

## Polymorphism

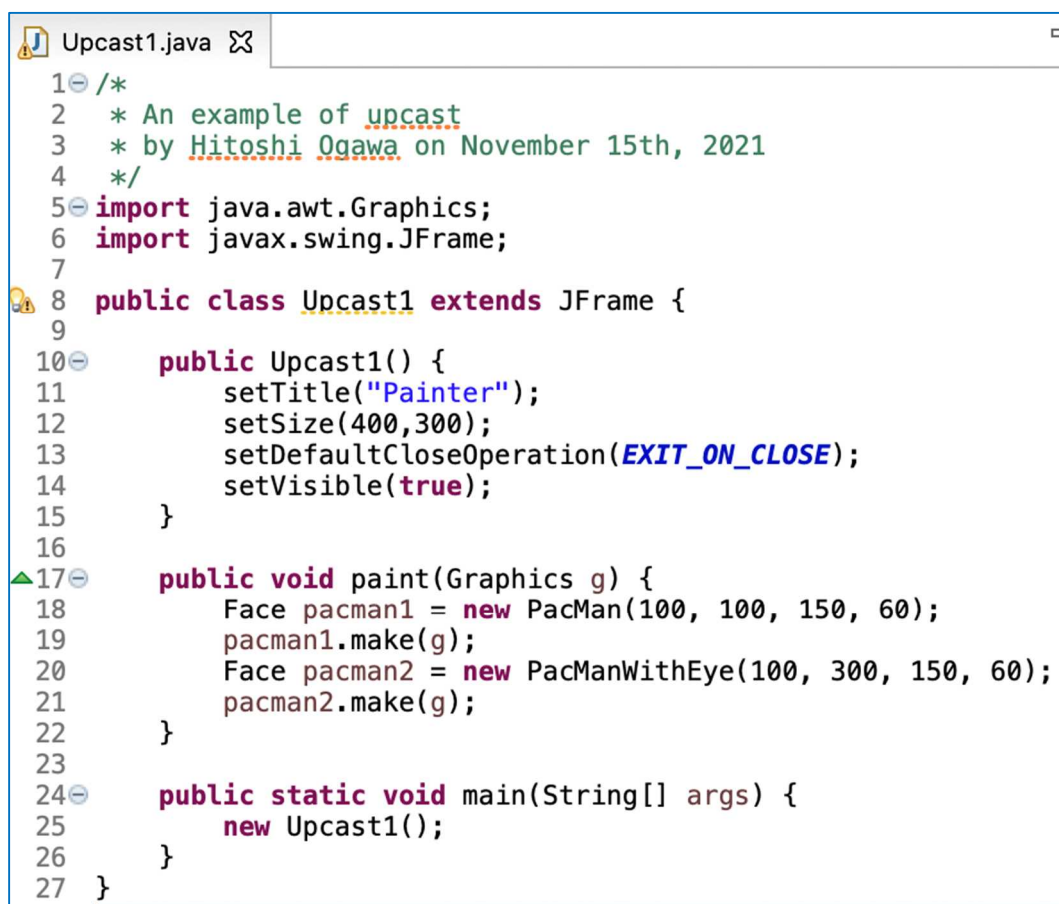
### 1. Upcasting

Upcasting is important part of Java, which allows us to build complicated programs using simple syntax, and gives us great advantages. Java permits an object of a subclass type to be treated as an object of any superclass type. This is called upcasting.

In the case of PacMan program that inherits Face, instead of assigning the object of PacMan to PacMan variable, we can assign it to Face variable as follows.

```
Face pacman = new PacMan(100, 200, 150, 90);
```

Figure 1 shows a program that uses upcasting to draw PacMan an PacManWithEye. The execution result is shown in Figure 2.



```
1  /*
2   * An example of upcast
3   * by Hitoshi Ogawa on November 15th, 2021
4   */
5  import java.awt.Graphics;
6  import javax.swing.JFrame;
7
8  public class Upcast1 extends JFrame {
9
10     public Upcast1() {
11         setTitle("Painter");
12         setSize(400,300);
13         setDefaultCloseOperation(EXIT_ON_CLOSE);
14         setVisible(true);
15     }
16
17     public void paint(Graphics g) {
18         Face pacman1 = new PacMan(100, 100, 150, 60);
19         pacman1.make(g);
20         Face pacman2 = new PacManWithEye(100, 300, 150, 60);
21         pacman2.make(g);
22     }
23
24     public static void main(String[] args) {
25         new Upcast1();
26     }
27 }
```

Figure 1. An example program using upcasting.

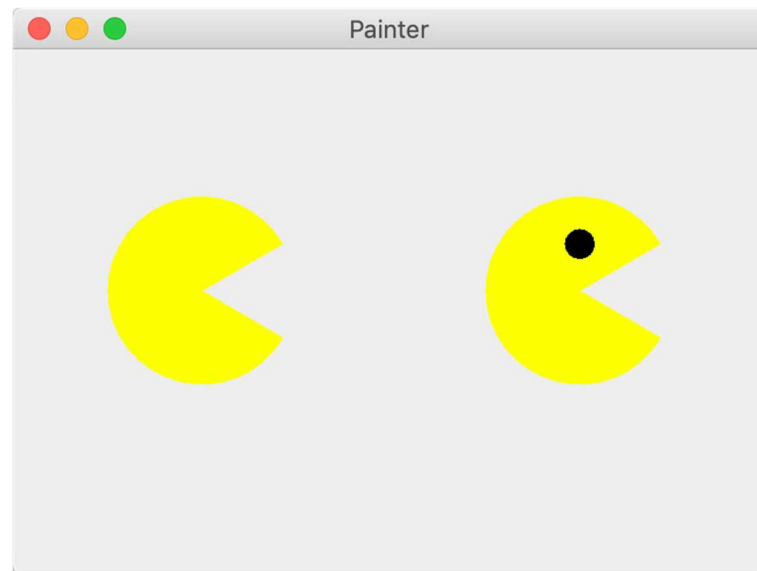


Figure 2. The result of Upcast1.java.

In the program “Upcast1.java”, the data type of variables `pacman1` and `pacman2` are both `Face`. However, the results of executing the `make` method is different `Pacman`. What this means is that even for variables of the same data type, the method of the assigned object is used at run time. The `make` method of `PacMan` is executed in the line 19, and the `make` method of `PacManWithEye` is executed in the line 21.

In the program “Upcast1.java”, `Face`, `PacMan` and `PacManWithEye` have the same methods. Let’s consider a case where they have different methods. Let’s add “`showData`” method to the `PacMan` class and `int` variable “`special`” as shown in Figure 3.

In the program “Upcast2.java”, we set the `Face` data type variable `pacman1` and the `PacMan` data type variable `pacman2` as shown in Figure 4. Let’s call the `showData` method, and refer the variable `special` with both variables `pacman1` and `pacman2`. An error occurs in `pacman1` (lines 20 and 21). The error messages are “The method `showData()` is undefined for the type `Face`” and “`special` cannot be resolved or is not a field”. The variable `pacman2` can be used to invoke `showData` method and refer the variable `special`, but `pacman1` cannot be used.

In the examples above, we can see the following.

- ✓ Available methods and variables are determined by data type.
- ✓ When the method is executed, the method of the assigned object is used.

```

1  /*
2   * PacMan with "showData" method and variable "special"
3   * by Hitoshi Ogawa on November 15th, 2021
4   */
5  import java.awt.Color;
6  import java.awt.Graphics;
7
8  public class PacMan extends Face{
9      int special = 7;
10
11     public PacMan(int size_, int x_, int y_, int angle_) {
12         super(size_, x_, y_, angle_);
13     }
14
15     void make(Graphics g) {
16         super.make(g);
17     }
18
19     void showData() {
20         System.out.println("The size is " + getSize());
21         System.out.println("The coodinate of center is ("
22             + getXCenter() + ", " + getYCenter() + ")");
23     }
24 }

```

Figure 3. PacMan class with “showData” method and variable “special” added.

```

1  /*
2   * An example of available methods and variables
3   * determined by data type
4   * by Hitoshi Ogawa on November 15th, 2021
5   */
6  import java.awt.Graphics;
7  import javax.swing.JFrame;
8
9  public class Upcast2 extends JFrame {
10     public Upcast2() {
11         setTitle("Painter");
12         setSize(400,300);
13         setDefaultCloseOperation(EXIT_ON_CLOSE);
14         setVisible(true);
15     }
16
17     public void paint(Graphics g) {
18         Face pacman1 = new PacMan(100, 100, 150, 60);
19         pacman1.make(g);
20         pacman1.showData();
21         System.out.println(pacman1.special);
22         PacMan pacman2 = new PacMan(100, 300, 150, 60);
23         pacman2.make(g);
24         pacman2.showData();
25         System.out.println(pacman2.special);
26     }
27
28     public static void main(String[] args) {
29         new Upcast2();
30     }
31 }

```

Figure 4. Upcast2.java

## 2. Polymorphism

Polymorphism is the ability of an object to take on many forms. The most common use of polymorphism in OOP occurs when a superclass is used to refer to a subclass object.

Polymorphism is a mechanism that implements different behaviors depending on subclasses. We can invoke the same name method on multiple instances and invoke the appropriate implementation for each. By realizing polymorphism, we can call subclass methods exactly in the same way as object-oriented reusability increases.

In order to realize polymorphism, we inherit classes to define common methods. By using variables of the same type, different objects can be executed differently in the same operation.

To realize polymorphism using inheritance is as follows.

- (1) Create one superclass and define multiple subclasses. By overriding the method M of the superclass with each subclass, we can create methods with different processing contents.
- (2) In the calling class, prepare the variable V to assign the superclass.
- (3) Assign the instance of the subclass to be executed to the variable V.
- (4) Execute method M of subclass: V.M

When inheritance is used, it is necessary to observe Riskov's substitution principle. It is necessary for the programmer to decide whether to keep this principle.

## 3. Pacman System

An example program "Painter.java" of drawing two kinds of Pacman using polymorphism is introduced. The following functions are realized.

- (a) Using Face array, ten Pacman of PacMan or PacManWithEye are drawn.
- (b) The position of the center is decided by mouse press.
- (c) The size of Pacman can be determined by dragging.

The source of the program "Painter.java" is shown in Figure 5.

The following variables used for the whole program are declared in the field.

- ✓ Array "face" of Face type whose size is 10 (line 16).
- ✓ Variable "count" managing the number of Pacman (line 17). Each time the mouse is pressed, the count is increased by 1, so the initial value is -1.
- ✓ Variable "flag" that specifies the type of object to be created (line 18). If flag is 0, PacMan is created. If flag is 1, PacManWithEye is created.
- ✓ Variable "button1" for JButton, and variable "text" for JTextField.

```
*Painter.java
1  /*
2   * An example of polymorphism
3   * by Hitoshi Ogawa on November 15th, 2021
4   */
5  import java.awt.BorderLayout;
6  import java.awt.Color;
7  import java.awt.Graphics;
8  import java.awt.event.MouseAdapter;
9  import java.awt.event.MouseEvent;
10 import javax.swing.JButton;
11 import javax.swing.JFrame;
12 import javax.swing.JPanel;
13 import javax.swing.JTextField;
14
15 public class Painter extends JFrame{
16     Face face[] = new Face[10];
17     int count = -1; // the number of Pac-Man
18     int flag = 0; // 0: PacMan, 1: PacManWithEye
19
20     JButton button1;
21     JTextField text;
22
23     public Painter(){
24         setTitle("Painter");
25         setSize(500,500);
26         setDefaultCloseOperation(EXIT_ON_CLOSE);
27         MyListener myListener = new MyListener();
28         addMouseListener(myListener);
29         addMouseMotionListener(myListener);
30
31         button1 = new JButton();
32         button1.addMouseListener(new ButtonListener1());
33         button1.setForeground(Color.red);
34         text = new JTextField();
35         text.setColumns(16);
36         JPanel p = new JPanel();
37         p.add(button1);
38         p.add(text);
39         getContentPane().add(p, BorderLayout.SOUTH);
40
41         button1.setText("PacMan");
42         text.setText("Click the center point");
43         setVisible(true);
44     }
45
46     public void paint(Graphics g){
47         super.paint(g);
48         g.clearRect(0, 0, 500, 450);
49         if(count >= 0){
50             if(count > 9){
51                 count = 9;
52             }
53             for(int i = 0; i < count + 1; i++){
54                 face[i].make(g);
55             }
56         }
57     }
}
```

Figure 5 Painter.java.

```

58
59 public static void main(String[] args) {
60     new Painter();
61 }
62
63 class MyListener extends MouseAdapter {
64     int distance, xCenter, yCenter;;
65
66     public void mousePressed(MouseEvent e) {
67         if(++count < 10){
68             if(flag == 0){
69                 face[count] = new PacMan(0, e.getX(), e.getY(), 60);
70             } else if(flag == 1){
71                 face[count] = new PacManWithEye(0, e.getX(), e.getY(), 60);
72             }
73             repaint();
74             text.setText("The center is (" + e.getX() + ", " + e.getY() + ")");
75         } else {
76             text.setText("No more PacMan.");
77         }
78     }
79
80     public void mouseDragged(MouseEvent e) {
81         if(count < 10){
82             distance = (int)Math.hypot(e.getX() - face[count].getXCenter(),
83                                     e.getY() - face[count].getYCenter());
84             face[count].setSize(distance * 2);
85             text.setText(Integer.toString(distance * 2));
86             repaint();
87         }
88     }
89 }
90
91 class ButtonListener1 extends MouseAdapter{
92     public void mousePressed(MouseEvent e){
93         if(++flag > 1){
94             flag = 0;
95         }
96         if(flag == 0){
97             button1.setText("PacMan");
98         } else if(flag == 1){
99             button1.setText("PacManWithEye");
100         }
101     }
102 }
103 }

```

Figure 5 Painter.java (continue).

A window of 500 pixels wide by 500 pixels vertically is created in Printer class (line 25). It is necessary to correct Face class according to the size of this window. In the x direction, 0 to 500 can be used. On the other hand, since here is a title bar and the area for a button and text area in the y direction, it can only be used from 25 to 450.

Since mousePressed method and mouseDragged method are set in the inner class MyListener (lines 63 to 89), the event processing is registered using addMouseListener and addMouseMotionListener (lines 28 and 29). When the mouse button is pressed, the variable count is incremented by 1 in mousePressed method (line 67). If count is less than 10, the object of PacMan or PacManWithEye is substituted for face[count] according to the value of the variable flag. The position of the center is a pressed place. In the mouseDragged method, the distance between the position of the drag and the position of the center of Pacman is calculated, and double of the distance is registered as the size of Pacman.

Processing when the button is pressed is set in the inner class ButtonListener1 (lines 91 to 102). The initial value of the label of "button1" is "PacMan" (line 41). Every time it is pressed, the value of flag is changed (lines 93 to 95). When the value of flag is 0, "PacMan" is displayed on the button1 (line 97). When it is 1, "PacManWithEye" is displayed on the button1 (line 99).

The statement "super.paint(g)" (line 47), which is executed first in the paint method, is very important for running Swing in the AWT environment. Basically, the window is drawn by the system provide by AWT. Components provided by Swing, such as JFrame, JButton and JTextField, need to work on these environments. With "super.paint(g)", Swing is executed successfully using higher layer functions. The rectangle of 500 pixels wide by 450 pixels horizontally is repainted from the upper left coordinate (0, 0) to erase all the drawn figures. Then, Pacmans stored in the elements from 0 to count of face array are drawn.

An example of program execution is shown (a) to (e) in Figure 6. Figure 6 (a) shows the initial state. When the mouse button is pressed, the center coordinates are displayed in the text area (Figure 6 (b)). When dragging, the size of Pacman, which is twice the distance between the center coordinates and the mouse cursor, is displayed in the text area (Figure 6 (c)). When the mouse button is released, the size of Pacman is decided. If the button is pressed, the type of Pacman to draw will be changed (Figure 6 (d)). If we try to draw the eleventh Pacman, we will see the message "No more PacMan" in the text area (Figure 6 (e)).

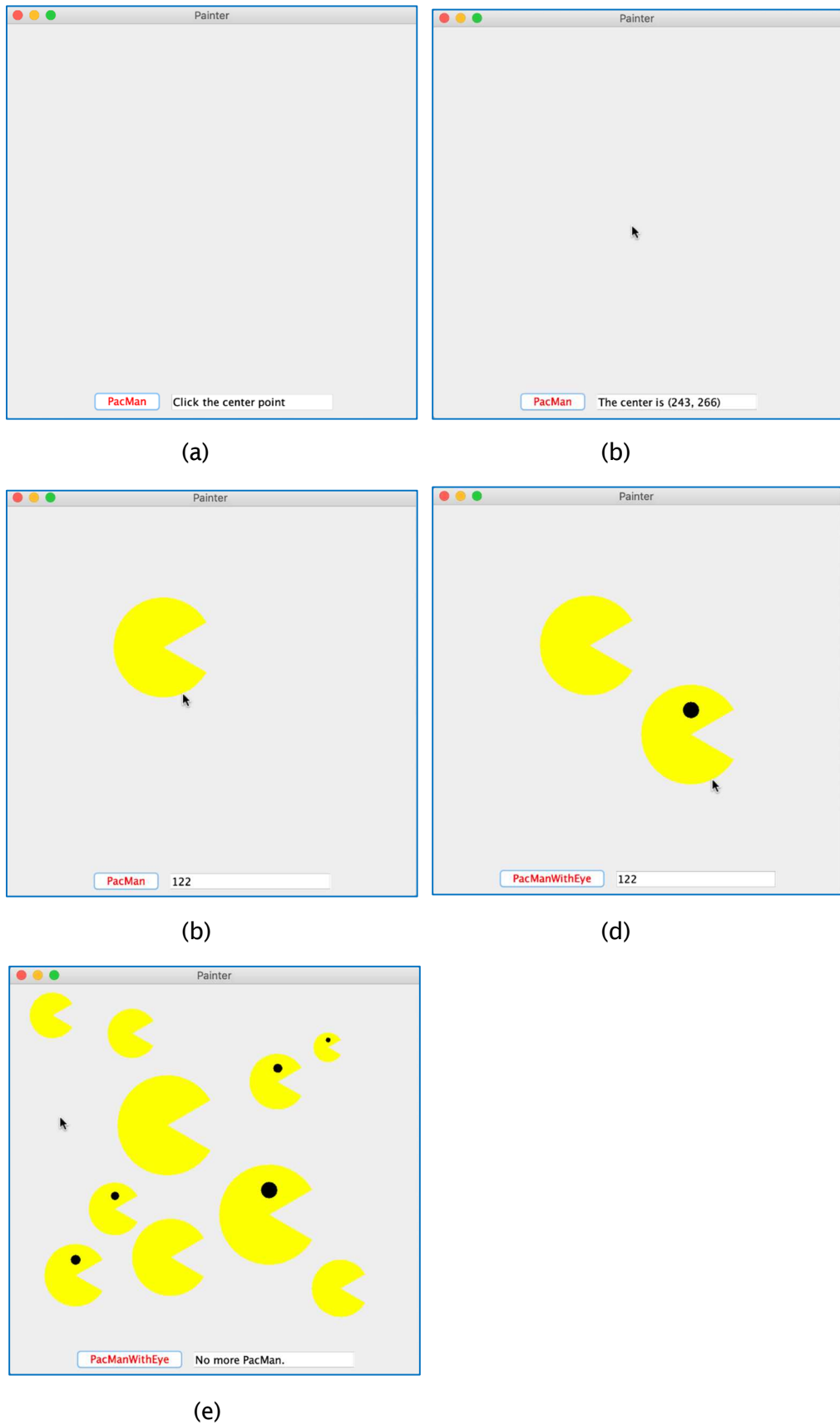


Figure 6. (a) The initial state, (b) When the mouse button is pressed, (c) Dragging to determine the size of Pacman, (d) Drawing PacManWithEye, (e) Status when trying to draw more than 10



#### 4. Development System

In Painter class, the multiple types of Pacman are treated with one array “face”. This makes it possible to handle various Pacman in the same way. In the paint method, drawing is made by the make method, regardless of the type of Pacman.

Suppose “ChangePacMan” class with three modes (create mode, check mode and change mode) by extending “Painter” class. The Painter class shown in Figure 5 realized the create mode. The outline of “ChangePacMan” is shown in Figure 7.

```
*ChangePacMan.java
1  /*
2   * ChangePacMan has three modes.
3   * mode = 0: create,
4   *       1: check the type of PacMan,
5   *       2: change the type of PacMan
6   * by Hitoshi Ogawa on November 15th, 2021
7   */
8  import java.awt.BorderLayout;
9  import java.awt.Color;
10 import java.awt.Graphics;
11 import java.awt.event.MouseAdapter;
12 import java.awt.event.MouseEvent;
13 import javax.swing.JButton;
14 import javax.swing.JFrame;
15 import javax.swing.JPanel;
16 import javax.swing.JTextField;
17
18 public class ChangePacMan extends JFrame{
19     Face face[] = new Face[10];
20     int count = -1; // the number of Pac-Man
21     int flag = 0; // 0: PacMan, 1: PacManWithEye
22     int mode = 0; // 0: create, 1: check, 2: change
23
24     JButton button1, button2;
25     JTextField text;
26
27     public ChangePacMan(){
28         setTitle("Painter");
29         setSize(500,500);
30         setDefaultCloseOperation(EXIT_ON_CLOSE);
31         setBackground(Color.white);
32         MyListener myListener = new MyListener();
33         addMouseListener(myListener);
34         addMouseMotionListener(myListener);
35
36         button1 = new JButton();
37         button1.addMouseListener(new ButtonListener1());
38         button2 = new JButton();
39         button2.addMouseListener(new ButtonListener2());
40         text = new JTextField();
41         text.setColumns(16);
42         JPanel p = new JPanel();
43         p.add(button2);
44         p.add(button1);
45         p.add(text);
46         getContentPane().add(p, BorderLayout.SOUTH);
47         button2.setText("create mode");
48         button1.setText("PacMan");
49         text.setText("Click the center point");
50         setVisible(true);
51     }
52
53     public void paint(Graphics g){
54         super.paint(g);
55         g.clearRect(0, 0, 500, 450);
56         if(count >= 0){
57             if(count > 9){
58                 count = 9;
59             }
60             for(int i = 0; i < count + 1; i++){
61                 face[i].make(g);
62             }
63         }
64     }
65
66     public static void main(String[] args) {
67         new ChangePacMan();
68     }
69 }
```

Figure 7. The source code of ChangePacMan class.

```

69
70
71 class MyListener extends MouseAdapter {
72     int distance, xCenter, yCenter;
73
74     public void mousePressed(MouseEvent e) {
75         if(mode == 0) {
76             if(++count < 10){
77                 if(flag == 0){
78                     face[count] = new PacMan(0, e.getX(), e.getY(), 60);
79                 } else if(flag == 1){
80                     face[count] = new PacManWithEye(0, e.getX(), e.getY(), 60);
81                 }
82                 repaint();
83                 text.setText("The center is (" + e.getX() + ", " + e.getY() + ")");
84             } else {
85                 text.setText("No more PacMan.");
86             }
87         } else if(mode == 1) {
88             Boolean found = false;
89             if(count > 9) count = 9;
90             for(int num = 0; num < count + 1; num++) {
91                 if(Math.hypot(e.getX() - face[num].getXCenter(),
92                     e.getY() - face[num].getYCenter()) < face[num].getSize() / 2) {
93                     found = true;
94                     if(face[num] instanceof PacMan) {
95                         text.setText("This is PacMan");
96                     } else if(face[num] instanceof PacManWithEye) {
97                         text.setText("This is PacManWithEye");
98                     } else {
99                         text.setText("Type is not found");
100                     }
101                 }
102             }
103             if(!found) {
104                 text.setText("Press a Pac-Man");
105             }
106         } else if(mode == 2) {
107
108             /* change mode */
109
110         }
111     }
112
113     public void mouseDragged(MouseEvent e) {
114         if(count < 10 && mode == 0){
115             distance = (int)Math.hypot(e.getX() - face[count].getXCenter(),
116                 e.getY() - face[count].getYCenter());
117             face[count].setSize(distance * 2);
118             text.setText(Integer.toString(distance * 2));
119             repaint();
120         }
121     }
122 }
123
124 class ButtonListener1 extends MouseAdapter{
125     public void mousePressed(MouseEvent e){
126         if(++flag > 1){
127             flag = 0;
128         }
129         if(flag == 0){
130             button1.setText("PacMan");
131         } else if(flag == 1){
132             button1.setText("PacManWithEye");
133         }
134     }
135 }
136
137 class ButtonListener2 extends MouseAdapter{
138     public void mousePressed(MouseEvent e){
139         if(++mode > 2){
140             mode = 0;
141         }
142         if(mode == 0){
143             button2.setText("create mode");
144         } else if(mode == 1){
145             button2.setText("check mode");
146         } else if(mode == 2) {
147             button2.setText("change mode");
148         }
149     }
150 }
151 }

```

Figure 7. The source code of ChangePacMan class (continue).

The following items need to be added to the program to handle “mode”.

- (a) Variable “mode” indicating mode (line 22).
- (b) Button variable “button2” for changing the mode (line 24). The “button2” is add to JPanel p (line 44) before “button1”. Therefore, “button2” is placed to the left of “button1”. The initial label of “button2” is “create mode”.
- (c) The inner class “ButtonListener2” for changing mode (line 167 to 180). This class changes the value of the variable “mode” to 0, 1, 2 each time it is called. Its instance is registered with “button2” using addMouseListener.

The execution result is shown in Figure 8. The button for changing the mode is added to the left.

When the mode is “create mode”, Pacman can be drawn in the same way as Painter class (Figure 8 (a)).

When the mode is “check mode”, the type of Pacman pressed is displayed in the text field (Figure 8 (b) (c)). If Pacman is not pressed, the message “Press a Pac-Man” is displayed (Figure 8 (d)).

When the mode is “change mode”, the pressed Pacman is changed to the type of Pacman displayed at “button1”. The message is displayed in the text field (Figure 8 (e) (f)).

Refer to the demonstration of Home Work 2 for the execution example.

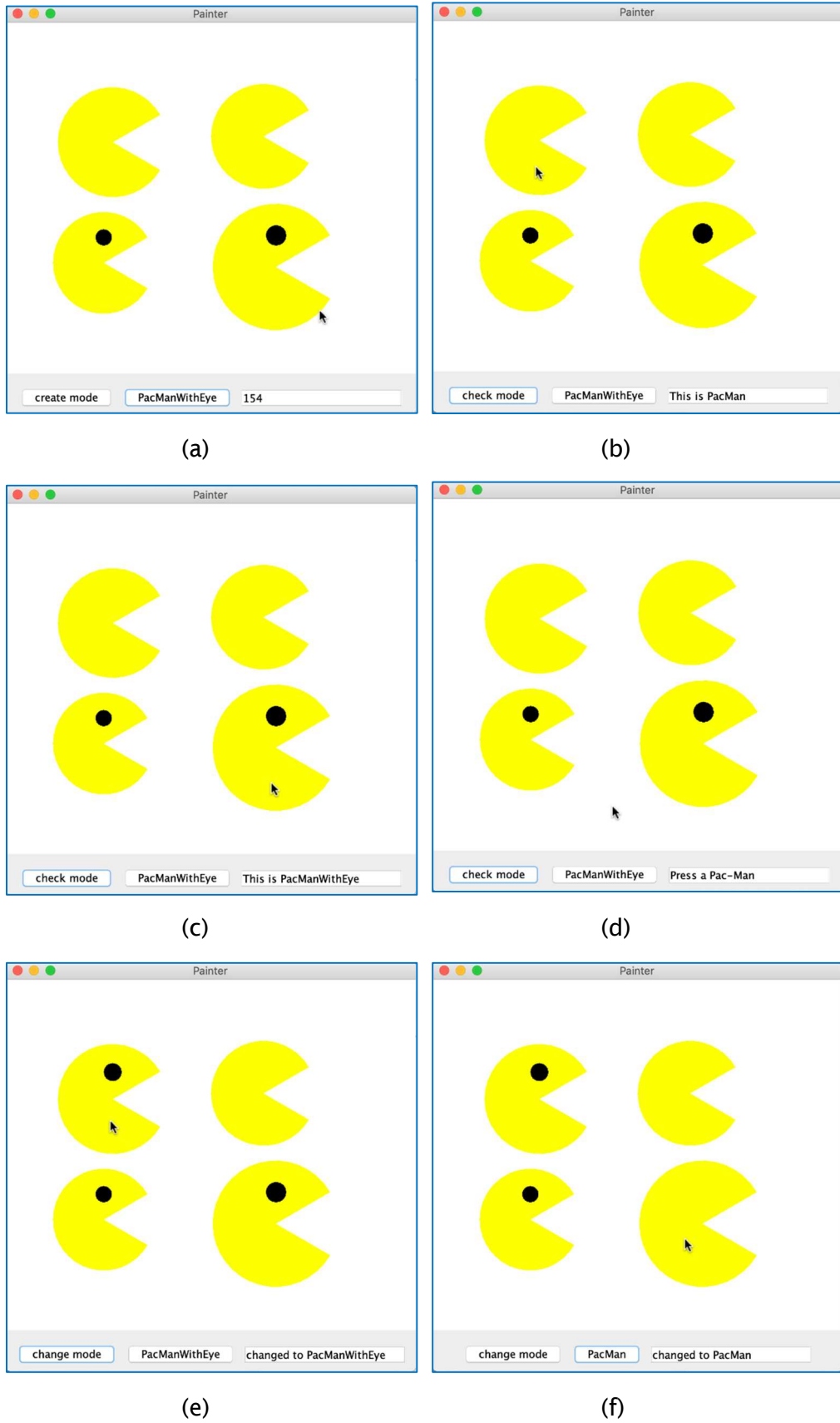


Figure 8. (a) An example of create mode.  
 (b) (c)(d) Examples of check mode.  
 (e)(f) Examples of change mode.

The Program code corresponding to the check mode is on lines 86 to 104.

When the mouse is pressed, if the distance between the pressed place and the center of Pacman is smaller than the radius of the Pacman, it can be determined that it is a pressed Pacman (lines 90 and 91). If no Pacman is applicable (the value of the variable found was not changed to true), it can be determined that a place other than Pacman is being pressed (lines 87, 102 to 104).

To test whether the object is an instance of the specified type (class or subclass), the “instanceof” operator is used. The “instanceof” operator is written as

|   |
|---|
| <i>object_reference_variable</i> <b>instanceof</b> <i>class</i> |
|---|

If the content of variable *object\_reference\_variable* is an object of *class*, then the result will be true. It is checked whether face[num] is an instance of PacMan (line 93) or PacManWithEye (line 95).