    int c = new Integer(input).intValue();
    TemperatureConvertor convert = new
    TemperatureConvertor();
    double f = convert.celsiusIntoFahrenheit(c);
    System.out.println(f);
}
```

05. Results from Methods: Functions

HANYANG UNIVERSITY

Remark

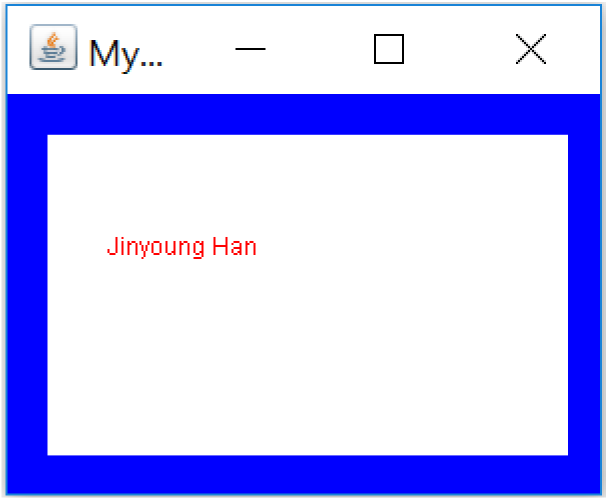
- Codes after “return” are not executed
  - “return” means the end of execution

- Taking a return value is not mandatory
  - convert.celsiusIntoFahrenheit(c);
  - can just be ignored..

06. Private Methods

An Example: GPOF

- General Purpose Output Frame ver. 2.0
  - Add a boarder
    - Using a private method



06. Private Methods

An Example: GPOF

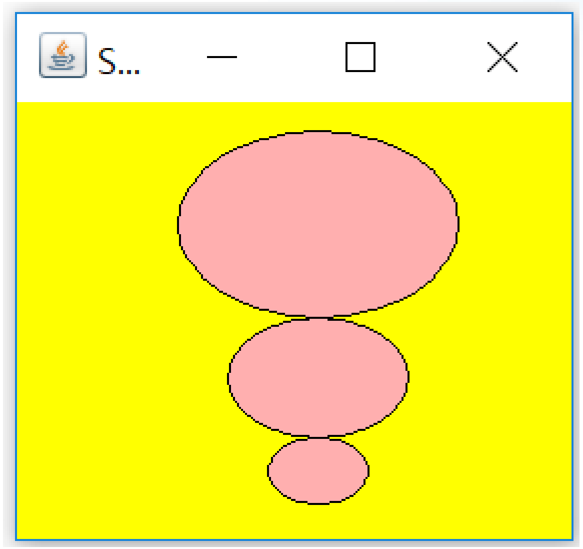
```
private void makeBorder(Graphics g)
{
    g.setColor(Color.blue);
    g.fillRect(0, 0, width, height);
    int border_size = 20;
    int center_rectangle_width = width - (2 * border_size);
    int center_rectangle_height = height - (2 * border_size);
    g.setColor(Color.white);
    g.fillRect(border_size, border_size, center_rectangle_width,
    center_rectangle_height);
}
```

```
public void paintComponent(Graphics g)
{
    makeBorder(g);
    g.setColor(Color.red);
    g.drawString(sentence, x_position, y_position);
}
```

06. Private Methods

An Example: Egg

- Stacked Eggs Writer
  - Call the paintAnEgg 3 times



06. Private Methods

StackedEggsWriter

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

public class StackedEggsWriter extends JPanel
{
    private int frame_width;
    private int frame_height;

    private int egg1_size;
    private int egg2_size;
    private int egg3_size;

    public StackedEggsWriter(int width, int height, int size1, int
    size2, int size3) { ... }
    private int paintAnEgg(int bottom, int width, Graphics pen) { ... }
    public void paintComponent(Graphics g) { ... }
    public static void main(String[] args) { ... }
}
```

06. Private Methods

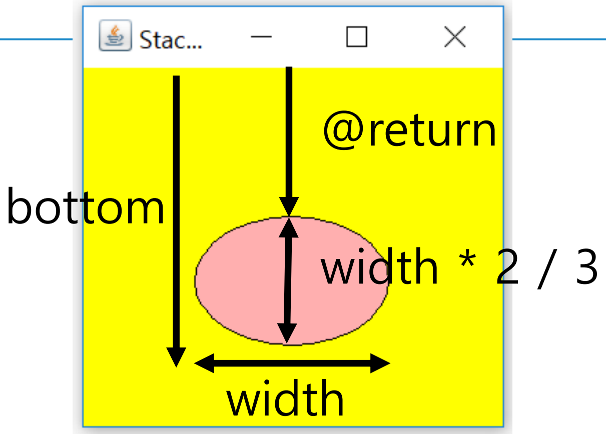
Constructor and Main

```
public StackedEggsWriter (int width, int height, int size1, int size2, int
size3)
{
    frame_width = width;
    frame_height = height;
    egg1_size = size1;
    egg2_size = size2;
    egg3_size = size3;
    JFrame my_frame = new JFrame();
    my_frame.getContentPane().add(this);
    my_frame.setTitle("StackedEggsWriter");
    my_frame.setSize(frame_width, frame_height);
    my_frame.setBackground(Color.yellow);
    my_frame.setVisible(true);
}

public static void main(String[] args)
{
    new StackedEggsWriter(300, 200, 50, 90, 140);
}
```

06. Private Methods

paintAnEgg

```
private int paintAnEgg(int bottom, int width, Graphics pen)
{
    int height = (2 * width) / 3;
    int top_edge = bottom - height;
    int left_edge = (frame_width - width) /2;
    pen.setColor(Color.pink);
    pen.fillOval(left_edge, top_edge, width, height);
    pen.setColor(Color.black);
    pen.drawOval(left_edge, top_edge, width, height);
    return top_edge;
}
```



06. Private Methods

Painter

```
public void paintComponent(Graphics g)
{
    int egg1_top = paintAnEgg(frame_height, egg1_size, g);
    int egg2_top = paintAnEgg(egg1_top, egg2_size, g);
    paintAnEgg(egg2_top, egg3_size, g);
}
```

06. Private Methods

Naming Rules

- Naming rules
  - class? method? varaible?
  - class: MyWriter, GregorianCalendar!
  - method: printBee, writeSentence!
  - formal parameters, field/local variables: answer, left_edge
    - If a field is used as a constant: FRAME_WIDTH

06. Private Methods

Static Methods

- Static methods can be called without creating an object
  - E.G., Math.abs(f)
  - Only static fields are accessible
  - Only static methods can be called

```
                Class A

static int a = ...;

static void f()
{
    .... a ...
}
```

07. Summary

Summary

- Method
  - public, private
- Parameter passing
- Function and its return value

Thanks

Week 5: Class and Method
Instructor: Jinyoung Han (jinyounghan@hanyang.ac.kr)