Práctica 3

Lienzo

```
import java.awt.*;
import javax.swing.JFrame;

public class Lienzo extends Canvas {
  private Figura figuras[];
  private boolean guardarFiguras;

  public void setGuardarFiguras(boolean guardarFiguras) {
    this.guardarFiguras = guardarFiguras;
  }

  public boolean getGuardarFiguras() {
    return this.guardarFiguras;
  }

  /**
   * Draws one square.
   * @param figura
   */
  public void pintar(Figura figura)
  {
    Figura figuras[] = new Figura[]{figura};
    // Call more generic method
    this.pintar(figuras);
  }

  /**
   * Paints multiple squares at once
   * @param figuras
  */
  public void pintar(Figura figuras[]) {
    if (this.getGuardarFiguras()) {
      // If flag guardarFiguras is activated, add figura to buffer
      this.addFiguras(figuras);
    } else {
      this.figuras = figuras;
    }
  }

  /**
   * Adds figuras to the figuras buffer
   * @param figuras
   * @return
   */
  private Figura[] addFiguras(Figura figuras[]) {
    // Copy old buffer and clear
    Figura oldBuffer[] = this.clearFiguras();
    // Create new buffer with extra spaces
    this.figuras = new Figura[oldBuffer.length + figuras.length];
    // New buffer is the sum of old buffer plus new items
    this.concatenateToFiguras(oldBuffer);
    this.concatenateToFiguras(figuras);

    // Return just in case user wants to use them
    return this.figuras;
  }

  /**
   * Concatenate an array of figuras to the figuras buffer.
   * @param figuras
   */
  private void concatenateToFiguras(Figura figuras[]) {
    int position = 0;
    for (int i = 0; i < this.figuras.length; i++) {
      if (this.figuras[i] == null && position < figuras.length) {
```

```java
        this.figuras[i] = figuras[position];
        position++;
      }
    }
  }

  /**
   * Return figuras array and clear
   * @return
   */
  public Figura[] clearFiguras() {
    Figura oldBuffer[];
    if (this.figuras == null) {
      oldBuffer = new Figura[0];
    } else {
      oldBuffer = this.figuras;
    }

    this.figuras = new Figura[0];
    return oldBuffer;
  }

  public void removeFigura(int pos) {
    if (pos >= 0 && pos < this.figuras.length) {
      // Very lazy!! Change so that position is eliminated not nulled
      this.figuras[pos] = null;
    }
  }

  /**
   * Paints figuras in figuras buffer
   * @param graphics
   */
  public void paint(Graphics graphics)
  {
    for (Figura figura:this.figuras) {

      if (figura != null) {
        figura.pintar(graphics);
      }
    }
  }
}
```

Dibujo

```java
import java.awt.*;
import javax.swing.JFrame;

/**
 * Updated Dibujo class
 * Modifications:
 *   - Updated pintar for Figura (or multiple figura)
 *   - Updated pintarPath for Figura
 */

public class Dibujo extends JFrame {
  // Frame attributes
  private Lienzo lienzo;

  private int sizeX;
  private int sizeY;

  public final int MAX_SIZE_X = 1920;
  public final int MAX_SIZE_Y = 1080;
  public final int DEFAULT_SIZE = 600;

  public Dibujo(int sizeX, int sizeY, boolean guardarFiguras)
  {
    super("Dibujo");

    // Set canvas size
    this.setSizeX(sizeX);
    this.setSizeY(sizeY);

    // Create canvas
    lienzo = new Lienzo();
    // Possible improvement, join in Lienzo constructor
    lienzo.setSize(this.getSizeX(), this.getSizeY());
    lienzo.setGuardarFiguras(guardarFiguras);

    this.add(lienzo);
    this.pack();
    this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    this.setVisible(true);
  }

  public Dibujo() {
    this(0,0, true); // Create canvas of default size
  }

  public void removeFigura(int pos) {
    lienzo.removeFigura(pos);
  }

  public void clearFiguras() {
    lienzo.clearFiguras();
  }

  public void setSizeX(int sizeX) {
    if (sizeX > 0 && sizeX < MAX_SIZE_X) {
      this.sizeX = sizeX;
    } else {
      this.sizeX = DEFAULT_SIZE;
    }
  }

  public int getSizeX() {
    return this.sizeX;
  }

  public void setSizeY(int sizeY) {
    if (sizeY > 0 && sizeY < MAX_SIZE_Y) {
      this.sizeY = sizeY;
    } else {
      this.sizeY = DEFAULT_SIZE;
    }
  }

  public int getSizeY() {
    return this.sizeY;
  }

  // Wrapper for single Figura
  public void pintar(Figura figura) {
    Figura figuras[] = {figura};
    this.pintar(figuras);
  }

  public void pintar(Figura figuras[]) {
    lienzo.pintar(figuras);
    this.repintar();
  }

  public void repintar() {
    lienzo.repaint();
  }

  /**
   * Renders the square and makes it go smoothly through the points passed.
   * From point a to point b it will generate n steps.
   * Framerate will determine how fast it will complete each step.
   * @param figura
   * @param points
   * @param steps
   * @param frameRate
   */
  public void pintarPath(Figura figura, Point[] points, int steps, int frameRate) {
    // Move to initial position
    figura.moveTo(points[0].getX(), points[0].getY());

    // Waiting times
    int waitTime = steps / frameRate;

    for (int i = 0; i < points.length; i++) {
      Point currentPoint = points[i];
      Point nextPoint;
      if ((i+1) == points.length) {
        nextPoint = points[i];
      } else {
        nextPoint = points[i+1];
      }

      // Calculate size of step to get from currentPoint to nextPoint
      // To increase smoothness use float/double
      int stepX = (nextPoint.getX() - currentPoint.getX()) / steps;
      int stepY = (nextPoint.getY() - currentPoint.getY()) / steps;

      for (int j = 0; j < steps; j++) {
        int nextPositionX = this.getNextPos(figura.getX(), nextPoint.getX(), stepX);
        int nextPositionY = this.getNextPos(figura.getY(), nextPoint.getY(), stepY);
        figura.moveTo(nextPositionX, nextPositionY);

        this.pintar(figura);
        Util.waitMilli(waitTime);
      }
    }
  }

  /**
   * Calculates the next position for pintarPath and avoids
   * overstepping. Since we are using ints, it is possible that
   * the step is rounded to a an higher value integer.
   * @param current
   * @param next
   * @param step
   * @return
   */
  public int getNextPos(int current, int next, int step) {
    int nextPosition = current + step;
    if (step > 0) {
      if (nextPosition > next) {
        nextPosition = next;
      }
    } else {
      if (nextPosition < next) {
        nextPosition = next;
      }
    }

    return nextPosition;
  }
}
```

Figura

```
import java.awt.*; // Import everything inside java.awt
import javax.swing.JFrame;

public abstract class Figura {
  private int x;
  private int y;
  private boolean relleno;
  private Color color;

  // Most comprehensive constructor
  Figura(int x, int y, boolean relleno, Color color) {
    this.setX(x);
    this.setY(y);
    this.setRelleno(relleno);
    this.setColor(color);
  }

  public void setX(int x) {
    // Possible upgrades, pair with Lienzo to avoid
    // setting a Figura outside canvas
    if (x >= 0) {
      this.x = x;
    }
  }

  public void setY(int y) {
    if (y >= 0) {
      this.y = y;
    }
  }

  public void moveTo(int x, int y) {
    this.setX(x);
    this.setY(y);
  }

  public void setRelleno(boolean relleno) {
    this.relleno = relleno;
  }

  public void setColor(Color color) {
    this.color = color;
  }

  public int getX() {
    return this.x;
  }
```

```java
  public int getY() {
    return this.y;
  }

  public boolean getRelleno() {
    return this.relleno;
  }

  // Method for sintatic bliss
  public boolean isRelleno() {
    return this.getRelleno();
  }

  public Color getColor() {
    return this.color;
  }

  public String getInfo() {
    return "\tPosición: (" + this.getX() + ", " + this.getY() + ")\n\tRelleno: " + this.getRelleno() + "\n\tColor: " + this.getColor()
  }

  public abstract void pintar(Graphics graphics);
}
```

Cuadrado

```java
import java.awt.*; // Probably could only import color

public class Cuadrado extends Figura {
  private int lado;

  Cuadrado(int x, int y, int lado, boolean relleno, Color color) {
    // Set common properties in figura
    super(x, y, relleno, color);
    this.setLado(lado);
  }

  public void setLado(int lado) {
    if (lado > 0) {
      this.lado = lado;
    }
  }

  public int getLado() {
    return this.lado;
  }

  @Override
  public String getInfo() {
    String info = "Figura: Cuadrado\n";
    String figureInfo = super.getInfo();
    info += figureInfo; // Join messages
    // Add custom properties
    info += "\n\tLado: " + this.getLado();
    return info;
  }

  @Override
  public void pintar(Graphics graphics) {
    graphics.setColor(this.getColor());
    graphics.drawRect(this.getX(), this.getY(), this.getLado(), this.getLado());
    if (this.isRelleno()) {
      graphics.fillRect(this.getX(), this.getY(), this.getLado(), this.getLado());
    }
  }
}
```

Circulo

```java
import java.awt.*; // Probably could only import color

public class Circulo extends Figura {
  private int radio;

  Circulo(int x, int y, int radio, boolean relleno, Color color) {
    super(x, y, relleno, color);
    this.setRadio(radio);
  }
  public void setRadio(int radio) {
    if (radio > 0) {
      this.radio = radio;
    }
  }

  public int getRadio() {
    return this.radio;
  }

  @Override
  public String getInfo() {
    String info = "Figura: Círculo\n";
    String figureInfo = super.getInfo();
    info += figureInfo; // Join messages
    // Add custom properties
    info += "\n\tRadio: " + this.getRadio();
    return info;
  }

  @Override
  public void pintar(Graphics graphics) {
    graphics.setColor(this.getColor());
    graphics.drawOval(this.getX(), this.getY(), this.getRadio(), this.getRadio());
    if (this.isRelleno()) {
      // radio + 1 to fully fill
      graphics.fillOval(this.getX(), this.getY(), this.getRadio() + 1, this.getRadio() + 1);
    }
  }
}
```

AppDibujo01

```
import java.awt.*; // Probably could only import color

public class AppDibujo01 {
  public static void main(String args[]) {
    Dibujo dibujo = new Dibujo();

    // Create two squares
    Cuadrado cuadrado1 = new Cuadrado(0,0,100,true,Color.GREEN);
    Cuadrado cuadrado2 = new Cuadrado(100,100,100,false,Color.BLUE);

    // Create two circles
    Circulo circulo1 = new Circulo(200,200,100,true,Color.ORANGE);
    Circulo circulo2 = new Circulo(300,300,100,true,Color.PINK);

    Figura[] figuras = {cuadrado1, circulo1, cuadrado2, circulo2};

    dibujo.pintar(figuras);
  }
}
```

AppDibujo02

```
import java.awt.Color;

public class AppDibujo02 {
  public static void main(String args[]) {
    Dibujo dibujo = new Dibujo();
    for (int i = 0; i < 10; i++) {
      boolean relleno = false;
      if (i % 3 == 0) {
        relleno = true;
        // Fill every third square
      }
      int x = 20 * i;
      int y = 20 * i;
      Cuadrado cuadrado = new Cuadrado(x, y, 100, relleno, Color.PINK);
      dibujo.pintar(cuadrado);
      Util.wait(1);
    }
  }
}
```

AppDibujo03

Pequeña animación usando AppDibujo02 como inspiración. El código no es muy bonito ni está bien estructurado pero, el resultado queda chulo.

```java
import java.awt.Color;

public class AppDibujo03 {
  public static void main(String args[]) {
    Dibujo dibujo = new Dibujo();
    int squares = 50;
    int waitTime = 40;
    while (true)  {
      for (int i = 0; i < squares; i++) {
        int pos = 5 * i + 80;

        Cuadrado cuadrado = new Cuadrado(pos, pos, 100, false, Color.PINK);
        dibujo.pintar(cuadrado);
        Util.waitMilli(waitTime);
      }

      for (int i = 0; i < squares; i++) {
        dibujo.removeFigura(i);
        dibujo.repintar();
        Util.waitMilli(waitTime);
      }

      dibujo.clearFiguras();
      dibujo.repintar();
      Util.waitMilli(waitTime);

      for (int i = 0; i < squares; i++) {
        int pos = (5 * (squares - 1) + 80) - (5 * i);
        Cuadrado cuadrado = new Cuadrado(pos, pos, 100, false, Color.PINK);
        dibujo.pintar(cuadrado);
        Util.waitMilli(waitTime);
      }

      for (int i = 0; i < squares; i++) {
        dibujo.removeFigura(i);
        dibujo.repintar();
        Util.waitMilli(waitTime);
      }

      dibujo.clearFiguras();
      dibujo.repintar();
      Util.waitMilli(waitTime);
    }

  }
}
```