

## Encapsulation

This document introduces the technology of “Encapsulation”. And “Liskov Substitution Principle” will be introduced.

Quiz 3 is related to the exercises in "Programming Practice 1", so it is recommended that both be solved at the same time.

### 1. Encapsulation

In order to clarify the necessity of encapsulation, a system that draws Pac-Man is made first.

#### 1.1 Pacman Class

There are two classes of Painter class and PacMan class. Painter creates an instance of PacMan class, designates the size, mouth size and drawing position, then draws it. PacMan class prepares variables to record the size, display position and mouth size in the field, and has “make” method for drawing.

Figure 1 and Figure 2 show the program sources of Painter class and PacMan class, respectively.

In paint method of Painter class, the instance of PacMan class is made first (line 18). The size, the x and y coordinates of the center, and the mouth opening angle are assigned to the variables size, xCenter, yCenter and angle, respectively (lines 19 to 22). In line 23, “make” method of PacMan class is executed, and a pacman is drawn as shown in Figure 3. This picture shows the Pac-Man we usually think of.

Figure 4 shows the result of setting the angle of Pac-Man’s mouth to 160 degrees. That is, line 22 of Painter class is changed as follows.

```
pacman1.angle = 160;
```

The picture in Figure 4 is hard to say as Pac-Man. The reason for this is because the angle of Pac-Man’s mouse is set to 160 in Painter class.

For any setting, in order to make it look like Pac-Man, we need to do the following:

- (a) Instance variables will be hidden from other objects.
- (b) Instance variables can be accessed only through the methods of their current object.

```

Painter.java
1  /*
2   * Painter
3   * by Hitoshi Ogawa on October 25th, 2021
4   */
5
6  import java.awt.Graphics;
7  import javax.swing.JFrame;
8
9  public class Painter extends JFrame {
10
11     public Painter() {
12         setSize(400,300);    // setting window size
13         setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
14         setVisible(true);
15     }
16
17     public void paint(Graphics g) {
18         PacMan pacman1 = new PacMan();
19         pacman1.size = 100;
20         pacman1.xCenter = 200;
21         pacman1.yCenter = 150;
22         pacman1.angle = 60;
23         pacman1.make(g);
24     }
25
26     public static void main(String[] args) {
27         new Painter();
28     }
29 }

```

Figure 1. Painter class.

```

PacMan.java
1  /*
2   * PacMan
3   * by Hitoshi Ogawa on October 25th, 2021
4   */
5
6  import java.awt.Color;
7  import java.awt.Graphics;
8
9  public class PacMan {
10     int size;
11     int xCenter;
12     int yCenter;
13     int angle;
14
15     public void make(Graphics g) {
16         g.setColor(Color.yellow);
17         g.fillArc(xCenter - size / 2, yCenter - size / 2,
18                 size, size, angle / 2, 360 - angle);
19     }
20 }

```

Figure 2. PacMan class.

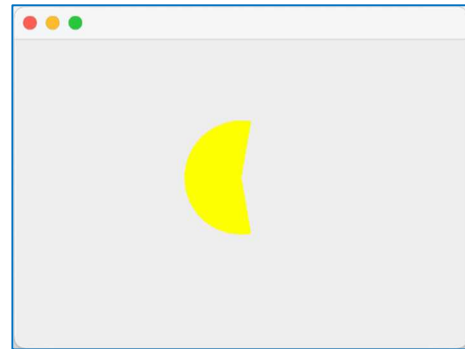
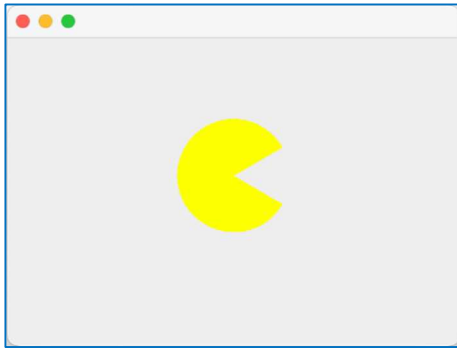


Figure 3. A pacman whose angle is 60.      Figure 4. A pacman whose angle is 160.

This is called “encapsulation”. To achieve encapsulation in Java, we need “controlling access” and “setter method.” (See Chapter 2 of the third day document “Inheritance (1)” for “Access Control”.

In order to realize (a), we use the “private” modifier as follows.

```
private int size;  
private int xCenter;  
private int yCenter;  
private int angle;
```

## 1. 2 Getter and Setter Methods

If a data member is declared “private”, then it can only be accessed within the same class. No outside class can access data member of that class. If we need to access these variables, we have to use public or no modifier “getter” and “setter” methods.

A getter method is a method that other classes can retrieve hidden data. In the case of getter methods, since there is no restriction under normal circumstances, it can be easily described. In case of PacMan class, it can be described as follows.

```
public int getSize() {  
    return(size);  
}  
  
public int getXCenter() {  
    return (xCenter);  
}
```

```
public int getYCenter() {  
    return(yCenter);  
}  
  
public int getAngle() {  
    return(angle);  
}
```

A setter method is a method that can assign values to data hidden from other classes. This makes it possible to properly correct problematic data. In the case of angle of PacMan class, we can create setAngle method that assigns an appropriate value to the variable angle. If the given value is greater than 90 degrees, 90 is substituted for variable angle. Otherwise, the value is assigned to variable angle.

The setAngle method is shown as an example for angle. To simplify this method, it is assumed that a positive number is input to the variable num.

```
public void setAngle(int num) {  
    if(num > 90) {  
        angle = 90;  
    } else {  
        angle = num;  
    }  
}
```

It is necessary to determine the behavior of the following three setter methods that change the value of size so that Pac-Man does not always protrude from the window. Let the width, height, and bar represent the width and height of the window and the height of the title bar of the window, respectively.

setXCenter(int num): Assign num to variable xCenter.

When the size is known, it is checked whether Pac-Man protrudes from the window. If it protrudes, change the size. Since the initial value of variable size is 0, a value is given unless the value is 0.

If variable num is smaller than  $\text{size} / 2$ , the value of size needs to be change to  $\text{num} * 2$ . If  $(\text{num} + \text{size} / 2)$  is larger than width, the value of size needs to be change to  $(\text{width} - \text{num}) * 2$ .

setYCenter(int num): Assign num to variable yCenter.

When the size is known, it is checked whether Pac-Man protrudes from the screen. If it protrudes, change the size. Since the initial value of variable size is 0, a value is given unless the value is 0.

If variable  $(\text{num} - \text{size} / 2)$  is smaller than bar, the value of size needs to be change to  $(\text{num} - \text{bar}) * 2$ . If  $(\text{num} + \text{size} / 2)$  is larger than height, the value of size needs to be change to  $(\text{height} - \text{num}) * 2$ .

`setSize(int num)`: When the coordinates of the center are known, it is checked whether Pac-Man protrudes from the screen. If it protrudes, change the size.  
If `xCenter` is given a value (not 0), use `setXCenter(xCenter)`.  
If `yCenter` is given a value (not 0), use `setYCenter(xCenter)`.

## 2. Three Kinds of Pac-Man

Let's create three kinds of Pac-Man as shown in Figure 5. Let's name them `PacMan`, `PacManWithEye` and `PacManWithSmile` from the left. The easiest way is that `PacManWithEye` and `PacManWithSmile` inherit `PacMan`. They can use variables and methods of `PacMan`. We can create programs simply by preparing eye or laughter eye. The program `Painter` that rendered Figure 5 is shown in Figure 6.

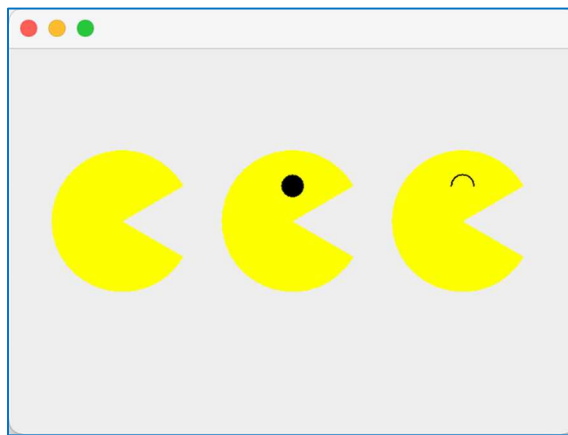


Figure 5. Three Kinds of Pac-Man

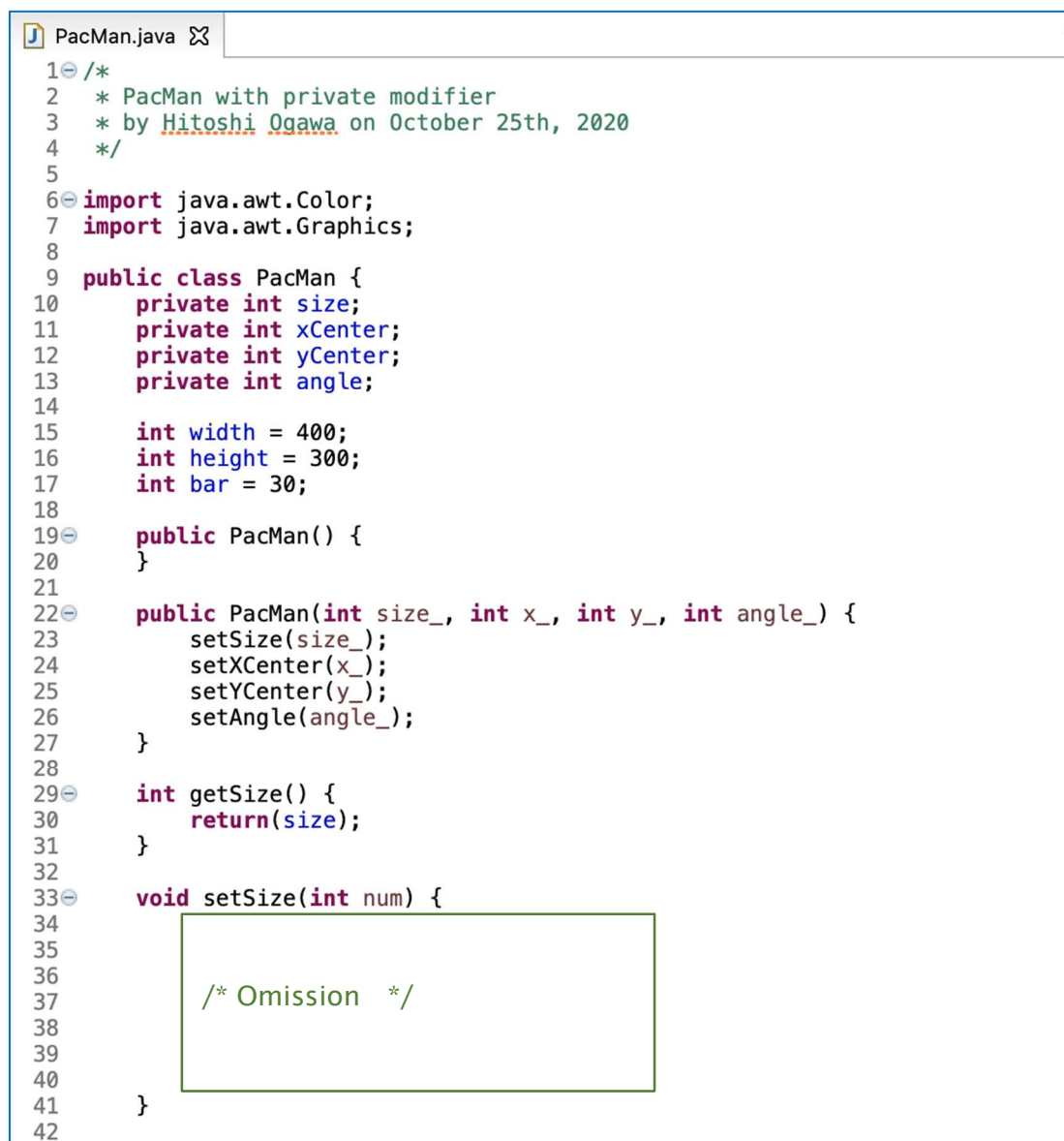
```
Painter.java
1  /*
2   * Painter to display three types of PacMan
3   * by Hitoshi Ogawa on October 25th, 2021
4   */
5
6  import java.awt.Graphics;
7  import javax.swing.JFrame;
8
9  public class Painter extends JFrame {
10     public Painter() {
11         setSize(400,300); // setting window size
12         setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
13         setVisible(true);
14     }
15
16     public void paint(Graphics g) {
17         PacMan pacman1 = new PacMan(100, 80, 150, 60);
18         pacman1.make(g);
19         PacManWithEye pacman2 = new PacManWithEye(100, 200, 150, 60);
20         pacman2.make(g);
21         PacManWithSmile pacman3 = new PacManWithSmile(100, 320, 150, 60);
22         pacman3.make(g);
23     }
24
25     public static void main(String[] args) {
26         new Painter();
27     }
28 }
```

Figure 6. Painter that rendered Figure 5.

Using the constructor, data is set with one instruction. The drawing instruction is make method.

The source of PacMan class used here is shown in Figure 7. The four variables have private modifiers. For this reason, setter methods for setting a value and getter method for referring to a value are prepared. The omitted parts are related to Quiz 3 and the exercises in "Programming Practice 1". Consider yourself with reference to Section 1.2

Since both PacManWithEye and PacManWithSmile add two types of eyes to PacMan, it seems easier if they inherit PacMan. Figure 8 and Figure 9 show PacManWithEye class and PacManWithSmile class respectively.



```
1  /*
2  * PacMan with private modifier
3  * by Hitoshi Ogawa on October 25th, 2020
4  */
5
6  import java.awt.Color;
7  import java.awt.Graphics;
8
9  public class PacMan {
10     private int size;
11     private int xCenter;
12     private int yCenter;
13     private int angle;
14
15     int width = 400;
16     int height = 300;
17     int bar = 30;
18
19     public PacMan() {
20     }
21
22     public PacMan(int size_, int x_, int y_, int angle_) {
23         setSize(size_);
24         setXCenter(x_);
25         setYCenter(y_);
26         setAngle(angle_);
27     }
28
29     int getSize() {
30         return(size);
31     }
32
33     void setSize(int num) {
34
35
36
37         /* Omission */
38
39
40     }
41
42 }
```

Figure 7. The source of PacMan class.

```

43  int getXCenter() {
44      return(xCenter);
45  }
46
47  void setXCenter(int num) {
48      

49          /* Omission */
50
51
52
53
54
55
56


57  }
58  int getYCenter() {
59      return(yCenter);
60  }
61
62  void setYCenter(int num) {
63      

64          /* Omission */
65
66
67
68
69
70
71


72  }
73  int getAngle() {
74      return(angle);
75  }
76
77  void setAngle(int num) {
78      if(num > 90) {
79          angle = 90;
80      } else {
81          angle = num;
82      }
83  }
84
85  void make(Graphics g) {
86      g.setColor(Color.yellow);
87      g.fillArc(xCenter - size / 2, yCenter - size / 2,
88              size, size, angle / 2, 360 - angle);
89  }
90 }

```

Figure 7. The source of PacMan class (continue).

```

1  /*
2   * PacManWithEye that inherits PacMan
3   * by Hitoshi Ogawa on October 25th, 2021
4   */
5
6  import java.awt.Color;
7  import java.awt.Graphics;
8
9  public class PacManWithEye extends PacMan {
10
11     public PacManWithEye(int size_, int x_, int y_, int angle_) {
12         super(size_, x_, y_, angle_);
13     }
14
15     public void make(Graphics g) {
16         int xCenter = getXCenter();
17         int yCenter = getYCenter();
18         int size = getSize();
19         super.make(g);
20         g.setColor(Color.black);
21         g.fillOval(xCenter - size / 12, yCenter - size / 3,
22                  size / 6, size / 6);
23     }
24 }

```

Figure 8. The source of PacManWithEye.

```

1  /*
2   * PacManWithWink that inherits PacMan
3   * by Hitoshi Ogawa on October 25th, 2021
4   */
5
6  import java.awt.Color;
7  import java.awt.Graphics;
8
9  public class PacManWithSmile extends PacMan{
10
11     public PacManWithSmile(int size_, int x_, int y_, int angle_) {
12         super(size_, x_, y_, angle_);
13     }
14
15     public void make(Graphics g) {
16         int xCenter = getXCenter();
17         int yCenter = getYCenter();
18         int size = getSize();
19         super.make(g);
20         g.setColor(Color.black);
21         g.drawArc(xCenter - size / 12, yCenter - size / 3,
22                  size / 6, size / 6, 0, 180);
23     }
24 }

```

Figure 9. The source of PacManWithEye.



### 3. Liskov Substitution Principle

The “Liskov Substitution Principle”, is simple, yet powerful concept that can be used to improve your design. It states:

“if S is a subtype of T, then objects of type T in a program may be replaced with objects of type S without altering any of the desirable properties of the program.”

All this really means is that objects in a program should be replaceable with instances of their subtypes without altering the correctness of that program.

In Pac-Man’s example, PacManWithEye class and PacManWithSmile class are subclasses of PacMan class. So, we need to indicate that replacing PacMan object with PacManWithEye object or PacManWithSmile object.

Let’s change paint method of Painter class shown in Figure 6. That is, we replace PacMan with PacManWithEye on line 17 as follows.

```
PacMan pacman1 = new PacManWithEye (100, 80, 150, 60);
```

The execution result is shown in Figure 10.

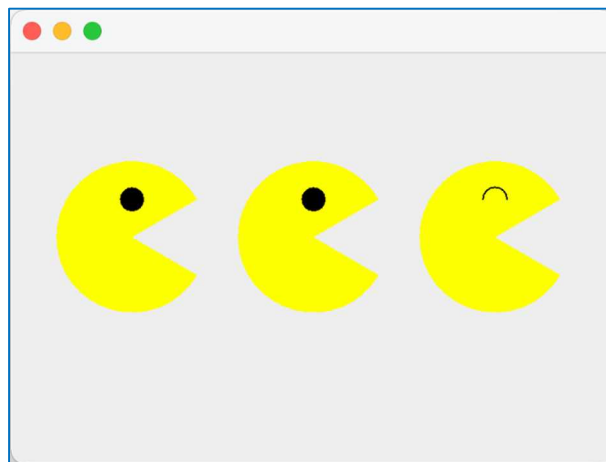


Figure 10. The result of replacing PacMan with PacManWithEye.

### 4. Class Hierarchy

Figure 10 shows that the programs shown in Figure 7, 8, 9 violated Liskov Substitution Principle. The reason is that it does not keep the class hierarchy. PacManWithEye and PacManWithSmile share the same look as PacMan, but they are not a kind of the PacMan type in the concept of "Liskov Substitution Principle".

It is better to think that we had better to make superclass common to the three classes. In this case, let’s consider “Face” class.

The source of Face class is shown in Figure 11.

```

1  /*
2   * Face which is an upper class of three types of PacMan
3   * by Hitoshi Ogawa on October 25th, 2021
4   */
5
6  import java.awt.Color;
7  import java.awt.Graphics;
8
9  public class Face {
10     private int size;
11     private int xCenter;
12     private int yCenter;
13     private int angle;
14
15     int width = 400;
16     int height = 300;
17     int bar = 30;
18
19     public Face() {
20     }
21
22     public Face(int size_, int x_, int y_, int angle_) {
23         setSize(size_);
24         setXCenter(x_);
25         setYCenter(y_);
26         setAngle(angle_);
27     }
28
29     int getSize() {
30         return(size);
31     }
32
33     void setSize(int num) {
34
35
36         /* Omission */
37
38
39
40     }
41
42
43     int getXCenter() {
44         return(xCenter);
45     }
46
47     void setXCenter(int num) {
48
49
50         /* Omission */
51
52
53
54
55     }
56
57

```

Figure 11. The source of Face class.

```

57
58-   int getYCenter() {
59       return(yCenter);
60   }
61
62-   void setYCenter(int num) {
63       

64           /* Omission */
65


66
67
68
69
70
71   }
72
73-   int getAngle() {
74       return(angle);
75   }
76
77-   void setAngle(int num) {
78       if(num > 90) {
79           angle = 90;
80       } else {
81           angle = num;
82       }
83   }
84
85-   void make(Graphics g) {
86       g.setColor(Color.yellow);
87       g.fillArc(xCenter - size / 2, yCenter - size / 2,
88               size, size, angle / 2, 360 - angle);
89   }
90 }

```

Figure 11. The source of Face class (continue).

The PacMan class turns into a simple description as shown in Figure 12 by inheriting Face. The head statements for PacManWithEye and PacManWithSmile are rewritten as follows.

```

9 public class PacManWithEye extends Face {

```

```

9 public class PacManWithSmile extends Face {

```

```
PacMan.java
1  /*
2   * PacMan that inherits Face
3   * by Hitohi Ogawa on October 25th, 2021
4   */
5
6  import java.awt.Graphics;
7
8  public class PacMan extends Face{
9
10     public PacMan(int size_, int x_, int y_, int angle_) {
11         super(size_, x_, y_, angle_);
12     }
13
14     void make(Graphics g) {
15         super.make(g);
16     }
17 }
```

Figure 12. PacMan class that inherits Face class.

When PacMan variable is assigned an instance of PacManWithEye that inherits Face class, an error is displayed in Painter class as shown in Figure 13. Therefore, on line 17, we must create an instance of “PacMan” as follows:

```
17      PacMan pacman1 = new PacMan(100, 80, 150, 60);
```

```
*Painter.java
1  /*
2   * Painter to display three types of PacMan
3   * by Hitoshi Ogawa on October 25th, 2021
4   */
5
6  import java.awt.Graphics;
7  import javax.swing.JFrame;
8
9  public class Painter extends JFrame {
10     public Painter() {
11         setSize(400,300); // setting window size
12         setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
13         setVisible(true);
14     }
15
16     public void paint(Graphics g) {
17         PacMan pacman1 = new PacManWithEye(100, 80, 150, 60);
18         pacman1.make(g);
19         PacManWithEye pacman2 = new PacManWithEye(100, 200, 150, 60);
20         pacman2.make(g);
21         PacManWithSmile pacman3 = new PacManWithSmile(100, 320, 150, 60);
22         pacman3.make(g);
23     }
24
25     public static void main(String[] args) {
26         new Painter();
27     }
28 }
```

Figure 13. The error message is  
“Type mismatch: cannot convert from PacManWithEye to PacMan”.

## 5. Quiz 3

Enter the answer to the following question in Title "Quiz 3" on the Tests tab, manaba+R. The test tab will open from 12:55 on October 26th. The deadline for submission is 18:00 on October 28th. The answer will be published after the submission deadline.

Let the variables width, height, and bar represent the width and height of the window and the height of the title bar of the window, respectively.

Enter the variable name if it is available, otherwise enter the appropriate number from (a) to (d) to complete the three methods of the Face class. Note that the same value may be entered in multiple places.

```
33 void setSize(int num) {
34     size = num;
35     if(xCenter != (a)) {
36         setXCenter(xCenter);
37     }
38     if(yCenter != (a)) {
39         setYCenter(yCenter);
40     }
41 }
```

```
47 void setXCenter(int num) {
48     xCenter = num;
49     if(size != 0) {
50         if(num - size / 2 < 0) {
51             size = num * 2;
52         } else if(num + size / 2 > (b)) {
53             size = ((b) - num) * 2;
54         }
55     }
56 }
```

```
62 void setYCenter(int num) {
63     yCenter = num;
64     if(size != 0) {
65         if(num - size / 2 < (c)) {
66             size = (num - (c)) * 2;
67         } else if(num + size / 2 > (d)) {
68             size = ((d) - num) * 2;
69         }
70     }
71 }
```