

Java Development Kit

Урок 2

Программные интерфейсы







Программные интерфейсы





На предыдущем уроке

Поговорили о графических интерфейсах пользователя:

-  создали немного окон,
-  разместили немного компонентов,
-  порисовали элементы,
-  обработали несколько событий.

Поняли, что для продолжения нужно изучить программные интерфейсы.





Что будет на уроке сегодня

- 📌 Программные интерфейсы — понятие и принцип работы;
- 📌 Ключевое слово `implements`;
- 📌 Наследование и множественное наследование интерфейсов;
- 📌 Реализация, реализации по-умолчанию;
- 📌 Частичная реализация интерфейсов, адаптеры;
- 📌 Анонимные классы;
- 📌 Исключения в графических фреймворках.



Disclaimer

Обычно — теория и примеры

Сегодня — от практики и плохого кода к хорошему коду и теоретическому обоснованию





Приложение
для примера





Что будем писать?





ОСНОВНОЕ ОКНО

```
5 ▶ public class MainWindow extends JFrame {  
6     private static final int POS_X = 400;  
7     private static final int POS_Y = 200;  
8     private static final int WINDOW_WIDTH = 800;  
9     private static final int WINDOW_HEIGHT = 600;  
10  
11     private MainWindow() {  
12         setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);  
13         setBounds(POS_X, POS_Y, WINDOW_WIDTH, WINDOW_HEIGHT);  
14         setTitle("Circles");  
15  
16         setVisible(true);  
17     }  
18  
19 ▶ public static void main(String[] args) {  
20     new MainWindow();  
21 }  
22 }
```




Канва для рисования

```
6 public class MainCanvas extends JPanel {
7     MainCanvas() {
8         setBackground(Color.BLUE);
9     }
10
11     @Override
12     protected void paintComponent(Graphics g) {
13         super.paintComponent(g);
14     }
15
16     public int getLeft() { return 0; }
19     public int getRight() { return getWidth() - 1; }
22     public int getTop() { return 0; }
25     public int getBottom() { return getHeight() - 1; }
28 }
29
```

Отделяем логику от рисования

```
10
11 private MainWindow() {
12     setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
13     setBounds(POS_X, POS_Y, WINDOW_WIDTH, WINDOW_HEIGHT);
14     setTitle("Circles");
15
16     MainCanvas canvas = new MainCanvas();
17     add(canvas);
18     setVisible(true);
19 }
20
21 public void onDrawFrame() {
22     update();
23     render();
24 }
25
26 private void update() {}
29 private void render() {}
32
```





Учим канву периодически сообщать о себе

```
6 public class MainCanvas extends JPanel {
7     private final MainWindow controller;
8
9     MainCanvas(MainWindow controller) {
10         setBackground(Color.BLUE);
11         this.controller = controller;
12     }
13
14     @Override
15     protected void paintComponent(Graphics g) {
16         super.paintComponent(g);
17         controller.onDrawFrame();
18     }
```



Учим канву периодически сообщать о себе

```
6 public class MainCanvas extends JPanel {
7     private final MainWindow controller;
8
9     MainCanvas(MainWindow controller) {
10         setBackground(Color.BLUE);
11         this.controller = controller;
12     }
13
14     @Override
15     protected void paintComponent(Graphics g) {
16         super.paintComponent(g);
17         controller.onDrawFrame();
18         repaint();
19     }
```

Учим канву периодически сообщать о себе

```
7      private final MainWindow controller;  
8  
9      MainCanvas(MainWindow controller) {  
10         setBackground(Color.BLUE);  
11         this.controller = controller;  
12     }  
13  
14     @Override  
15     protected void paintComponent(Graphics g) {  
16         super.paintComponent(g);  
17         controller.onDrawFrame();  
18         try {  
19             Thread.sleep(16);  
20         } catch (InterruptedException e) {  
21             throw new RuntimeException(e);  
22         }  
23         repaint();  
24     }
```





Учим канву периодически сообщать о себе

```
7      private final MainWindow controller;  
8  
9      MainCanvas(MainWindow controller) {  
10         setBackground(Color.BLUE);  
11         this.controller = controller;  
12     }  
13  
14     @Override  
15     protected void paintComponent(Graphics g) { // do {  
16         super.paintComponent(g); // something  
17         controller.onDrawFrame(); // useful  
18         try { // sleep  
19             Thread.sleep(16); // 16 ms  
20         } catch (InterruptedException e) {  
21             throw new RuntimeException(e);  
22         }  
23         repaint(); // } while (true);  
24     }
```



Отдаём канве «себя»

```
MainWindow.java x MainCanvas.java x
11 private MainWindow() {
12     setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
13     setBounds(POS_X, POS_Y, WINDOW_WIDTH, WINDOW_HEIGHT);
14     setTitle("Circles");
15
16     MainCanvas canvas = new MainCanvas(this);
17     add(canvas);
18     setVisible(true);
19 }
20
```



Отдаём основному окну «себя» и «точное время»

```
21
22     public void onDrawFrame(MainCanvas canvas, Graphics g, float deltaTime) {
23         update(canvas, deltaTime);
24         render(canvas, g);
25     }
26
27     private void update(MainCanvas canvas, float deltaTime) {}
30     private void render(MainCanvas canvas, Graphics g) {}
33
```


Отдаём основному окну «себя» и «точное время»

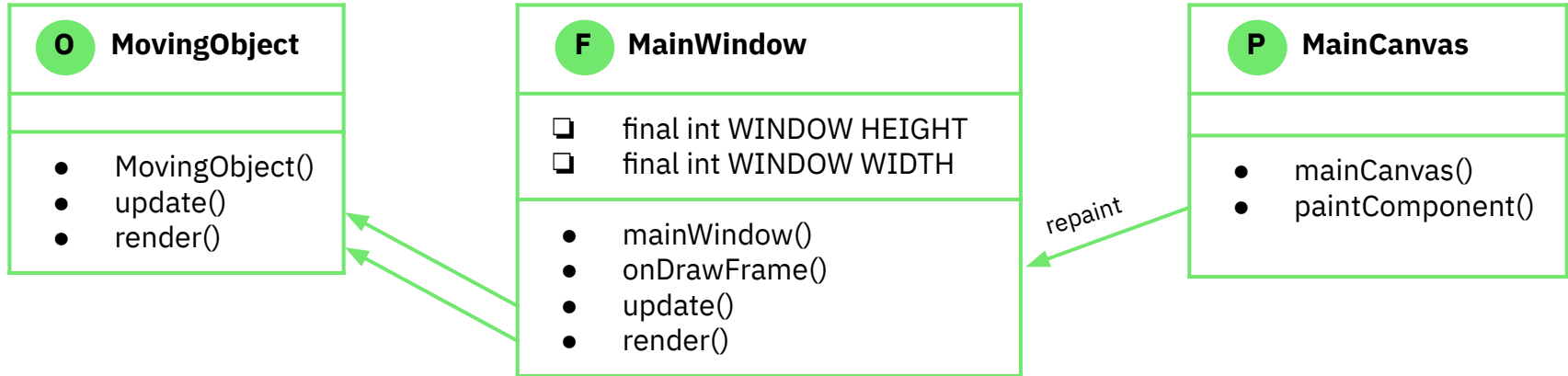
```
MainWindow.java  MainCanvas.java  Sprite.java  Ball.java

7      private final MainWindow controller;
8      private long lastFrameTime;
9
10     MainCanvas(MainWindow controller) {
11         this.controller = controller;
12         lastFrameTime = System.nanoTime();
13     }
14
15     @Override
16     protected void paintComponent(Graphics g) {
17         super.paintComponent(g);
18         try {...} catch (InterruptedException e) {
19             throw new RuntimeException(e);
20         }
21     }
22
23     float deltaTime = (System.nanoTime() - lastFrameTime) * 0.000000001f;
24     controller.onDrawFrame(this, g, deltaTime);
25     lastFrameTime = System.nanoTime();
26     repaint();
27 }
```





Общий маршрут обновления объектов





Абстрактный двумерный объект

```
5 public abstract class Sprite {
6     protected float x;
7     protected float y;
8     protected float halfWidth;
9     protected float halfHeight;
10
11     protected float getLeft() { return x - halfWidth; }
14     protected void setLeft(float left) { x = left + halfWidth; }
17     protected float getRight() { return x + halfWidth; }
20     protected void setRight(float right) { x = right - halfWidth; }
23     protected float getTop() { return y - halfHeight; }
26     protected void setTop(float top) { y = top + halfHeight; }
29     protected float getBottom() { return y + halfHeight; }
32     protected void setBottom(float bottom) { y = bottom - halfHeight; }
35
36     protected float getWidth() { return 2f * halfWidth; }
39     protected float getHeight() { return 2f * halfHeight; }
42
43     void update(MainCanvas canvas, float deltaTime) {}
44     void render(MainCanvas canvas, Graphics g) {}
45 }
```



Конкретный двумерный объект

```
MainWindow.java x MainCanvas.java x Sprite.java x Ball.java x
6      public class Ball extends Sprite {
7          private static Random rnd = new Random();
8          private final Color color;
9          private float vX;
10         private float vY;
11
12         Ball() {
13             halfHeight = 20 + (float) (Math.random() * 50f);
14             halfWidth = halfHeight;
15             color = new Color(rnd.nextInt());
16             vX = 100f + (float) (Math.random() * 200f);
17             vY = 100f + (float) (Math.random() * 200f);
18         }
```



Конкретный двумерный объект

```
42      @Override
43      @Override
44      void render(MainCanvas canvas, Graphics g) {
45          g.setColor(color);
46          g.fillOval((int) getLeft(), (int) getTop(),
47                    (int) getWidth(), (int) getHeight());
48      }
```





Конкретный двумерный объект

```
20      @Override
21      void update(MainCanvas canvas, float deltaTime) {
22          x += vX * deltaTime;
23          y += vY * deltaTime;
24
25          if (getLeft() < canvas.getLeft()) {
26              setLeft(canvas.getLeft());
27              vX = -vX;
28          }
29          if (getRight() > canvas.getRight()) {...}
33          if (getTop() < canvas.getTop()) {...}
37          if (getBottom() > canvas.getBottom()) {...}
41      }
42
```

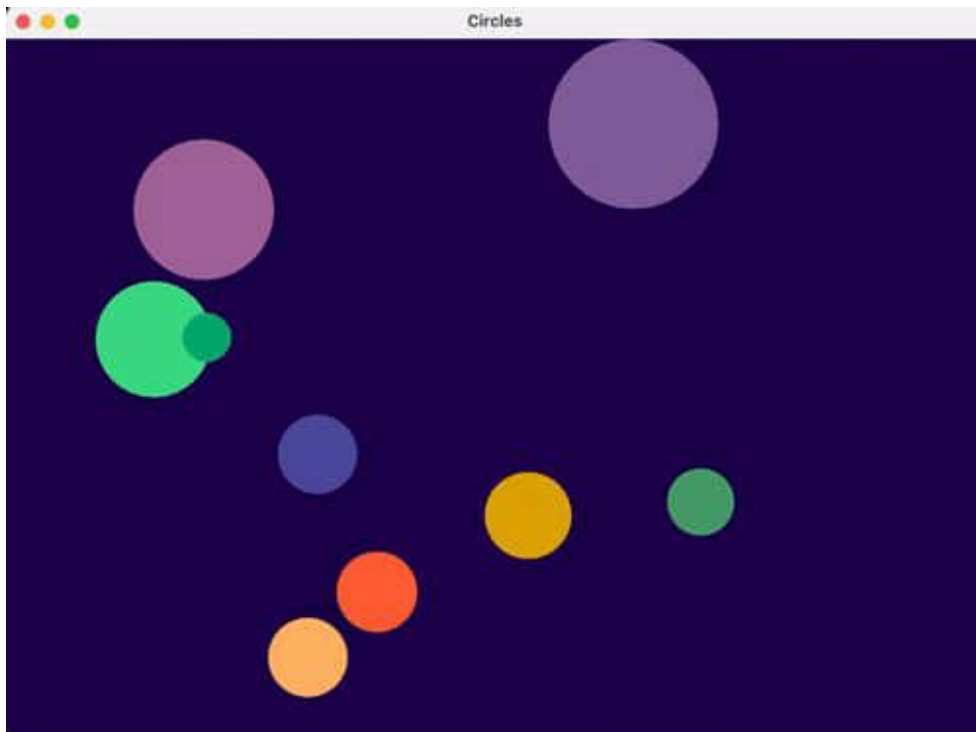


Новые шарiki

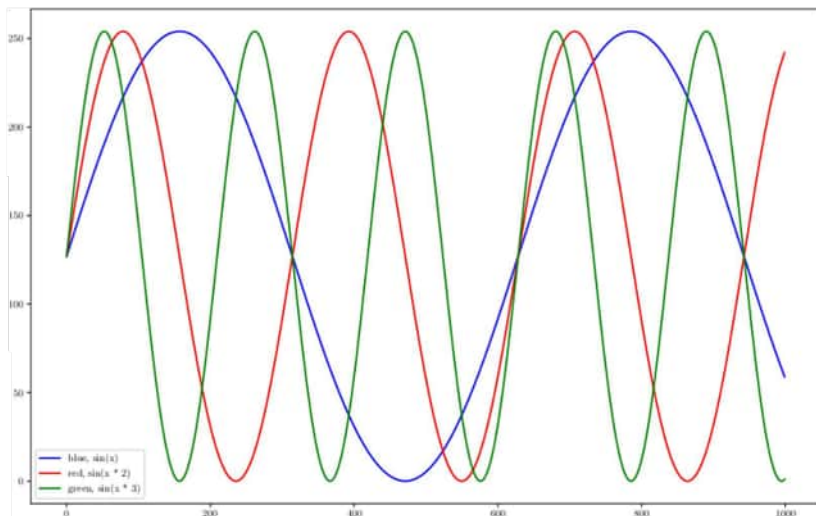
```
MainWindow.java  MainCanvas.java  Sprite.java  Ball.java
12     private final Sprite[] sprites = new Sprite[10];
13
14     private MainWindow() {
15         setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
16         setBounds(POS_X, POS_Y, WINDOW_WIDTH, WINDOW_HEIGHT);
17         setTitle("Circles");
18         for (int i = 0; i < sprites.length; i++) {
19             sprites[i] = new Ball();
20         }
21         MainCanvas canvas = new MainCanvas(this);
22         add(canvas);
23         setVisible(true);
24     }
25
26     public void onDrawFrame(MainCanvas canvas, Graphics g, float deltaTime) {...}
27
28
29
30
31     private void update(MainCanvas canvas, float deltaTime) {
32         for (int i = 0; i < sprites.length; i++) {
33             sprites[i].update(canvas, deltaTime);
34         }
35     }
36
37     private void render(MainCanvas canvas, Graphics g) {
```



Результаты подготовки



Изменение цвета фона



```
1 public class Background extends Sprite {
2     private float time;
3     private static final float AMPLITUDE = 255f / 2f;
4     private Color color;
5
6     @Override
7     public void update(MainCanvas canvas, float deltaTime) {
8         time += deltaTime;
9         int red = Math.round(AMPLITUDE + AMPLITUDE * (float) Math.sin(time * 2f));
10        int green = Math.round(AMPLITUDE + AMPLITUDE * (float) Math.sin(time * 3f));
11        int blue = Math.round(AMPLITUDE + AMPLITUDE * (float) Math.sin(time));
12        color = new Color(red, green, blue);
13    }
14
15    @Override
16    public void render(MainCanvas canvas, Graphics g) {
17        canvas.setBackground(color);
18    }
19 }
```



Понятие интерфейса



Понятие интерфейса

Интерфейс — это описание способов взаимодействия с объектом



Интерфейсы определяют функционал, не имеющий конкретной реализации.



Синтаксис интерфейса

```
Human.java
1 package ru.gb.jdk.two.online.samples;
2
3 public interface Human {
4     public void walk();
5     public void talk();
6 }
7

Bull.java
1 package ru.gb.jdk.two.online.samples;
2
3 public interface Bull {
4     void walk();
5     void talk();
6 }
7
```



Реализация интерфейсов

```
Man.java
3 public class Man implements Human {
4     @Override
5     public void walk() {
6         System.out.println("Walks on two feet");
7     }
8
9     @Override
10    public void talk() {
11        System.out.println("Talks meaningful words");
12    }
13 }

Ox.java
3 public class Ox implements Bull {
4     @Override
5     public void walk() {
6         System.out.println("Walks on hooves");
7     }
8
9     @Override
10    public void talk() {
11        System.out.println("Moo0oo0ooo00oo");
12    }
13 }

Human.java
1 package ru.gb.jdk.two.online.samples;
2
3 public interface Human {
4     public void walk();
5     public void talk();
6 }

Bull.java
1 package ru.gb.jdk.two.online.samples;
2
3 public interface Bull {
4     void walk();
5     void talk();
6 }
```



Реализация интерфейсов

```
Human.java x Main.java x
3  ▶ public class Main {
4  ▶  public static void main(String[] args) {
5      Man man0 = new Man();
6      Ox ox0 = new Ox();
7      Human man1 = new Man();
8      Bull ox2 = new Ox();
9      }
10 }
```

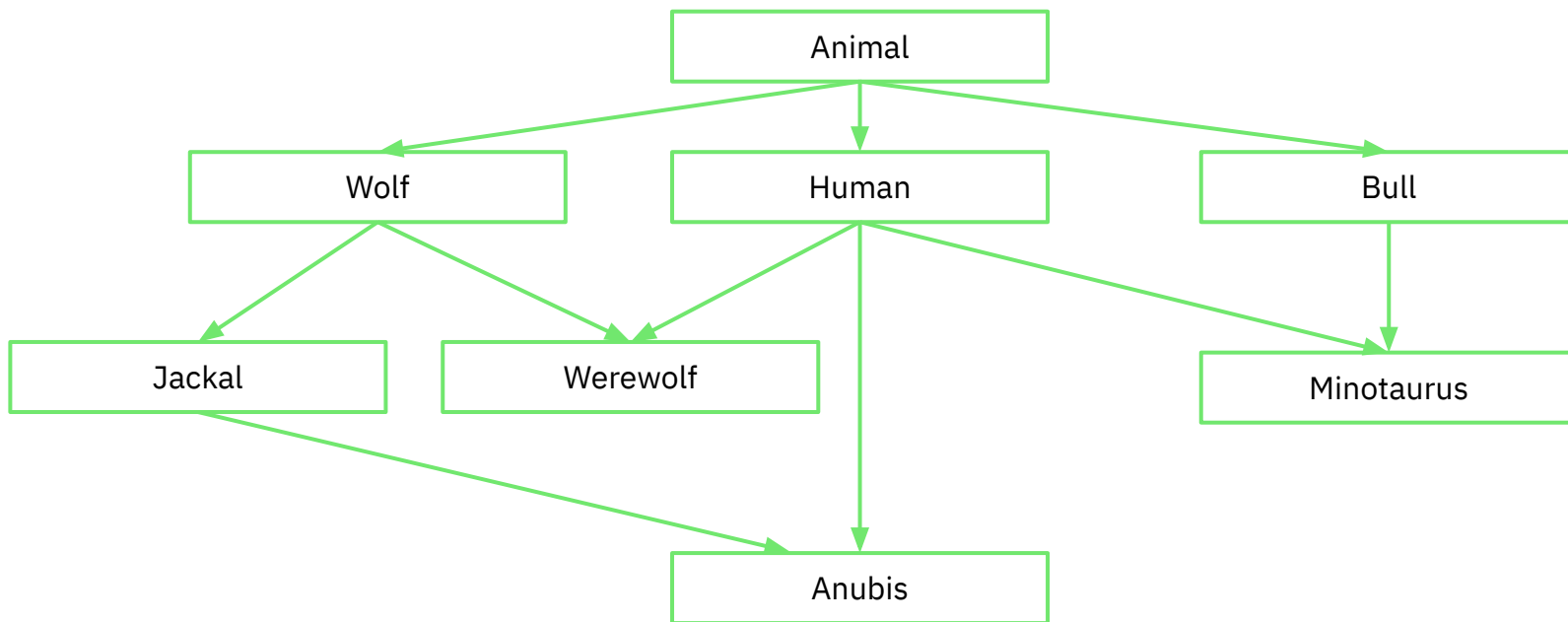
Реализация нескольких интерфейсов

```
Human.java Main.java Man.java Ox.java
3 public class Main {
4     private static class Minotaurus implements Human, Bull {
5         @Override public void walk() {
6             System.out.println("Walks on two legs");
7         }
8
9         @Override public void talk() {
10             System.out.println("Asks you a riddle");
11         }
12     }
13     public static void main(String[] args) {
14         Bull minos0 = new Minotaurus();
15         Human minos1 = new Minotaurus();
16         Minotaurus minos = new Minotaurus();
17         Human man1 = new Man();
18         Bull ox2 = new Ox();
19         Bull[] allBulls = {ox2, minos0, minos};
20         Human[] allHumans = {man1, minos, minos1};
21     }
```





Наследование интерфейсов



Ответьте на вопросы сообщением в чат

Вопросы:

- **Программный интерфейс — это:**
 - a. окно приложения в ОС;
 - b. реализация методов объекта;
 - c. объявление методов, реализуемых в классах.
- **Интерфейсы нужны для:**
 - a. компенсации отсутствия множественного наследования;
 - b. отделения API и реализации;
 - c. оба варианта верны.
- **Интерфейсы позволяют:**
 - a. удобно создавать новые объекты, не связанные наследованием;
 - b. единообразно обращаться к методам объектов, не связанных наследованием;
 - c. полностью заменить наследование.





Фон как спрайт

```
MainWindow.java Background.java MainCanvas.java Sprite.java Ball.java
4
5 public class Background extends Sprite {
6     private float time;
7     private static final float AMPLITUDE = 255f / 2f;
8     private Color color;
9
10    @Override
11    public void update(MainCanvas canvas, float deltaTime) {
12        time += deltaTime;
13        int red = Math.round(AMPLITUDE + AMPLITUDE * (float) Math.sin(time * 2f));
14        int green = Math.round(AMPLITUDE + AMPLITUDE * (float) Math.sin(time * 3f));
15        int blue = Math.round(AMPLITUDE + AMPLITUDE * (float) Math.sin(time));
16        color = new Color(red, green, blue);
17    }
18
19    @Override
20    public void render(MainCanvas canvas, Graphics g) {
21        canvas.setBackground(color);
22    }
23 }
```



Внедрение нового интерфейса

```
5 public abstract class Sprite implements Interactable {
29     protected float getBottom() {
32     protected void setBottom(float bottom) {
35
36     protected float getWidth() {
39     protected float getHeight() {
42
43     void update(MainCanvas canvas, float deltaTime) {
44     void render(MainCanvas canvas, Graphics g) {

1 package ru.gb.jdk.two.online;
2
3 import java.awt.*;
4
5 public interface Interactable {
6     void update(MainCanvas canvas, float deltaTime);
7     void render(MainCanvas canvas, Graphics g);
8 }
9
```

Внедрение нового интерфейса

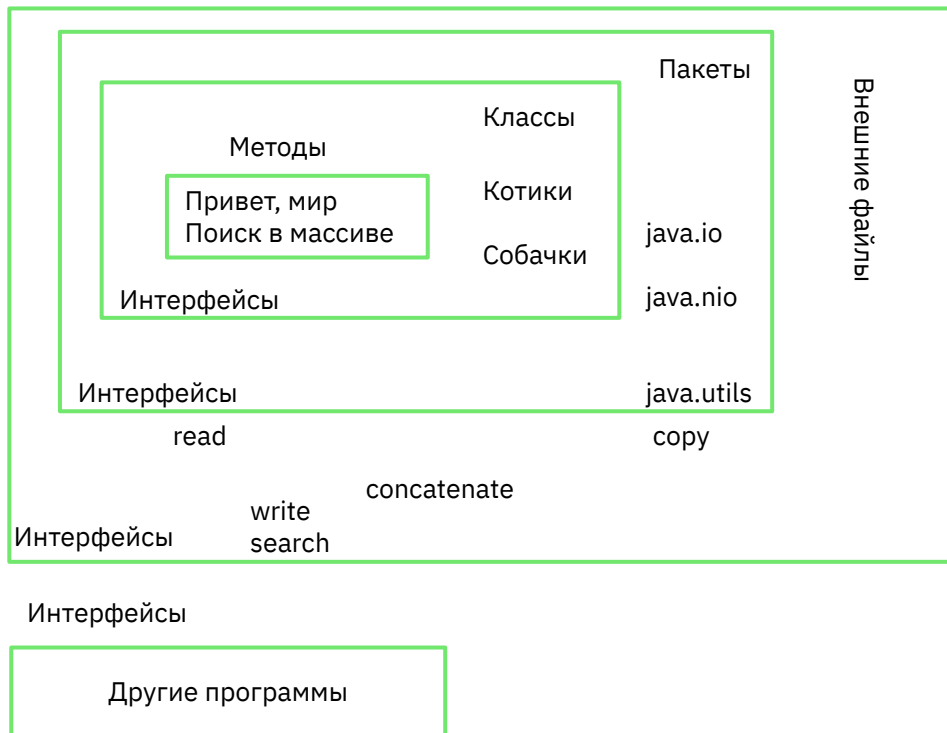
```
Sprite.java
1
2
3
4
5 public abstract class Sprite implements Interactable {
6     protected float x;
7     protected float y;
8     protected float halfWidth;
9     protected float halfHeight;
10
11     protected float getLeft() { return x - halfWidth; }
12     protected void setLeft(float left) { x = left + halfWidth; }
13     protected float getRight() { return x + halfWidth; }
14     protected void setRight(float right) { x = right - halfWidth; }
15     protected float getTop() { return y - halfHeight; }
16     protected void setTop(float top) { y = top + halfHeight; }
17     protected float getBottom() { return y + halfHeight; }
18     protected void setBottom(float bottom) { y = bottom - halfHeight; }
19
20     protected float getWidth() { return 2f * halfWidth; }
21     protected float getHeight() { return 2f * halfHeight; }
22
23     @Override
24     public void update(MainCanvas canvas, float deltaTime) {
25
26     }
27
28     @Override
29     public void render(MainCanvas canvas, Graphics g) {
30
31     }
32 }
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48

Background.java
1
2
3
4
5 public class Background implements Interactable {
6     private float time;
7     private static final float AMPLITUDE = 255f / 2;
8     private Color color;
9
10     @Override
11     public void update(MainCanvas canvas, float deltaTime) {
12         time += deltaTime;
13         int red = Math.round(AMPLITUDE * Math.sin(time));
14         int green = Math.round(AMPLITUDE * Math.cos(time));
15         int blue = Math.round(AMPLITUDE * Math.sin(time));
16         color = new Color(red, green, blue);
17     }
18
19     @Override
20     public void render(MainCanvas canvas, Graphics g) {
21         canvas.setBackground(color);
22     }
23 }
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
```





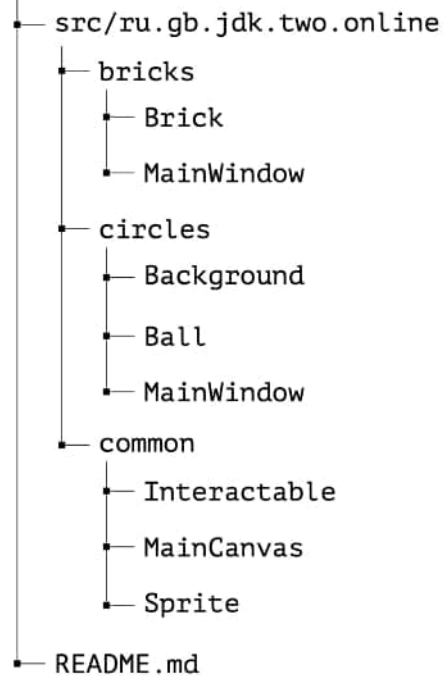
Применение интерфейсов в программах





Применение интерфейсов в программах

JDKit



Тиражирование приложений

```
17  
18 private MainWindow() {  
19     setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);  
20     setBounds(POS_X, POS_Y, WINDOW_WIDTH, WINDOW_HEIGHT);  
21     setTitle("Bricks");  
22     interactables[0] = new Background();  
23     for (int i = 1; i < interactables.length; i++) {  
24         interactables[i] = new Brick();  
25     }  
26     MainCanvas canvas = new MainCanvas(this);  
27     add(canvas);  
28     setVisible(true);  
29 }  
30  
31 public void onDrawFrame(MainCanvas canvas,  
32     update(canvas, deltaTime);  
33     render(canvas, g);  
34 }
```

Required type: ru.gb.jdk.two.online.circles.MainWindow
Provided: ru.gb.jdk.two.online.bricks.MainWindow
Create constructor: More actions...
ru.gb.jdk.two.online.common.MainCanvas
public MainCanvas(
 MainWindow controller
) {
 JDKIT



Тиражирование приложений

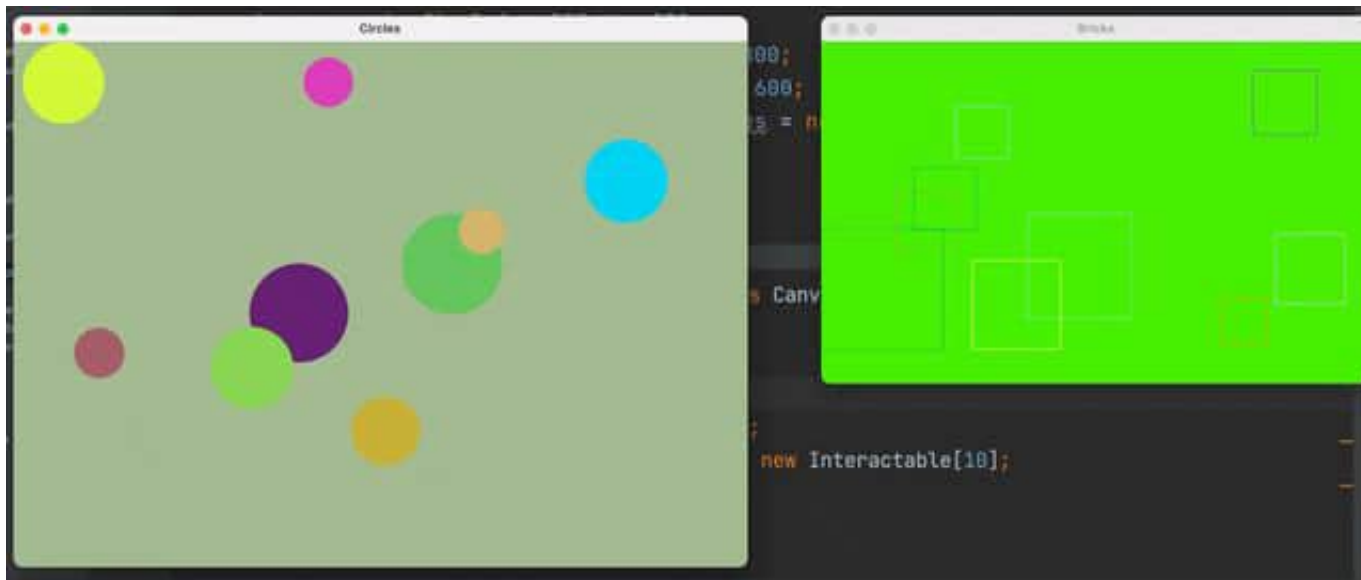
```
CanvasRepaintListener.java
1  package ru.gb.jdk.two.online.common;
2
3  import java.awt.*;
4
5  public interface CanvasRepaintListener {
6      void onDrawFrame(MainCanvas canvas, Graphics g, float deltaTime);
7  }
8
MainCanvas.java  bricks/MainWindow.java  circles/MainWindow.java
6  public class MainCanvas extends JPanel {
7      private final CanvasRepaintListener controller;
8      private long lastFrameTime;
9
10     public MainCanvas(CanvasRepaintListener controller) {
11         this.controller = controller;
12         lastFrameTime = System.nanoTime();
13     }
```


Тиражирование приложений

```
10 public class MainWindow extends JFrame implements CanvasRepaintListener {
11     private static final int POS_X = 200;
12     private static final int POS_Y = 200;
13     private static final int WINDOW_WIDTH = 800;
14     private static final int WINDOW_HEIGHT = 600;
15     private final Interactable[] interactables = new Interactable[10];
16
17     private MainWindow() {...}
18
19     @Override
20     public void onDrawFrame(MainCanvas canvas, Graphics g, float deltaTime) {...}
21 }
22
23 public class MainWindow extends JFrame implements CanvasRepaintListener {
24     private static final int POS_X = 1200;
25     private static final int POS_Y = 200;
26     private static final int WINDOW_WIDTH = 600;
27     private static final int WINDOW_HEIGHT = 400;
28     private final Interactable[] interactables = new Interactable[10];
29
30     private MainWindow() {...}
31
32     @Override
33     public void onDrawFrame(MainCanvas canvas, Graphics g, float deltaTime) {
```



Результат применения программных интерфейсов





Реализация по-умолчанию

```
1 package ru.gb.jdk.two.online.samples;
2
3 public interface Bull {
4     void walk() {
5         System.out.println("Walks on four hooves");
6     }
7     void talk();
8 }

```

```
1 package ru.gb.jdk.two.online.samples;
2
3 public interface Human {
4     default void walk() {
5         System.out.println("Walks on two feet");
6     }
7
8     public void talk();
9 }

```

Поля интерфейсов

```
1 package ru.gb.jdk.two.online.samples;
2
3 public interface Bull {
4     public static final int amount = 2;
5     default void walk() {
6         System.out.println("Walks on " + amount + " hooves");
7     }
8     void talk();
9 }
10
11 package ru.gb.jdk.two.online.samples;
12
13 public interface Human {
14     default void walk() {
15         System.out.println("Walks on two feet");
16     }
17
18     public void talk();
19 }
```





Анонимные классы



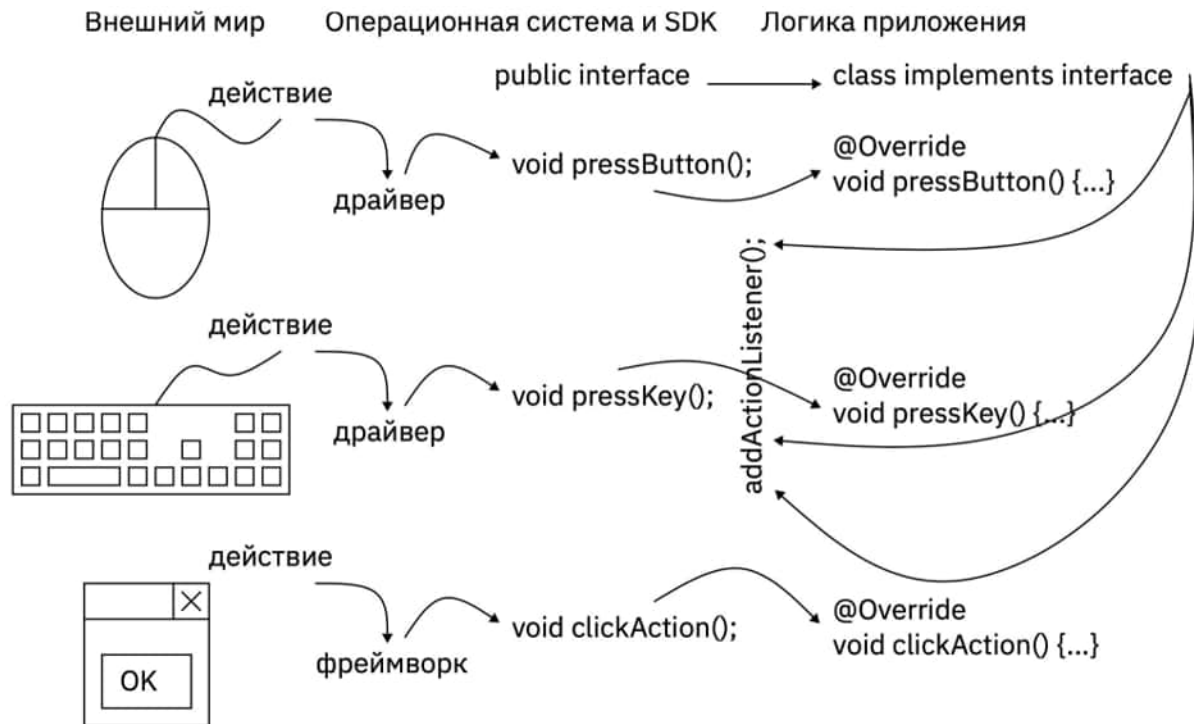
Реализация интерфейса классом

Анонимный класс — это класс без названия

```
3  ▶ public class Main {  
4      /*...*/  
14  ◀ public interface MouseListener {  
15  ◀     void mouseUp();  
16  ◀     void mouseDown();  
17  ◀ }  
18  
19  ◀ private static class MouseAdapter implements MouseListener {  
20  ◀     @Override public void mouseUp() {}  
23  ◀     @Override public void mouseDown() {}  
26  ◀ }  
27  
28  ▶ public static void main(String[] args) {  
29      MouseAdapter m = new MouseAdapter();  
30      m.mouseDown();  
31      m.mouseUp();  
}
```



Отвязка событий от обработчиков






«Передача интерфейса» в метод

```
Main.java
3  public class Main {
4      /*...*/
14  public interface MouseListener {...}
18  private static class MouseAdapter implements MouseListener {...}
26
27  @ private static void addMouseListener(MouseListener l) {
28      l.mouseDown();
29      l.mouseUp();
30  }
31
32  public static void main(String[] args) {
33      MouseAdapter m = new MouseAdapter();
34      addMouseListener(m);
```




«Передача интерфейса» в метод



```
3 public class Main {
4     /*...*/
14     public interface MouseListener {...}
18     private static class MouseAdapter implements MouseListener {...}
26
27     @
28     private static void addMouseListener(MouseListener l) {
29         l.mouseDown();
30         l.mouseUp();
31     }
32
33     public static void main(String[] args) {
34         MouseAdapter m = new MouseAdapter();
35         addMouseListener(m);
36         addMouseListener(new MouseAdapter());
37     }
38 }
```



«Передача интерфейса» в метод

```
3 public class Main {
4     /*...*/
14     public interface MouseListener {...}
18     private static class MouseAdapter implements MouseListener {...}
26
27     private static void addMouseListener(MouseListener l) {
28         l.mouseDown();
29         l.mouseUp();
30     }
31
32     public static void main(String[] args) {
33         MouseAdapter m = new MouseAdapter();
34         addMouseListener(m);
35         addMouseListener(new MouseAdapter());
36         MouseListener l = new MouseListener() {
37             @Override public void mouseUp() { }
38             @Override public void mouseDown() { }
39         };
40         addMouseListener(l);
    }
```



«Передача интерфейса» в метод

```
14 public interface MouseListener {...}
18 private static class MouseAdapter implements MouseListener {
19     @Override public void mouseUp() {}
22     @Override public void mouseDown() {}
25 }
26
27 @ private static void addMouseListener(MouseListener l) {...}
31
32 public static void main(String[] args) {
33     MouseAdapter m = new MouseAdapter();
34     addMouseListener(m);
35     addMouseListener(new MouseAdapter());
36     MouseListener l = new MouseListener() {...};
40     addMouseListener(l);
41     addMouseListener(new MouseListener() {
42         @Override public void mouseUp() { }
43         @Override public void mouseDown() { }
44     });
45 }
```

Применение адаптера

```
24 }
25 MainCanvas canvas = new MainCanvas(this);
26 canvas.addMouseListener(new Mouse);
27 add(canvas);
28 setVisible(true);
29 }

78 public abstract class MouseAdapter implements MouseListener, MouseWheelListener,
    MouseMotionListener {
79     /**
80      * {@inheritDoc}
81      */
82     public void mouseClicked(MouseEvent e) {}
83
84     /** {@inheritDoc} */
85     public void mousePressed(MouseEvent e) {}
86
87     /** {@inheritDoc} */
88     public void mouseReleased(MouseEvent e) {}
89
90     /**
91      * Invoked when the Mouse enters a component.
92      */
93     public void mouseEntered(MouseEvent e) {}
94
95     /**
96      * Invoked when the mouse exits a component.
97      */
98     public void mouseExited(MouseEvent e) {}
99 }
```

```
26 }
27 MainCanvas canvas = new MainCanvas(this);
28 canvas.addMouseListener(new MouseAdapter() {
29     @Override
30     public void mouseReleased(MouseEvent e) {
31         super.mouseReleased(e);
32     }
33 });
```

Реализация интерфейса «собой»

```
13 public class MainWindow extends JFrame implements  
14     CanvasRepaintListener, MouseListener {  
  
57     @Override  
58     public void mouseClicked(MouseEvent e) {}  
  
61  
62     @Override  
63     public void mousePressed(MouseEvent e) {}  
66  
67     @Override  
68     public void mouseReleased(MouseEvent e) {  
69         System.out.println("Clicked!");  
70     }  
71  
72     @Override  
73     public void mouseEntered(MouseEvent e) {}  
76  
77     @Override  
78     public void mouseExited(MouseEvent e) {}  
81 }  
82
```

```
29     MainCanvas canvas = new MainCanvas(this);  
30     canvas.addMouseListener(this);  
31     add(canvas);  
32     setVisible(true);  
33 }
```



Ответьте на вопросы сообщением в чат

Вопросы:

- **Программный интерфейс — это способ**
 - a. рисования объектов
 - b. взаимодействия объектов
 - c. взаимодействия программы с пользователем
- **Анонимный класс — это класс без**
 - a. интерфейса
 - b. объекта
 - c. имени
- **Поле в интерфейсе**
 - a. невозможно
 - b. public static final
 - c. private final
- **Метод по-умолчанию**
 - a. можно переопределять
 - b. можно не переопределять
 - c. можно использовать с полем интерфейса
 - d. все варианты верны





Исключения в графическом интерфейсе



Исключение, которое никто не увидит

The screenshot shows an IDE with a project named 'Exceptional'. The source code for 'Exceptional.java' is displayed, showing a Swing window with a 'Push me!' button. The code is as follows:

```
7 public class Exceptional extends JFrame implements ActionListener {
8     private Exceptional() {
9         setDefaultCloseOperation(EXIT_ON_CLOSE);
10        setBounds(1100, 200, 500, 300);
11        JButton btn = new JButton("Push me!");
12        btn.addActionListener(this);
13        add(btn);
14        setVisible(true);
15    }
16
17    public static void main(String[] args) { new Exceptional(); }
18
19    @Override
20    public void actionPerformed(ActionEvent e) {
21        throw new ArrayIndexOutOfBoundsException("Bad thing happened!");
22    }
23 }
```

The window titled 'Exceptional' is visible, containing a button labeled 'Push me!'. The Run console at the bottom shows the following output:

```
Run: Exceptional
/Library/Java/JavaVirtualMachines/liberica-jdk-11.jdk/Contents/Home/bin/java -javaagent:/Applications/IntelliJ IDEA CE.a
Exception in thread "AWT-EventQueue-0" java.lang.ArrayIndexOutOfBoundsException Create breakpoint : Bad thing happened!
at ru.gb.jdk.two.online.samples.Exceptional.actionPerformed(Exceptional.java:21) <4 internal lines>
```




Пусть тайное станет явным

```
17 ▶ public static void main(String[] args) {  
18     SwingUtilities.invokeLater(new Runnable() {  
19         @Override  
20         public void run() {  
21             new Exceptional();  
22         }  
23     });  
24 }
```



Пусть тайное станет явным

```
7 ▶ public class Exceptional extends JFrame implements
8     ActionListener, Thread.UncaughtExceptionHandler {
9     private Exceptional() {...}
17
18 ▶ public static void main(String[] args) {...}
26
27     @Override
28     public void actionPerformed(ActionEvent e) {
29         throw new ArrayIndexOutOfBoundsException("Bad thing happened!");
30     }
31
32     @Override
33     public void uncaughtException(Thread t, Throwable e) {
34
35     }
36 }
```

Пусть тайное станет явным

```
10 private Exceptional() {
11     Thread.setDefaultUncaughtExceptionHandler(this);
12     setDefaultCloseOperation(EXIT_ON_CLOSE);
13     setBounds(1100, 200, 500, 300);
14     JButton btn = new JButton("Push me!");
15     btn.addActionListener(this);
16     add(btn);
17     setVisible(true);
18 }
19
20 @Override
21 public void uncaughtException(Thread t, Throwable e) {
22     JOptionPane.showMessageDialog(
23         null, e.getMessage(),
24         "Exception!", JOptionPane.ERROR_MESSAGE);
25 }
26
27 public static void main(String[] args) { ... }
```

Run: Exceptional

/Library/Java/JavaVirtualMachines/liberica-jdk-11.jdk/Contents/Home/bin/java -javaagent:/Applications/IntelliJ IDEA CE.app/Contents/lib/idea_rt.jar=63806:/Applications/IntelliJ IDEA CE.app/Content



На этом уроке

- 📌 Программные интерфейсы — понятие и принцип работы;
- 📌 Ключевое слово `implements`;
- 📌 Наследование и множественное наследование интерфейсов;
- 📌 Реализация, реализации по-умолчанию;
- 📌 Частичная реализация интерфейсов, адаптеры;
- 📌 Анонимные классы;
- 📌 Исключения в графических фреймворках.



Практическое задание

- 📌 Полностью разобраться с кодом;
- 📌 Для приложения с шариками описать появление и убирание шариков по клику мышки левой и правой кнопкой соответственно;
- 📌 Написать, выбросить и обработать такое исключение, которое не позволит создавать более, чем 15 шариков;
- 📌 ** Описать ещё одно приложение, в котором на белом фоне будут перемещаться изображения формата png, лежащие в папке проекта.





Ученикам, чтобы преуспеть, надо догонять тех,
кто впереди, и не ждать тех, кто позади.

Аристотель

