Práctica 0

He añadido alguna funcionalidad más a las clases Util y Dibujo, por eso las he puesto también. Además hice una clase Point para AppCuadrado03.

Util

```
/** Clase de funcionalidad variada que proporciona una ayuda al alumno */
public class Util
{
  /**
    Detiene el programa el tiempo especificado
    @param segundos número de segundos a esperar
  */
  public void wait(int segundos)
  {
    try
    {
      Thread.sleep(segundos*1000);
    }
    catch(Exception e)
    {
      System.out.println(e);
    }
  }

  /**
   * Detiene el programa el tiempo especificado, en milisegundos
   * @param milliseconds
   */
  public void waitMilli(int milliseconds)
  {
    try
    {
      Thread.sleep(milliseconds);
    }
    catch(Exception e)
    {
      System.out.println(e);
    }
  }
}
```

Point

```
public class Point {

  int x;
  int y;

  Point(int x,int y) {
    this.setX(x);
    this.setY(y);
```

```
  }

  public void setX(int x) {
    this.x = x;
  }

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

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

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

  public void print() {
    System.out.println(this.getX() + ", " + this.getY());
  }
}
```

Dibujo

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

/**
  Facilita la representación gráfica de objetos creados por el alumno mediante una ventana gráfica y un lienzo
*/
public class Dibujo extends JFrame
{
  private Lienzo lienzo;

  int sizeX;
  int sizeY;

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

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

    // Create canvas
    lienzo = new Lienzo();
    lienzo.setSize(this.getSizeX(), this.getSizeY());
    this.add(lienzo);
    this.pack();
    this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    this.setVisible(true);
  }

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

  public void setSizeX(int sizeX) {
    if (sizeX > 0 && sizeX < 1920) {
      this.sizeX = sizeX;
    } else {
      this.sizeX = 800;
    }
  }

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

```
  }

  public void setSizeY(int sizeY) {
    if (sizeY > 0 && sizeY < 1080) {
      this.sizeY = sizeY;
    } else {
      this.sizeY = 600;
    }
  }

  public int getSizeY() {
    return this.sizeY;
  }
  /**
    Pinta el cuadrado recibido por el App y actualiza el lienzo (canvas)
    @param cuadrado cuadrado a pintar
  */
  public void pintar(Cuadrado cuadrado)
  {
    lienzo.pintar(cuadrado);
    lienzo.repaint();
  }

  /**
   * Draws the square in the middle of the canvas, regardless
   * of the size of the square and the size of the square
   * @param cuadrado
   */
  public void setInTheMiddle(Cuadrado cuadrado) {
    int sizeX = this.getSizeX();
    int sizeY = this.getSizeY();
    int lado = cuadrado.getLado();

    int x = (sizeX - lado) / 2;
    int y = (sizeY - lado) / 2;

    cuadrado.setX(x);
    cuadrado.setY(y);
  }

  /**
   * 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 cuadrado
   * @param points
   * @param steps
   * @param frameRate
   */
  public void pintarPath(Cuadrado cuadrado, Point[] points, int steps, int frameRate) {
    // Move to initial position
    cuadrado.moveTo(points[0].getX(), points[0].getY());

    // Waiting times
    int waitTime = steps / frameRate;
    Util util = new Util();

    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(cuadrado.getX(), nextPoint.getX(), stepX);
        int nextPositionY = this.getNextPos(cuadrado.getY(), nextPoint.getY(), stepY);
        cuadrado.moveTo(nextPositionX, nextPositionY);

        this.pintar(cuadrado);
        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;
  }

}
```

Cuadrado

```
public class Cuadrado {
  int x;
  int y;
  int lado;

  Cuadrado(int x, int y, int lado) {
    this.setX(x);
    this.setY(y);
    this.setLado(lado);
  }

  // If lado is not given, generate cuadrado of lado 1
  Cuadrado(int x, int y) {
    this(x, y, 1);
  }

  public void setX(int x) {
    this.x = x;
  }

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

  public void setLado(int lado) {
    if (lado > 0) {
      this.lado = lado;
    } else {
      this.lado = 1; // Defualt value
    }
  }
```

```
  // Wrapper for easy moving
  public void moveTo(int x, int y) {
    this.setX(x);
    this.setY(y);
  }

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

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

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

}
```

AppCuadrado01

```
public class AppDibujo01 {

  public static void main(String[] args) {
    // Create frame 600 by 600
    Dibujo dibujo = new Dibujo(600, 600);
    Util util = new Util();
    // Create the square
    Cuadrado cuadrado = new Cuadrado(100, 100, 200);
    dibujo.pintar(cuadrado);
    util.wait(1);

    Cuadrado cuadrado2 = new Cuadrado(400, 400, 100);
    dibujo.pintar(cuadrado2);
    util.wait(1);

    Cuadrado cuadrado3 = new Cuadrado(100, 400, 50);
    dibujo.pintar(cuadrado3);
    util.wait(1);
  }
}
```

AppCuadrado02

```
public class AppDibujo02 {
  public static void main(String[] args) {
    // Create frame 600 by 600
    Dibujo dibujo = new Dibujo(600, 600);
    Util util = new Util();
    // Create the square
    Cuadrado cuadrado = new Cuadrado(100, 100, 200);

    for (int i = 0; i < 2; i++) {
      cuadrado.moveTo(200*i,200*i);
      dibujo.pintar(cuadrado);
      util.wait(1);
    }

    cuadrado.setLado(500);
    dibujo.setInTheMiddle(cuadrado);
    dibujo.pintar(cuadrado);
    util.wait(1);
  }
}
```

AppCuadrado03

```
public class AppDibujo03 {
  /**
   * When calling the app, you have two options, pass no
   * args which will render a square following a defautl path
   * or you can enter pair of points, that the square will follow.
   * For example:
   * java AppDibujo03 0 0 400 0 400 400 0 400
   * Will draw a square which goes to all corners of the canvas
   * @param args
   */
  public static void main(String[] args) {
    // Create frame 600 by 600
    Dibujo dibujo = new Dibujo(600, 600);

    // Create the square
    Cuadrado cuadrado = new Cuadrado(100, 100, 200);

    if (args.length < 2 || args.length % 2 != 0) {
      doDefaultPath(cuadrado, dibujo);
    } else {
      int points = args.length / 2;
      Point[] path = new Point[points];

      for (int i = 0; i < points; i++) {
        int x = Integer.parseInt(args[i*2]);
        int y = Integer.parseInt(args[i*2 + 1]);

        Point newPoint = new Point(x,y);
        path[i] = newPoint;
      }

      dibujo.pintarPath(cuadrado, path, 50, 5);
    }

  }

  public static void doDefaultPath(Cuadrado cuadrado, Dibujo dibujo) {
    Point point1 = new Point(0,0);
    Point point2 = new Point(400, 0);
    Point point3 = new Point(400,400);
    Point point4 = new Point(0, 400);
    Point point5 = new Point(300, 300);

    Point[] path = new Point[]{point1, point2, point3, point4, point5};
    dibujo.pintarPath(cuadrado, path, 50, 5);
    dibujo.setInTheMiddle(cuadrado);
    dibujo.pintar(cuadrado);
  }
}
```