

Programação Orientada por Objetos

Polimorfismo - Exemplo

Prof. Cédric Grueau

Prof. José Sena Pereira

Departamento de Sistemas e Informática
Escola Superior de Tecnologia de Setúbal
Instituto Politécnico de Setúbal

2022/2023



Sumário



- ▶ Exemplo Desenho Casa – Solução 1
- ▶ Exemplo Desenho Casa – Solução 2
- ▶ Exemplo Desenho Casa – Herança e Polimorfismo
- ▶ Organização de Código

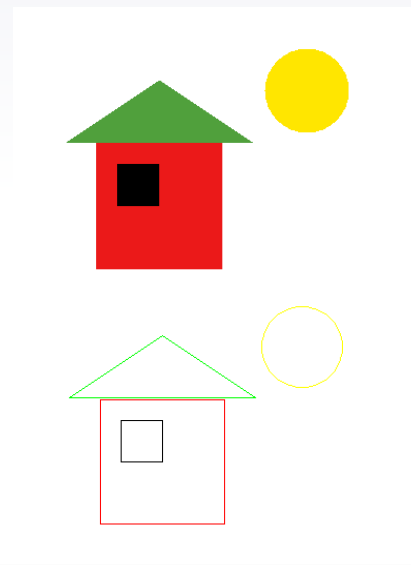
Desenho Casa – Solução 1

- ▶ Polimorfismo



Exemplo – Desenho Casa

- ▶ Requisitos do desenho (Revisitado):
 - ▶ O desenho é composto por quadrados, triângulos, círculos e retângulos.
 - ▶ Cada uma das formas geométricas tem uma cor e uma posição.
 - ▶ O desenho reutiliza as classes **Pen** e **Canvas** do ambiente gráfico. Neste caso as figuras são desenhadas apenas com as linhas de contorno.
 - ▶ Deverá ser possível criar facilmente outros desenhos com as figuras geométricas existentes (circulo, quadrado, triângulo e retângulo).



Exemplo – Desenho Casa

Foram criadas 2 soluções:

- ▶ Solução 1:
 - ▶ Classes para cada tipo de forma geométrica: **Square**, **Rectangle**, **Circle**, **Triangle**.
 - ▶ Uma classe **Figure** que pode ser qualquer das figuras existentes.
 - ▶ Uma classe **Drawing** para o desenho que guarda uma lista de figuras.

Exemplo – Desenho casa

- ▶ Classe **Position** para representar a posição das figuras:

```
public class Position {  
  
    private int x;  
    private int y;  
  
    public Position() {  
        x = 0;  
        y = 0;  
    }  
  
    public Position(int x, int y) {  
        this.x = x;  
        this.y = y;  
    }  
  
}
```

// continua ao lado...

```
    public int getX() {  
        return x;  
    }  
  
    public int getY() {  
        return y;  
    }  
  
    public void setX(int x) {  
        this.x = x;  
    }  
  
    public void setY(int y) {  
        this.y = y;  
    }  
  
}
```

Exemplo – Desenho casa

- ▶ **Solução 1** – Implementar classes para as figuras geométricas
 - ▶ Classe **Square**

```
public class Square {  
  
    private Pen pen;  
    private Color color;  
    private Position position;  
    private int side;  
  
    public Square(Pen pen, Color color, Position position, int side) {  
        this.pen = pen;  
        this.color = color;  
        this.position = position;  
        this.side = side;  
    }  
    // restante código  
}
```

É necessária uma pen externa para desenhar o quadrado na superfície associada

Exemplo – Desenho casa

- ▶ **Solução 1** – Implementar classes para as figuras geométricas

- ▶ Classe **Square**

Através dos movimentos da pen é possível efetuar o desenho duma figura

```
public class Square {  
  
    // restante código  
    public void draw() {  
        pen.setColor(color);  
        pen.penUp();  
        pen.moveTo(position.getX(), position.getY());  
        pen.penDown();  
        pen.turnTo(0);  
  
        for(int i=0; i<4; i++) {  
            pen.move(side);  
            pen.turn(90);  
        }  
    }  
}
```


Exemplo – Desenho casa

- ▶ **Solução 1** – Implementar classes para as figuras geométricas
- ▶ Classe **Circle**

Vários atributos
em comum com
a classe Square

```
public class Circle {  
    private Pen pen;  
    private Color color;  
    private Position position;  
    private int radius;  
  
    public Circle(Pen pen, Color color, Position position, int radius) {  
        this.pen = pen;  
        this.color = color;  
        this.position = position;  
        this.radius = radius;  
    }  
    // restante código  
}
```

Exemplo – Desenho casa

► Solução 1 – Implementar classes para as figuras geométricas

► Classe **Circle**

Estamos a criar círculos aproximados por polígonos regulares de 50 lados.

```
public class Circle {  
    // restante código  
    public void draw() {  
        pen.setColor(color);  
        pen.penUp();  
        pen.moveTo(position.getX(),position.getY());  
        pen.penDown();  
        pen.turnTo(0);  
  
        int sides = 50;  
        int side = (int)(2*Math.PI*radius/sides);  
        for(int i=0; i<sides+1; i++) {  
            pen.move(side);  
            pen.turn((int)(360.0/sides));  
        }  
    }  
}
```

Exemplo – Desenho casa

- ▶ **Solução 1** – Implementar classes para as figuras geométricas

- ▶ Classe **Triangle**

Com atributos
em comum com
as outras classes

```
public class Triangle {  
    private Pen pen;  
    private Color color;  
    private Position position  
    private int height, width;  
  
    // restante código  
    void draw() {  
        pen.penUp();  
        pen.moveTo(x, y);  
        pen.penDown();  
        pen.turnTo(0);  
  
        pen.moveTo(x + (width / 2), y - height);  
        pen.moveTo(x + width, y);  
        pen.moveTo(x, y);  
    }  
}
```

Exemplo – Desenho casa

- ▶ **Solução 1** – Implementar classes para as figuras geométricas
 - ▶ Classe **Rectangle**

Com atributos
em comum com
as outras classes

```
public class Rectangle {  
    private Pen pen;  
    private Color color;  
    private Position position  
    private int height, width;  
    void draw() {  
        pen.penUp();  
        pen.moveTo(x,y);  
        pen.penDown();  
        pen.turnTo(0);  
        pen.move(width);  
        pen.turn(90);  
        pen.move(height);  
        pen.turn(90);  
        pen.move(width);  
        pen.turn(90);  
        pen.move(height);  
        pen.turn(90);  
    } // restante código  
}
```

Exemplo – Desenho casa

- ▶ **Solução 1** – Implementar uma classe para representar figuras
 - ▶ Classe **Figure**

```
public class Figure {  
  
    private FigureType figureType;  
    private Square square;  
    private Retangulo rectangle;  
    private Triangulo triangle;  
    private circle circle;  
  
    // restante código  
}
```

Tipo enumerado **FigureType** para definir o tipo de figura guardado

Atributos necessários para os vários tipos de figura que poderão ser utilizados

Exemplo – Desenho casa

- ▶ **Solução 1** – Implementar uma classe para representar figuras
 - ▶ Tipo **FigureType**

```
public enum FigureType {  
  
    TRIANGLE, CIRCLE, RECTANGLE, SQUARE  
  
}
```

Exemplo – Desenho casa

- **Solução 1** – Implementar uma classe para representar figuras

- Classe **Figure** - Construtores

```
public Figure(FigureType figureType, Pen pen, Color color,
              Position position, int dimension) {
    this.figureType = figureType;
    if (figureType == FigureType.CIRCLE) {
        circle = new circle(pen, color, position, dimension);
    } else {
        square = new Square(pen, color, position, dimension);
    }
}

public Figure(FigureType figureType, Pen pen, Color color,
              Position position, int height, int width) {
    this.figureType = figureType;
    if (figureType == FigureType.TRIANGLE) {
        triangle = new Triangulo(pen, color, position, height, width);
    } else {
        rectangle = new Retangulo(pen, color, position, height, width);
    }
}
```



Confuso!

Exemplo – Desenho casa

- ▶ **Solução 1** – Implementar uma classe para representar figuras
 - ▶ Classe **Figure** – **draw**

Um switch para lidar com os diferentes tipos de figura

Muitas vezes as instruções **switch** são pistas que nos indicam que poderá existir um **problema de coesão** onde estão representadas várias entidades e em que a **solução passa pela herança (e polimorfismo)**

```
public void draw() {  
    switch (figureType) {  
        case CIRCLE:  
            circle.draw();  
            break;  
        case TRIANGLE:  
            triangle.draw();  
            break;  
        case RECTANGLE:  
            rectangle.draw();  
            break;  
        case SQUARE:  
            square.draw();  
            break;  
    }  
}
```


Exemplo – Desenho casa

- ▶ **Solução 1** – Implementar uma classe para representar figuras
 - ▶ Classe **Drawing**

```
public class Drawing {  
    private ArrayList<Figure> figures;  
  
    public Drawing() {  
        figures = new ArrayList<>();  
    }  
  
    public void addFigure(Figure figure) {  
        if (figure != null) {  
            figures.add(figure);  
        }  
    }  
  
    public void draw() {  
        for (Figure figure : figures) {  
            figure.draw();  
        }  
    }  
}
```

Exemplo – Desenho casa

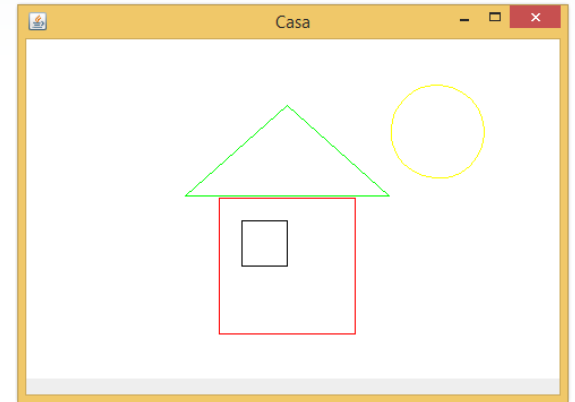
- ▶ **Solução 1** – Programa principal
 - ▶ Ex: representar o **desenho da casa**

```
public class Program {  
  
    public static void main() {  
        Canvas canvas = new Canvas("House", 500, 300, Color.WHITE);  
        Pen pen = new Pen(0, 0, canvas);  
        Drawing drawing = new Drawing();  
  
        Figure wall = new Figure(FigureType.SQUARE, pen, Color.RED,  
                                new Position(170, 140), 120);  
        Figure window = new Figure(FigureType.SQUARE, pen, Color.BLACK,  
                                   new Position(190, 160), 40);  
        Figure roof = new Figure(FigureType.TRIANGLE, pen, Color.GREEN,  
                                 new Position(140, 138), 80, 180);  
        Figure sun = new Figure(FigureType.CIRCLE, pen, Color.YELLOW,  
                                new Position(360, 40), 40);  
  
        // continua
```

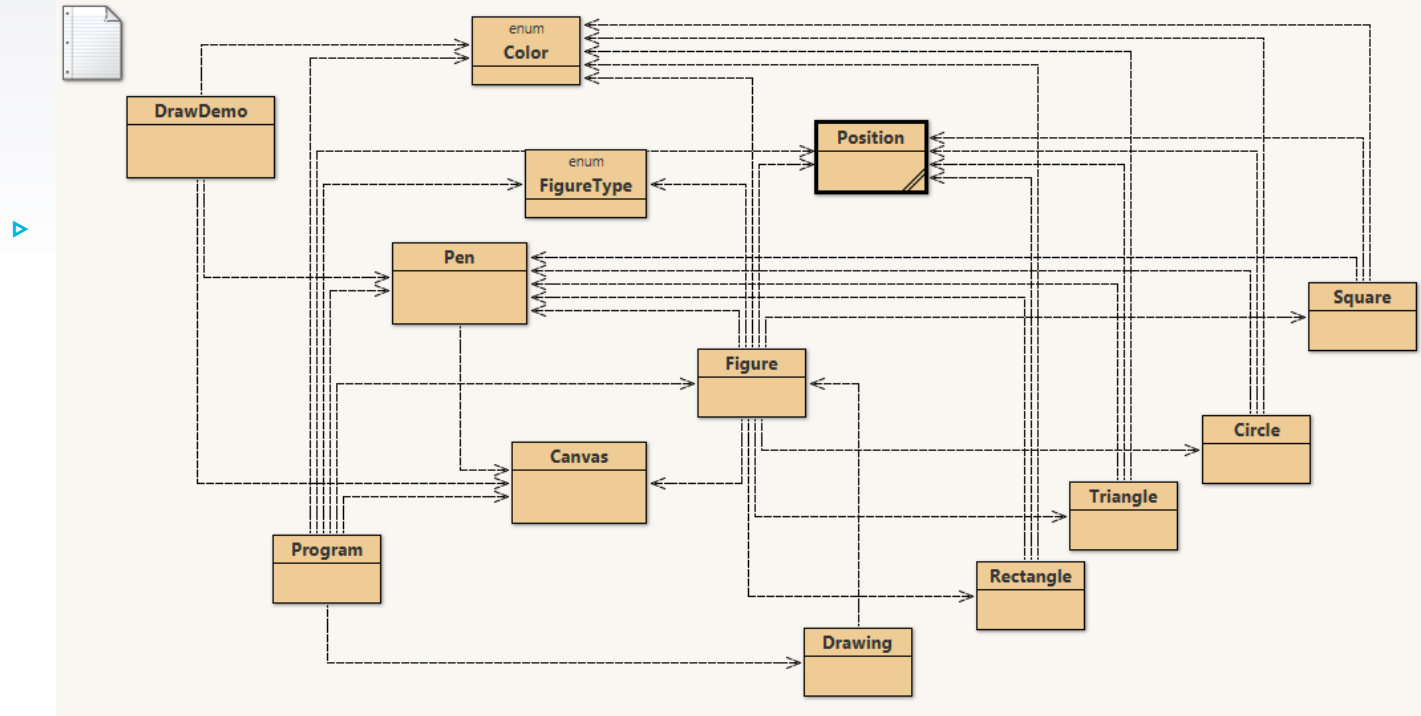
Exemplo – Desenho casa

- ▶ **Solução 1** – Programa principal
 - ▶ Ex: representar o **desenho da casa**

```
        drawing.addFigure(wall);  
        drawing.addFigure(window);  
        drawing.addFigure(roof);  
        drawing.addFigure(sun);  
  
        drawing.draw();  
    }  
}
```



Exemplo – Desenho casa



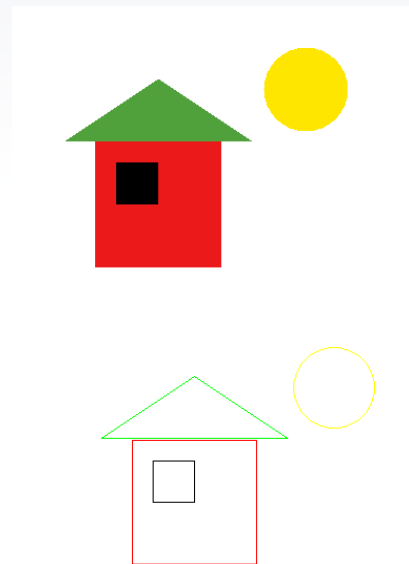
Desenho Casa – Solução 2

- Polimorfismo



Exemplo – Desenho Casa

- ▶ Requisitos do desenho (Revisitado):
 - ▶ O desenho é composto por quadrados, triângulos, círculos e retângulos.
 - ▶ Cada uma das formas geométricas tem uma cor e uma posição.
 - ▶ O desenho reutiliza as classes **Pen** e **Canvas** do ambiente gráfico. Neste caso as figuras são desenhadas apenas com as linhas de contorno.
 - ▶ Deverá ser possível criar facilmente outros desenhos com as figuras geométricas existentes (círculo, quadrado, triângulo e retângulo).



Exemplo – Desenho Casa

- ▶ Foram criadas 2 soluções:
 - ▶ Solução 1:
 - ▶ Classes para cada tipo de forma geométrica: **Square**, **Rectangle**, **Circle**, **Triangle**.
 - ▶ Um classe **Figure** que pode ser qualquer das figuras existentes.
 - ▶ Uma classe **Drawing** para o desenho que guarda uma lista de figuras.
 - ▶ Solução 2:
 - ▶ Uma classe **Figure** que representa qualquer figura.
 - ▶ Uma classe **Drawing** para o desenho que guarda uma lista de figuras.

Exemplo – Desenho casa

- ▶ **Solução 2** – Implementar uma classe para representar figuras
 - ▶ Classe **Figure**

```
public class Figure {  
  
    private FigureType type;  
    private Pen pen;  
    private Position position;  
    private Color color;  
    private int height;  
    private int width;  
    private int radius;  
  
    // restante código  
}
```

Tipo enumerado **TipoFigura** para definir o tipo de figura guardado

Atributos necessários para os vários tipos de figura que poderão ser utilizados

Exemplo – Desenho casa

- ▶ **Solução 2** – Implementar uma classe para representar figuras
 - ▶ Tipo **FigureType**

```
public enum FigureType {  
    NONE, TRIANGLE, CIRCLE, RECTANGLE, SQUARE  
}
```

Exemplo – Desenho casa

- ▶ **Solução 2** – Implementar uma classe para representar figuras
 - ▶ Classe **Figure** – Construtor

```
public Figure(Pen pen) {  
    this.pen = pen;  
    type = FigureType.NONE;  
    position = new Position();  
}
```

Exemplo – Desenho casa

- ▶ **Solução 2** – Implementar uma classe para representar figuras
 - ▶ Classe **Figure** – **setCircle** e **setSquare**

```
public void setCircle(Position position, int radius) {  
    type = FigureType.CIRCLE;  
    this.position = position;  
    this.radius = radius;  
}  
public void setSquare(Position position, int side) {  
    type = FigureType.SQUARE;  
    this.position = position;  
    this.width = side;  
    this.height = side;  
}
```

Exemplo – Desenho casa

- ▶ **Solução 2** – Implementar uma classe para representar figuras
 - ▶ Classe **Figure** – **setRectangle** e **setTriangle**

```
public void setRectangle(Position position, int width, int height) {  
    type = FigureType.RECTANGLE;  
    this.position = position;  
    this.width = width;  
    this.height = height;  
}  
  
public void setTriangle(Position position, int base, int height) {  
    type = FigureType.TRIANGLE;  
    this.position = position;  
    this.width = base;  
    this.height = height;  
}
```

Exemplo – Desenho casa

- ▶ **Solução 2** – Implementar uma classe para representar figuras
 - ▶ Classe **Figure** – **draw**

```
public void draw() {  
    if (position == null || type == FigureType.NONE || pen == null) {  
        return;  
    }  
  
    pen.penUp();  
    pen.moveTo(position.getX(), position.getY());  
    pen.setColor(color);  
    pen.penDown();  
  
    // continua...
```

Exemplo – Desenho casa

- ▶ **Solução 2** – Implementar uma classe para representar figuras
 - ▶ Classe **Figure** – **draw**

```
switch (type) {  
    case CIRCLE:  
        drawCircle();  
        break;  
    case TRIANGLE:  
        drawTriangle();  
        break;  
    case RECTANGLE:  
        drawRectangle();  
        break;  
    case SQUARE:  
        drawSquare();  
        break;  
}
```

Exemplo – Desenho casa

- ▶ **Solução 2** – Implementar uma classe para representar figuras
 - ▶ Classe **Figure** – **drawSquare** e **drawCircle**

```
private void drawSquare() {  
    for (int i = 0; i < 4; i++) {  
        pen.move(width);  
        pen.turn(90);  
    }  
}  
private void drawCircle() {  
    int sides = 50;  
    int side = (int) (2 * Math.PI * radius / sides);  
  
    for (int i = 0; i < sides + 1; i++) {  
        pen.move(side);  
        pen.turn((int) (360.0 / sides));  
    }  
}
```

Exemplo – Desenho casa

- ▶ **Solução 2** – Implementar uma classe para representar figuras
 - ▶ Classe **Figure** – **drawRectangle**

```
private void drawRectangle() {  
    pen.move(width);  
    pen.turn(90);  
    pen.move(height);  
    pen.turn(90);  
    pen.move(width);  
    pen.turn(90);  
    pen.move(height);  
    pen.turn(90);  
}
```


Exemplo – Desenho casa

- ▶ **Solução 2** – Implementar uma classe para representar figuras
 - ▶ Classe **Figure** – **drawTriangle**

```
private void drawTriangle() {  
    int x = position.getX();  
    int y = position.getY();  
  
    pen.moveTo(x + (width / 2), y - height);  
    pen.moveTo(x + width, y);  
    pen.moveTo(x, y);  
}
```

Exemplo – Desenho casa

- ▶ **Solução 2** – Implementar uma classe para representar figuras

- ▶ Classe **Drawing**

Idêntica à solução 1

```
public class Drawing {  
    private ArrayList<Figure> figures;  
  
    public Drawing() {  
        figures = new ArrayList<>();  
    }  
  
    public void addFigure(Figure figure) {  
        if (figure != null) {  
            figures.add(figure);  
        }  
    }  
  
    public void draw() {  
        for (Figure figure : figures) {  
            figure.draw();  
        }  
    }  
}
```

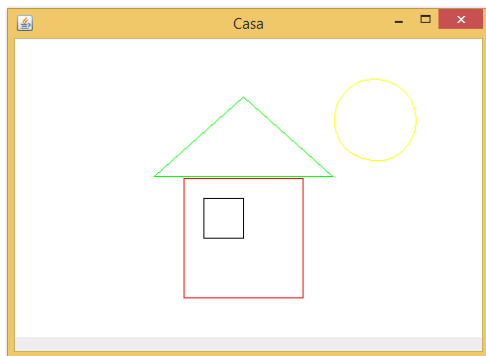
Exemplo – Desenho casa

- ▶ **Solução 2** – Programa principal
 - ▶ Ex: representar o **desenho da casa**

```
public class Program {  
  
    public static void main() {  
        Canvas canvas = new Canvas("House", 500, 300, Color.WHITE);  
        Pen pen = new Pen(0, 0, canvas);  
        Drawing drawing = new Drawing();  
  
        Figure wall = new Figure(pen);  
        wall.setCor(Color.RED);  
        wall.setSquare(new Position(170, 140), 120);  
        drawing.addFigure(wall);  
  
        Figure window = new Figure(pen);  
        window.setCor(Color.BLACK);  
        window.setSquare(new Position(190, 160), 40);  
        drawing.addFigure(window);  
        // continua  
    }  
}
```

Exemplo – Desenho casa

- ▶ **Solução 2** – Programa principal
 - ▶ Ex: representar o **desenho da casa**



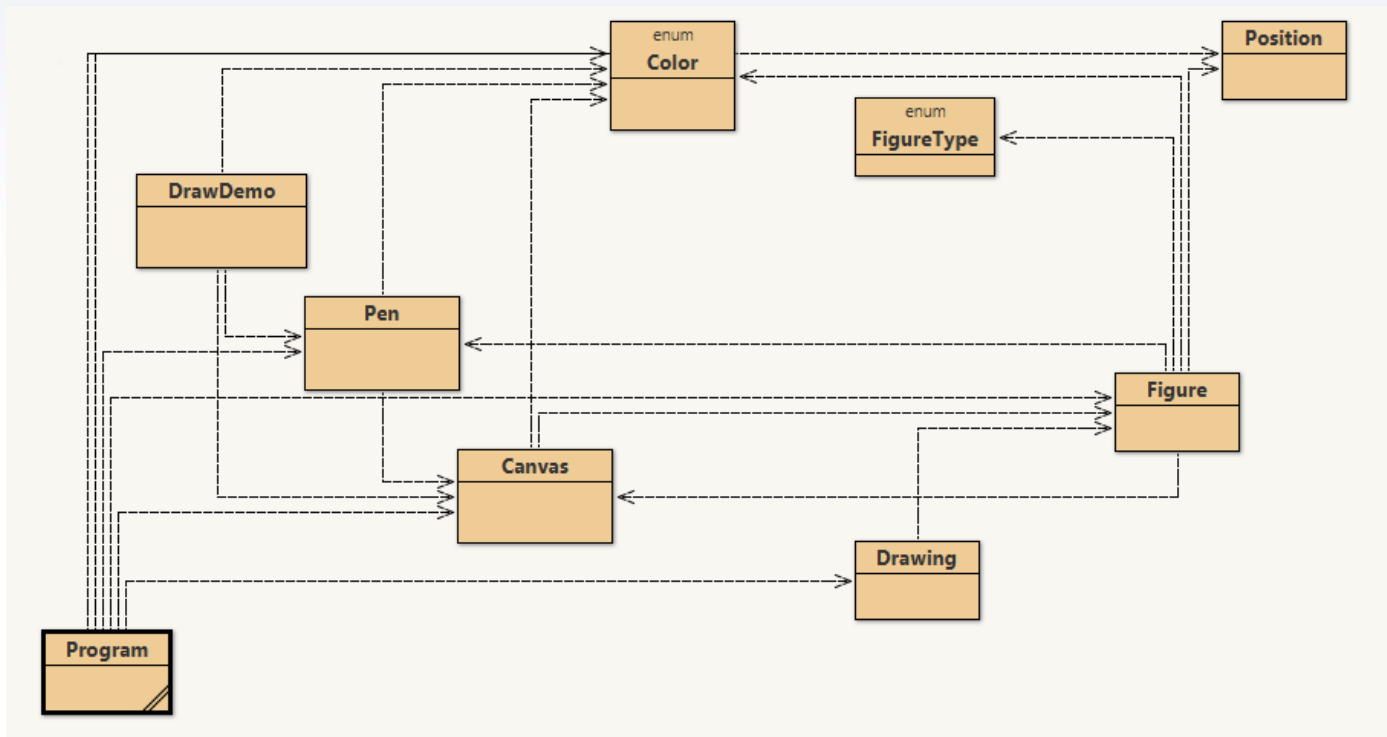
```
Figure roof = new Figure(pen);  
roof.setCor(Color.GREEN);  
roof.setTriangle(new Position(140, 138), 180, 60);  
drawing.addFigure(roof);
```

```
Figure sun = new Figure(pen);  
sun.setCor(Color.YELLOW);  
sun.setCircle(new Position(360, 50), 40);  
drawing.addFigure(sun);
```

```
drawing.draw();
```

```
}
```

Exemplo – Desenho casa





Desenho Casa: Herança e Polimorfismo



- ▶ Polimorfismo

Exemplo – Desenho Casa

- ▶ **Problemas das soluções** apresentadas:
 - ▶ Solução 1
 - ▶ Código duplicado nas várias formas geométricas
 - ▶ Classe **Figure** com um atributo por cada tipo de forma geométrica
 - ▶ Solução 2
 - ▶ Classe **Figure** complexa
 - ▶ Classe **Figure** com problemas de coesão ao representar as várias formas geométricas

Exemplo – Desenho Casa

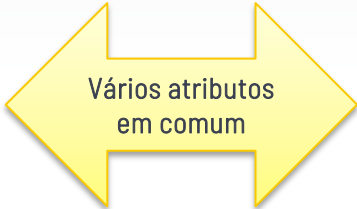
- ▶ Solução com **herança de classes** e **polimorfismo**:
 - ▶ Criar uma **superclasse Figure** com os atributos comuns das formas geométricas
 - ▶ Criar **sublasses** para cada uma das formas geométricas.
 - ▶ Criar uma classe **Drawing** que irá guardar uma lista de formas geométricas tirando partido do Principio da Substituição
 - ▶ Método **draw** da classe Figure polimórfico.
 - ▶ O resultado da sua execução depende da figura que estiver na lista.

Exemplo – Desenho Casa

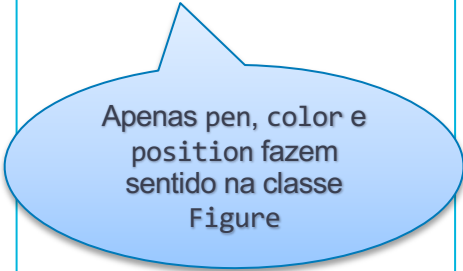
▶ Classes da solução 1:

```
public class Square {  
  
    private Pen pen;  
    private Color color;  
    private Position position;  
    private int side;  
  
    // restante código  
}
```

```
public class Circle {  
  
    private Pen pen;  
    private Color color;  
    private Position position;  
    private int radius;  
  
    // restante código  
}
```



Vários atributos
em comum



Apenas pen, color e
position fazem
sentido na classe
Figure

```
public class Rectangle {  
  
    private Pen pen;  
    private Color color;  
    private Position position;  
    private int height, width;  
  
    // restante código  
}  
  
public class Triangle {  
  
    private Pen pen;  
    private Color color;  
    private Position position;  
    private int height, width;  
  
    // restante código  
}
```

Exemplo – Desenho Casa

- **Solução 3** – classe base **Figure**

Construtor sem
argumentos

```
public class Figure {  
  
    private Pen pen;  
    private Position position;  
    private Color color;  
  
    public Figure() {  
        pen = new Pen();  
        position = new Position();  
        color = Color.BLACK;  
    }  
  
    // restante código  
}
```

Exemplo – Desenho Casa

► Solução 3 – classe base **Figure** – construtores

```
public Figure() {
    pen = new Pen();
    position = new Position();
    color = Color.BLACK;
}

public Figure(Pen pen, Color color) {
    if (pen != null) {
        this.pen = pen;
    } else {
        this.pen = new Pen();
    }
    position = new Position();
    if (color != null) {
        this.color = color;
    } else {
        this.color = Color.BLACK;
    }
}
```

Aqui também existe
alguma duplicação
de código

```
public Figure(Position position,
              Pen pen, Color color) {
    if (pen != null) {
        this.pen = pen;
    } else {
        this.pen = new Pen();
    }
    if (position != null) {
        this.position =
            new Position(position.getX(),
                          position.getY());
    } else {
        this.position = new Position();
    }
    if (color != null) {
        this.color = color;
    } else {
        this.color = Color.BLACK;
    }
}
```

Construtores – `this()`

- ▶ **this()** nos construtores
 - ▶ Da mesma maneira que se utiliza **super()** para a chamada ao construtor da classe base é possível usar **this()** para a chamada doutro construtor da própria classe.
 - ▶ Neste caso mantém-se a obrigatoriedade da instrução **this()** ser a primeira do construtor

```
public class Clock {  
  
    private int hours;  
    private int minutes;  
    private int seconds;  
  
    public Clock(){  
        this(0,0,0);  
    }  
  
    public Clock(int hours, int minutes){  
        this(hours, minutes, 0);  
    }  
  
    public Clock(int hours, int minutes, int seconds){  
        this.hours = hours;  
        this.minutes = minutes;  
        this.seconds = seconds;  
    }  
  
    // [...]  
}
```

Chamada ao construtor:
Clock(int hours, int minutes, int seconds)

Exemplo – Desenho Casa

- ▶ Solução 3 – classe base **Figure** – construtores

```
public Figure() {  
    this(new Position(), new Pen(), Color.BLACK);  
}  
  
public Figure(Pen pen, Color color) {  
    this(new Position(), pen, color);  
}
```

Assim evita-se a
duplicação de código nos
construtores

```
public Figure(Position position,  
              Pen pen, Color color) {  
    if (pen != null) {  
        this.pen = pen;  
    } else {  
        this.pen = new Pen();  
    }  
    if (position != null) {  
        this.position =  
            new Position(position.getX(),  
                          position.getY());  
    } else {  
        this.position = new Position();  
    }  
    if (color != null) {  
        this.color = color;  
    } else {  
        this.color = Color.BLACK;  
    }  
}
```

Exemplo – Desenho Casa

- ▶ Solução 3 – classe base **Figure** – construtores

```
public Figure() {  
    this(new Position(), new Pen(), Color.BLACK);  
}  
  
public Figure(Pen pen, Color color) {  
    this(new Position(), pen, color);  
}
```

Outra simplificação que pode ser feita é a utilização do operador ternário em substituição dos if

```
public Figure(Position position,  
              Pen pen, Color color) {  
    if (pen != null) {  
        this.pen = pen;  
    } else {  
        this.pen = new Pen();  
    }  
    if (position != null) {  
        this.position =  
            new Position(position.getX(),  
                          position.getY());  
    } else {  
        this.position = new Position();  
    }  
    if (color != null) {  
        this.color = color;  
    } else {  
        this.color = Color.BLACK;  
    }  
}
```

Operador ternário

- ▶ Operador ternário - **?!**

Condição lógica valor se a condição for verdadeira valor se a condição for falsa

↓ ↓ ↙

condição ? valor 1 : valor 2

- Exemplo:

```
String period = (hours < 12) ? "am" : "pm";
```

- Equivalente com **ifs**:

```
String period;  
if (hours < 12) {  
    period = "am";  
} else {  
    period = "pm";  
}
```

Exemplo – Desenho Casa

- **Solução 3** – classe base **Figure** – construtores

```
public Figure() {  
    this(new Position(), new Pen(), Color.BLACK);  
}  
  
public Figure(Pen pen, Color color) {  
    this(new Position(), pen, color);  
}  
  
public Figure(Position position, Pen pen, Color color) {  
    this.pen = (pen != null) ? pen : new Pen();  
    this.position = (position != null) ? new Position(position.getX(), position.getY()) : new Position();  
    this.color = (color != null) ? color : Color.BLACK;  
}
```


Exemplo – Desenho Casa

- ▶ **Solução 3** – classe derivada **Circle**

```
public class Circle extends Figure {  
  
    private int radius;  
  
    public Circle() {  
        this.radius = 1;  
    }  
  
    public Circle(int radius) {  
        this.radius = radius;  
    }  
  
    public Circle(int radius, Position position, Pen pen, Color color) {  
        super(position, pen, color);  
        this.radius = radius;  
    }  
  
    // restante código  
}
```

Exemplo – Desenho Casa

- **Solução 3** – classe derivada **Circle**: método **draw**

```
@Override
public void draw() {
    Pen pen = getPen();
    pen.setColor(getColor());
    pen.penUp();
    pen.moveTo(getX(), getY());
    pen.penDown();
    pen.turnTo(0);

    int sides = 50;
    int side = (int) (2 * Math.PI * radius / sides);

    for (int i = 0; i < sides + 1; i++) {
        pen.move(side);
        pen.turn((int) (360.0 / sides));
    }
}
```

Exemplo – Desenho Casa

- **Solução 3** – classe derivada **Square**

```
public class Square extends Figure {  
  
    private int side;  
  
    public Square() {  
        this.side = 1;  
    }  
  
    public Square(int side) {  
        this.side = side;  
    }  
  
    public Square(int side, Position position, Pen pen, Color color) {  
        super(position, pen, color);  
        this.side = side;  
    }  
  
    // restante código  
}
```

Exemplo – Desenho Casa

- ▶ **Solução 3** – classe derivada
Square: método **draw**

```
@Override
public void draw() {
    Pen pen = getPen();
    pen.setColor(getColor());
    pen.penUp();
    pen.moveTo(getX(),getY());
    pen.penDown();
    pen.turnTo(0);

    for (int i = 0; i < 4; i++) {
        pen.move(side);
        pen.turn(90);
    }
}
```

Exemplo – Desenho casa

- ▶ **Solução 3** – Implementar uma classe para representar desenhos
 - ▶ Classe **Drawing**

Esta classe mantém-se idêntica.

```
public class Drawing {  
  
    private ArrayList<Figure> figures;  
  
    public Drawing() {  
        figures = new ArrayList<>();  
    }  
  
    public void addFigure(Figure figure) {  
        if (figure != null) {  
            figures.add(figure);  
        }  
    }  
  
    public void draw() {  
        for (Figure figure : figures) {  
            figure.draw();  
        }  
    }  
}
```

Exemplo – Desenho casa

- ▶ **Solução 3** – Implementar uma classe para representar desenhos
 - ▶ Classe **Drawing**

O polimorfismo manifesta-se no método draw. Cada figura desenha-se à sua maneira

```
public class Drawing {  
  
    private ArrayList<Figure> figures;  
  
    public Drawing() {  
        figures = new ArrayList<>();  
    }  
  
    public void addFigure(Figure figure) {  
        if (figure != null) {  
            figures.add(figure);  
        }  
    }  
  
    public void draw() {  
        for (Figure figure : figures) {  
            figure.draw();  
        }  
    }  
}
```

Exemplo – Desenho casa

- ▶ **Solução 3** – Implementar uma classe para representar desenhos
 - ▶ Classe **Drawing**

Não é necessária a utilização de nenhum switch. Com o polimorfismo é selecionado automaticamente o método draw a ser executado

```
public class Drawing {  
  
    private ArrayList<Figure> figures;  
  
    public Drawing() {  
        figures = new ArrayList<>();  
    }  
  
    public void addFigure(Figure figure) {  
        if (figure != null) {  
            figures.add(figure);  
        }  
    }  
  
    public void draw() {  
        for (Figure figure : figures) {  
            figure.draw();  
        }  
    }  
}
```

Exemplo – Desenho casa

- ▶ **Solução 3** – programa principal
 - ▶ Ex: representar o **desenho da casa**

```
public class Program {  
  
    public static void main(String[] args) {  
  
        Canvas canvas = new Canvas("Casa", 500, 300, Color.WHITE);  
        Pen pen = new Pen(0, 0, canvas);  
        Drawing drawing = new Drawing();  
  
        Square wall = new Square(120, new Position(170, 140), pen, Color.RED);  
        Square window = new Square(40, new Position(190, 160), pen, Color.BLACK);  
        Triangle roof = new Triangle(180, 80, new Position(140, 138), pen, Color.GREEN);  
        Circle sun = new Circle(40, new Position(360, 40), pen, Color.YELLOW);  
  
        drawing.addFigure(wall);  
        drawing.addFigure(window);  
        drawing.addFigure(roof);  
        drawing.addFigure(sun);  
  
        drawing.draw();  
    }  
}
```

Código idêntico
ao da solução 1

Exemplo – Desenho casa

- **Solução 3** – Diagrama de classes

