**Code Analysis**

1. Assume we want to create a new JComponent called TwoHomers that will display two Homer drawings side by side:

A picture containing chart

Description automatically generated

We are going to evaluate the following code examples to determine which is the “best” solution.

What criteria should we use?

* Easy to read/debug
* Ability to be reused
* Good commenting
* Efficiency
* Simplicity
* Easy to understand
* Not to many lines of code
* Function as intended

Provide the pro’s and con’s of each of the following coding options:

***Option 1:  Modify the main method***

public static void main(String[] args)

{

  JFrame frame = new JFrame();

  JPanel contents = new JPanel();

  StickFigure homer = new StickFigure();

  StickFigure homer2 = new StickFigure();

  contents.add(homer);

  contents.add(homer2);

  frame.setContentPane(contents);

  frame.pack();

  frame.setVisible(true);

 }

}

* Pro: OOP so its scalable , Con: Tedious to add the StickFigure object to the JPanel each time so you can make a parameter in the StickFigure constructor to pass it in and add it in the StickFigure Class - it just makes it easier to add multiple StickFigures

It does not have the single JComponent to draw two Homers

-

* Pros: can be used for multiple homers

-Cons: No comment lines

***Option 2:  Create a Helper Method that Draws the Figure Twice***

   public void paintComponent(Graphics g)

   {

      super.paintComponent(g);

      this.paintHomer(g);

      this.paintHomer(g);

   }

public void paintHomer(Graphics g)

  {

     //Head

     g.setColor(Color.YELLOW);

     g.fillOval(60, 0, 60, 60);

     //Body

     g.setColor(Color.WHITE);

     g.fillRect(0, 60, 180, 30);

     g.fillRect(30, 60, 120, 90);

     //Legs

     g.setColor(Color.BLUE);

     g.fillRect(30, 150, 120, 120);

     g.setColor(Color.BLACK);

     g.drawLine(90, 150, 90, 270);

  }

* It draws over the first stickfigure
* Con: It may paint over the first homer

***Option 3:  Paint Two Homers***

public void paintComponent(Graphics g)

{

      super.paintComponent(g);

     //Head1

     g.setColor(Color.YELLOW);

     g.fillOval(60, 0, 60, 60);

     //Body1

     g.setColor(Color.WHITE);

     g.fillRect(0, 60, 180, 30);

     g.fillRect(30, 60, 120, 90);

     //Legs1

     g.setColor(Color.BLUE);

     g.fillRect(30, 150, 120, 120);

     g.setColor(Color.BLACK);

     g.drawLine(90, 150, 90, 270);

     //Head2

     g.setColor(Color.YELLOW);

     g.fillOval(240, 0, 60, 60);

     //Body2

     g.setColor(Color.WHITE);

     g.fillRect(180, 60, 180, 30);

     g.fillRect(210, 60, 120, 90);

     //Legs2

     g.setColor(Color.BLUE);

     g.fillRect(210, 150, 120, 120);

     g.setColor(Color.BLACK);

     g.drawLine(270, 150, 270, 270);

}

-con: it is long and doesn’t have this.paintHomer

- Con: Not scalable (can only paint 2 homers at the fixed coordinates)

* more difficult to read

-no constructor

unnecessary added sections(could fit all heads in one, body, etc)

-pro: at least it does what’s intended

* Hard coding each Homer is inefficient; not a scalable approach

***Option 4:  Create a Helper Method to Paint Second Homer***

public void paintComponent(Graphics g)

{

      super.paintComponent(g);

      this.paintHomer(g);

      this.paintSecondHomer(g);

}

private void paintHomer(Graphics g)

{

     //Head1

     g.setColor(Color.YELLOW);

     g.fillOval(60, 0, 60, 60);

     //Body1

     g.setColor(Color.WHITE);

     g.fillRect(0, 60, 180, 30);

     g.fillRect(30, 60, 120, 90);

     //Legs1

     g.setColor(Color.BLUE);

     g.fillRect(30, 150, 120, 120);

     g.setColor(Color.BLACK);

     g.drawLine(90, 150, 90, 270);

}

public void paintSecondHomer(Graphics g)

{

     //Head

     g.setColor(Color.YELLOW);

     g.fillOval(240, 0, 60, 60);

     //Body

     g.setColor(Color.WHITE);

     g.fillRect(180, 60, 180, 30);

     g.fillRect(210, 60, 120, 90);

     //Legs

     g.setColor(Color.BLUE);

     g.fillRect(210, 150, 120, 120);

     g.setColor(Color.BLACK);

     g.drawLine(270, 150, 270, 270);

}

* Con: Lots of code has been reused
* Still not scalable as you have to hardcode a different method each time you want to draw Homer
* It has two methods to paint Homer

***Option 5:  Create a Helper Method with Extra Parameters***

public void paintComponent(Graphics g)

{

      super.paintComponent(g);

      this.paintHomer(g, 0, 0);

      this.paintHomer(g, 182, 0);

}

private void paintHomer(Graphics g2, int x, int y)

{

      // Head

      g2.setColor(Color.YELLOW);

      g2.fillOval(x+60, y+0, 60, 60);

      // Body

      g2.setColor(Color.WHITE);

      g2.fillRect(x+0, y+60, 180, 40);

      g2.fillRect(x+40, y+60, 100, 90);

      // Legs

      g2.setColor(Color.BLUE);

      g2.fillRect(x+40, y+150, 100, 120);

      g2.setColor(Color.BLACK);

      g2.drawLine(x+90, y+180, x+90, y+270);

}

* clear and concise, easy to understand
* properly makes a a paintHomer helper method
* pro:Does what's asked in one command
* Pro: Scalable (takes in x,y as arguments)/ Con: Could make this even more scalable by taking in x,y coordinates in the class constructor and passing it into the paintComponent method to allow the user to change the x,y position each time its called