

Programación Orientada a Objetos

Introducción

CEIS

2019-01

Agenda

Iniciando

- Las tres P
- Investigación

POOB-Curso

- Descripción
- Prácticas, lenguajes y herramientas

Orientación a objetos

- Materia prima
- Clases y objetos
- Atributos
- Métodos
- Casa

Herramienta. BlueJ

- General
- Editar
- Compilar
- Ejecutar
- Documentar

Agenda

Iniciando

Las tres P

Investigación

POOB-Curso

Descripción

Prácticas, lenguajes y herramientas

Orientación a objetos

Materia prima

Clases y objetos

Atributos

Métodos

Casa

Herramienta. BlueJ

General

Editar

Compilar

Ejecutar

Documentar

Programa

Componentes

Calidad

Programa

Componentes

- ▶ Ejecutable
- ▶ Fuentes
- ▶ Manual de usuario
- ▶ Manual técnico

Calidad

- ▶ Corrección
- ▶ Extensibilidad
- ▶ Facilidad de Uso
- ▶ Eficiencia
- ▶ Portabilidad

Proceso

Etapas

Calidad

Proceso

Etapas

1. Requisitos
2. Análisis
3. Diseño
4. Construcción
5. Pruebas

Calidad

- ▶ Cronograma
- ▶ Alcance
- ▶ Presupuesto

Proceso

Etapas

1. Requisitos

¿ Qué necesita el cliente?

2. Análisis

¿ Qué vamos a hacer?

3. Diseño

¿ Cómo lo vamos a hacer?

4. Construcción

¡ Hacerlo!

5. Pruebas

¿ Lo hicimos bien?

Calidad

▶ Cronograma

▶ Alcance

▶ Presupuesto

Personas

Técnicos

- ▶ Requisitos
- ▶ Análisis
- ▶ Diseño
- ▶ Construcción
- ▶ Pruebas

Personas

Técnicos

- ▶ Requisitos
- ▶ Análisis
- ▶ Diseño
- ▶ Construcción
- ▶ Pruebas de unidad

Analista

Diseñador

¡ ARQUITECTO !

Programador

¡ EQUIPO DE CALIDAD !

Investigación

Lo ágil

Manifesto for Agile Software Development

We are uncovering better ways of developing software by doing it and helping others do it.
Through this work we have come to value:

Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

That is, while there is value in the items on the right, we value the items on the left more.

XP

Kent Beck

Mike Beedle
Arie van Bennekum
Alistair Cockburn
Ward Cunningham
Martin Fowler

James Grenning
Jim Highsmith
Andrew Hunt
Ron Jeffries
Jon Kern
Brian Marick

SOLID

Robert C. Martin

Steve Mellor
Ken Schwaber
Jeff Sutherland
Dave Thomas



The Rules and Practices of Extreme Programming.

Lessons Learned

Planning

- ◊ 2 User stories are written.
- ◊ 2 Release planning creates the schedule.
- ◊ 2 Make frequent small releases.
- ◊ 2 The Project Velocity is measured.
- ◊ 2 The project is divided into iterations.
- ◊ 2 Iteration planning starts each iteration.
- ◊ 2 Move people around.
- ◊ 2 A stand-up meeting starts each day.
- ◊ 2 Fix XP when it breaks.

Coding

- ✖ The customer is always available.
- ◊ 2 Code must be written to agreed standards.
- ◊ 2 Code the unit test first.
- ◊ 2 All production code is pair programmed.
- ◊ 2 Only one pair integrates code at a time.
- ◊ 2 Integrate often.
- ◊ 2 Use collective code ownership.
- ◊ 2 Leave optimization till last.
- ◊ 2 No constants.

Designing

- ◊ 2 Simplicity.
- ◊ 2 Choose a system metaphor.
- ◊ 2 Use CRC cards for design sessions.
- ◊ 2 Create spike solutions to reduce risk.
- ◊ 2 No functionality is added early.
- ◊ 2 Refactor whenever and wherever possible.

Testing

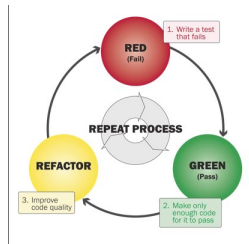
- ◊ 2 All code must have unit tests.
- ◊ 2 All code must pass all unit tests before it can be released.
- ◊ 2 When a bug is found tests are created.
- ◊ 2 Acceptance tests are run often and the score is published.

ExtremeProgramming.org home | XP Map | Email the webmaster

Copyright 1999-2004 by all rights reserved

Investigación

BDD



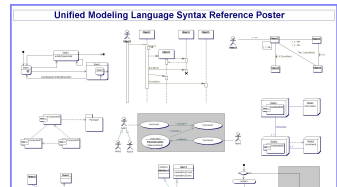
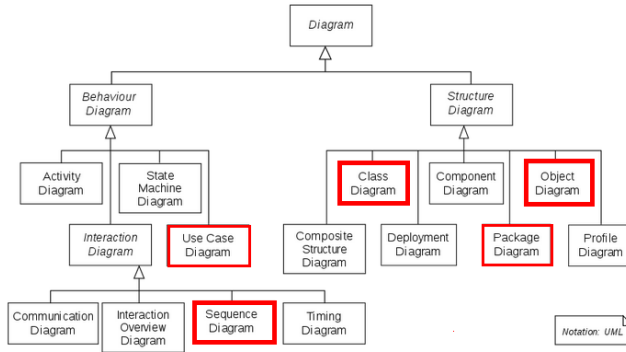
Proceso

MDD



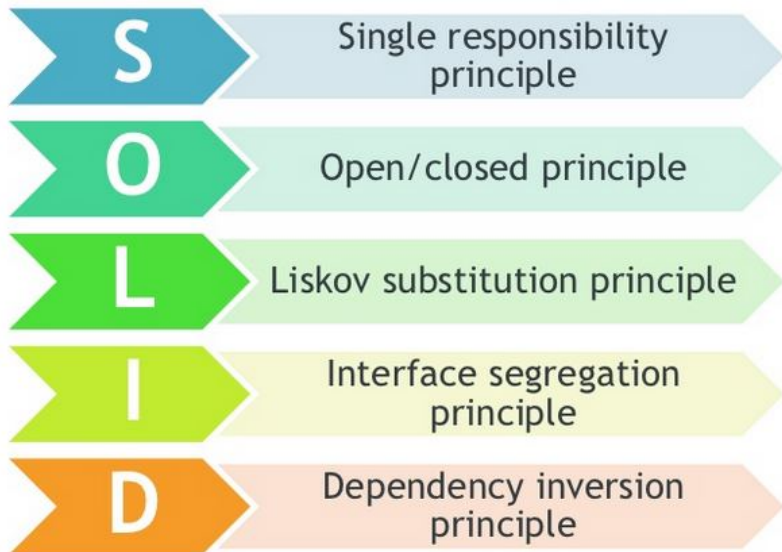
Investigación

UML



Investigación

SOLID



Investigación

Java

```
import com.lauchenauer.istockhelper.  
import com.lauchenauer.lib.ui.Vertic  
import com.lauchenauer.lib.util.Brow  
  
public class AboutDialog extends JDia  
    protected CardLayout mLayout;  
    protected JButton mCredits;  
    protected JPanel mMainPanel;  
  
    public AboutDialog(JFrame owner) {  
        super(owner);  
        setModal(true);  
        setUndecorated(true);  
        initUI();  
    }  
  
    protected void initUI() {  
        setSize(440, 600);  
        Container cont = getContentPane  
        JPanel p =
```


Agenda

Iniciando

Las tres P

Investigación

POOB-Curso

Descripción

Prácticas, lenguajes y herramientas

Orientación a objetos

Materia prima

Clases y objetos

Atributos

Métodos

Casa

Herramienta. BlueJ

General

Editar

Compilar

Ejecutar

Documentar

Objetivo

¿Cuál es el propósito?

Construir un producto software o **mejorar** uno existente.

¡Lograr que el software **funcione** y **evolucione** !

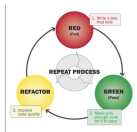
Objetivo

¿Cuál es el propósito?

Construir un producto software o **mejorar** uno existente.

¡Lograr que el software **funcione** y **evolucione** !

BDD



MDD



SOLID



XP

 **The Rules and Practices of Extreme Programming.** *Extreme Programming*

Planning

- User stories are written.
- Business planning creates the schedule.
- Make frequent small releases.
- The project is divided into sprints.
- Iteration planning starts each sprint.
- Status people around.
- A stand-up meeting starts each day.
- Stop when it breaks.

Coding

- The customer is **always** available.
- Code must be written to agreed standards.
- Code flow must be **test first**.
- All production code is **unit** tested.
- Only one pair **owns** code.
- All code is **simple**.
- The customer code **owns** the code.
- All code is **run** often.
- No **complexity**.

Designing

- Simplicity.
- Create a **working** solution.
- Use **UML** models for design.
- Create **simple** designs to reduce risk.
- No **flexibility** in **code**.
- **Refactor** whenever and wherever possible.

Testing

- All code must have **unit** tests.
- All code must pass all **unit** tests before it is released.
- When a **bug** is found, tests are created.
- **Acceptance** tests are run often and the score is published.

<http://www.programmingsays.com> (32-36p) [Download the information](http://www.programmingsays.com)

Copyright © 1999-2000 by the author. All rights reserved.

Metodología

► Clase

Teoría

Trabajo en clase

► Laboratorio

Semanas pares Viernes a.m.

Sustentación con el monitor

[Entrega final Mc]

► Proyecto

Inicial. [1ero y 2do tercio] Cuatro ciclos dos por tercio

Final. [3er tercio] Estructura + dos ciclos

[Entrega Ju]

Evaluación

- ▶ 50% Examen parcial

T1 [Semana 6 (Vi)], T2 [Semana 11 (Vi)] T3 [Proyecto]

- ▶ 15% Quices y trabajos en clase

- ▶ 15% Laboratorio

Maratón *HackerRank Java* BONO 3er tercio

Inicio: semana 1 Cierre: semana 16

Evaluación de verificación

- ▶ 20% Proyecto

Prácticas

XP



The Rules and Practices of Extreme Programming.

Lessons Learned

Planning

- ❖❖ User stories are written.
- ❖❖ Release planning creates the schedule.
- ❖❖ Make frequent small releases.
- ❖❖ The Project Velocity is measured.
- ❖❖ The project is divided into 1 iterations.
- ❖❖ Iteration planning starts each iteration.
- ❖❖ Move people around.
- ❖❖ A stand-up meeting starts each day.
- ❖❖ Fix XP when it breaks.

Designing

- ❖❖ Simplicity. 4
- ❖❖ Choose a system metaphor.
- ❖❖ Use CRC cards for design sessions.
- ❖❖ Create spike solutions to reduce risk.
- ❖❖ No functionality is added early.
- ❖❖ Refactor whenever and wherever possible. 6

Coding

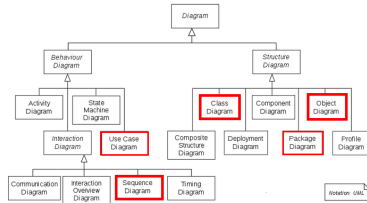
- ❖❖ The customer is always available.
- ❖❖ Code must be written to agreed standards. 4
- ❖❖ Code the unit test first. 2
- ❖❖ All production code is pair programmed. 1
- ❖❖ Only one pair integrates code at a time. 3
- ❖❖ Integrate often.
- ❖❖ Use collective code ownership. 3
- ❖❖ Leave optimization till last.
- ❖❖ No overtime.

Testing

- ❖❖ All code must have unit tests. 2
- ❖❖ All code must pass all unit tests before it can be released. 6
- ❖❖ When a bug is found tests are created. 5
- ❖❖ Acceptance tests are run often and the score is published. 5

ExtremeProgramming.org home | [XP Map](#) | [Email the webmaster](#)

Lenguajes



Herramientas

Herramientas

- ▶ **JDK Conjunto de herramientas de desarrollo**
- ▶ JUnit Herramienta de pruebas unitarias
- ▶ **BlueJ Ambiente de desarrollo**
- ▶ ECLIPSE Ambiente de desarrollo
- ▶ ASTAH Herramienta de modelado
- ▶ Trello Administración de proyectos
- ▶ GitHub Plataforma para alojar proyectos

Herramientas

JDK

There have been similar discontinuities in the way in which the **Java Development Kit (JDK)** has been referred to. The JDK is the software “bundle” used by developers to build Java applications and consisting of

- The Java Virtual Machine (JVM)
- The Java compiler (javac)
- The Java Archive (jar) utility
- The Java documentation (javadoc) utility

Moodle

[illegible]

Agenda

Iniciando

Las tres P

Investigación

POOB-Curso

Descripción

Prácticas, lenguajes y herramientas

Orientación a objetos

Materia prima

Clases y objetos

Atributos

Métodos

Casa

Herramienta. BlueJ

General

Editar

Compilar

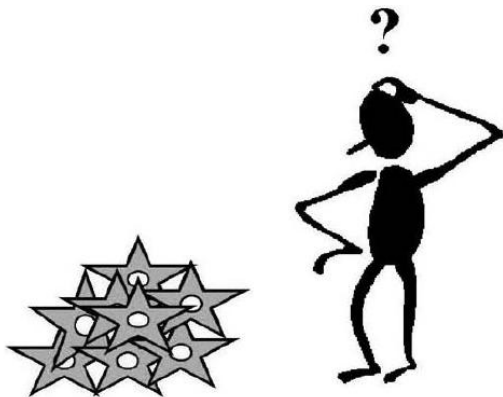
Ejecutar

Documentar

Historia



Historia



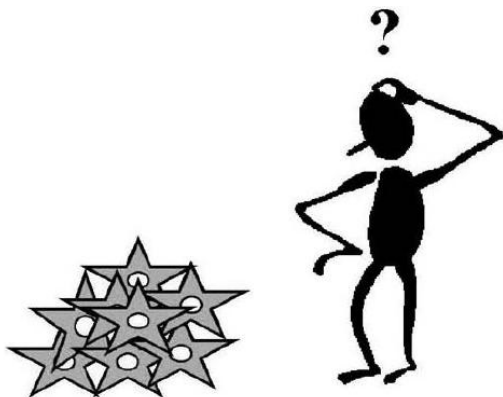
Historia



Historia



Clases y objetos



¿Qué son?

Objeto

Objeto (en el mundo real)

Objeto (en software)

Objeto

Objeto (en el mundo real)

Un **objeto** es algo mental o físico hacia el cual dirigimos nuestros sentimientos, pensamiento o acción.

Objeto (en software)

Objeto

Objeto (en el mundo real)

Un **objeto** es algo mental o físico hacia el cual dirigimos nuestros sentimientos, pensamiento o acción.

Objeto (en software)

Un **objeto** es un artefacto software que representa una abstracción de un objeto del mundo real por medio de su estado (datos) y comportamiento (funciones).

Clase

Clase (en el mundo real)

Clase (en software)

Clase

Clase (en el mundo real)

Una **clase** es una abstracción que describe todas las características comunes de todos los objetos de un grupo similar de objetos.

Clase (en software)

Objetos

Objeto (en software)

Un **objeto** es un artefacto software que representa una abstracción de un objeto del mundo real por medio de su estado (datos) y comportamiento (funciones).

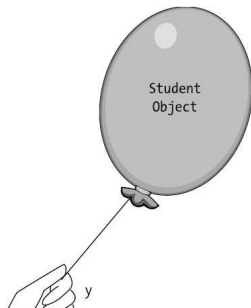
Objetos

Objeto (en software)

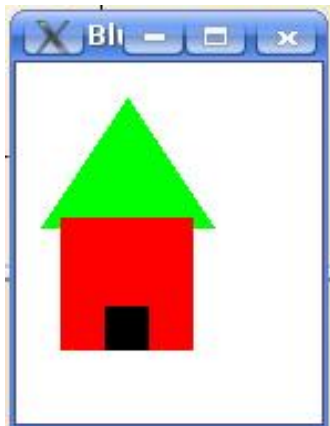
Un **objeto** es un artefacto software que representa una abstracción de un objeto del mundo real por medio de su estado (datos) y comportamiento (funciones).

Objeto

```
y = new Student();
```

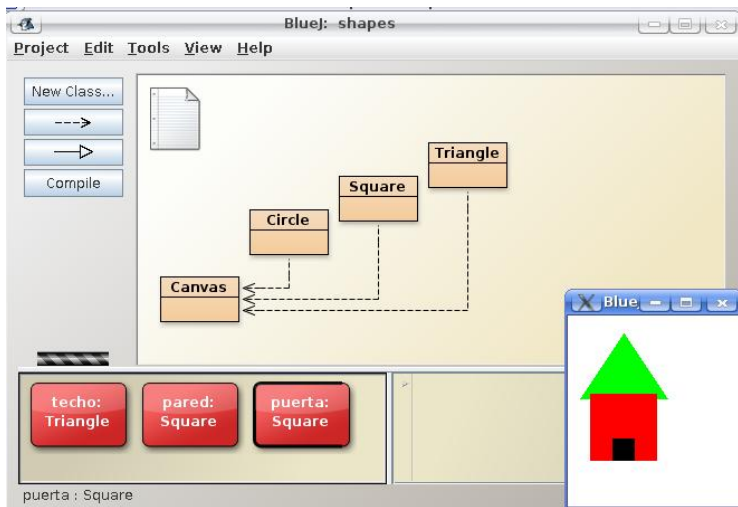


Clases y objetos



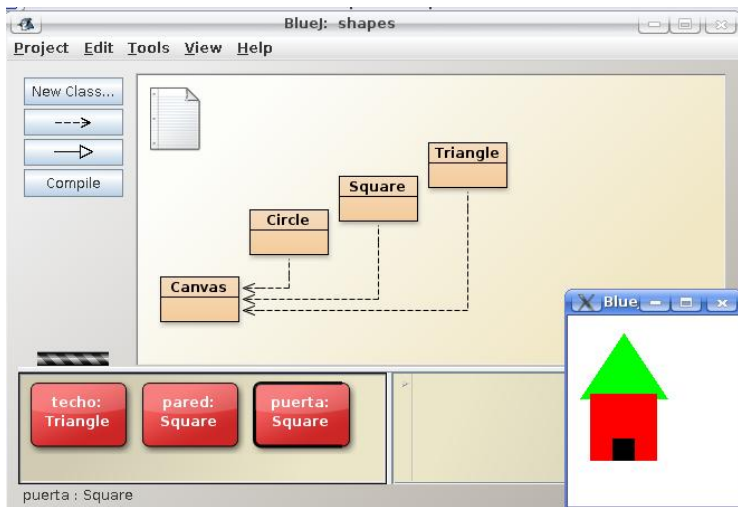
¿Clases? ¿Objetos?

Clases y objetos



¿Clases? ¿Objetos?

Clases y objetos



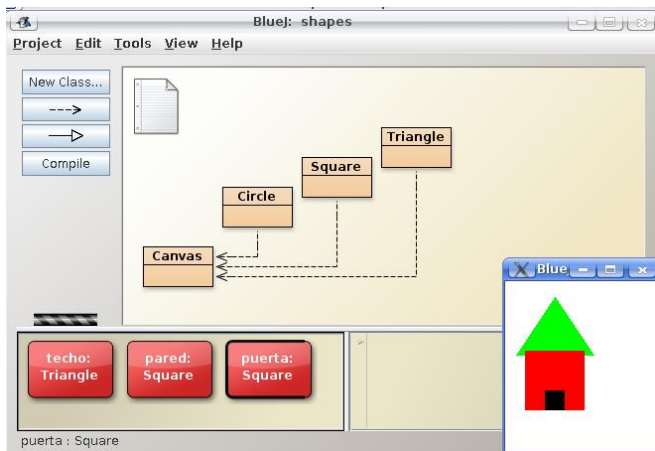
Clase cuadrado ¿Qué datos debe tener?

Clases y objetos

```
public class Square {  
    private int size;  
    private int xPosition;  
    private int yPosition;  
    private String color;  
    private boolean isVisible;  
  
    ...  
}
```

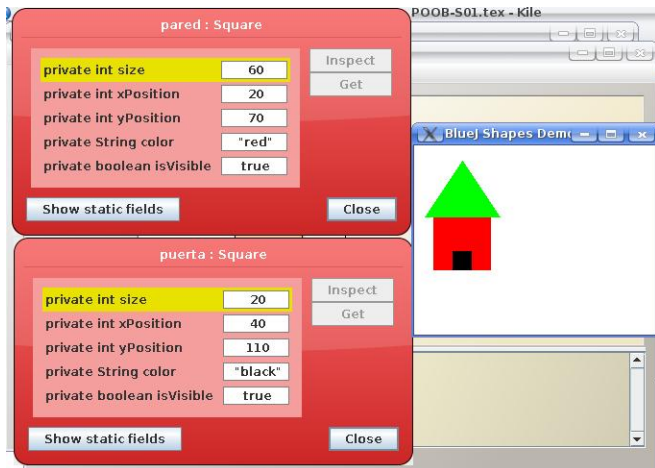
Clase cuadrado ¿Atributos?

Clases y objetos



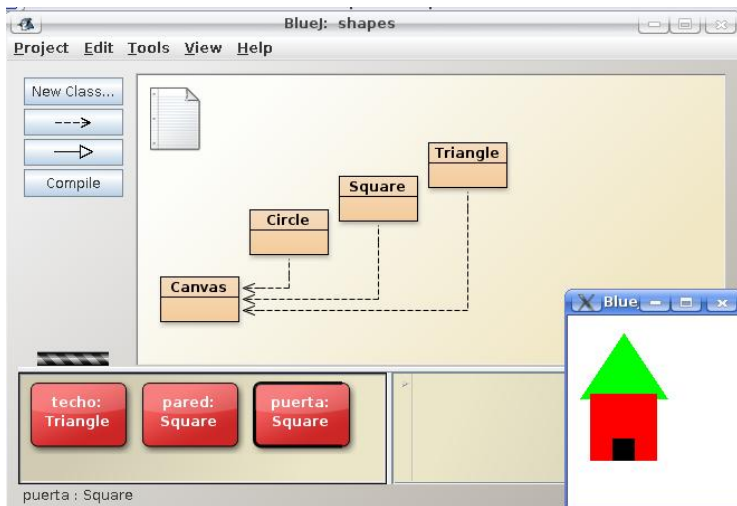
Objetos cuadrados ¿Valores de atributos?

Clases y objetos



Objetos cuadrados ¿Valores de atributos?

Clases y objetos



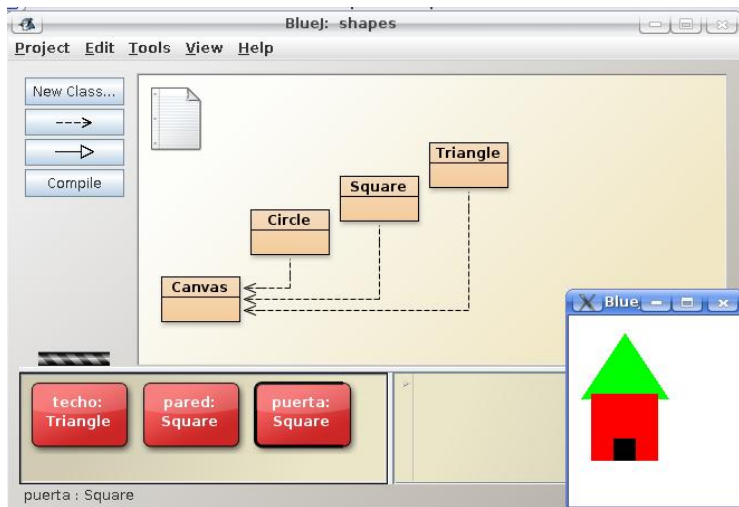
¿Método para crear un cuadrado?

Clases y objetos

```
public class Square {  
    private int size;  
    private int xPosition;  
    private int yPosition;  
    private String color;  
    private boolean isVisible;  
  
    /**  
     * Create a new square at default position with default color.  
     */  
    public Square() {  
        size = 30;  
        xPosition = 60;  
        yPosition = 50;  
        color = "red";  
        isVisible = false;  
    }  
    ...  
}
```

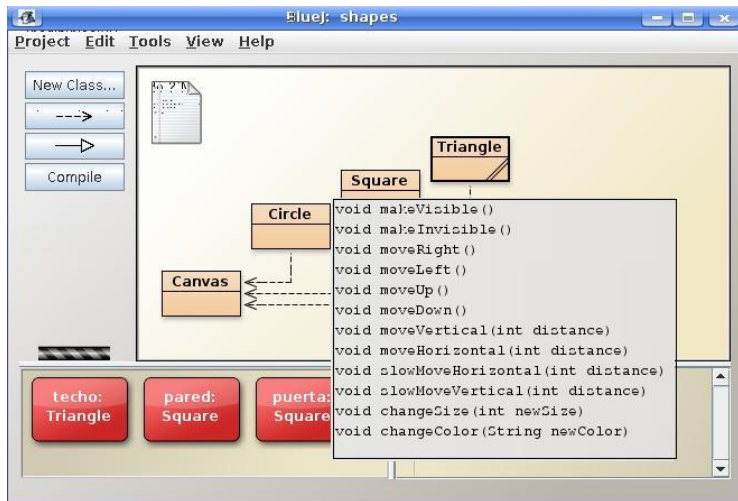
¿Método para crear un cuadrado?

Clases y objetos



Clase cuadrado ¿Qué debe permitir hacer?

Clases y objetos



Objetos cuadrados ¿Métodos?

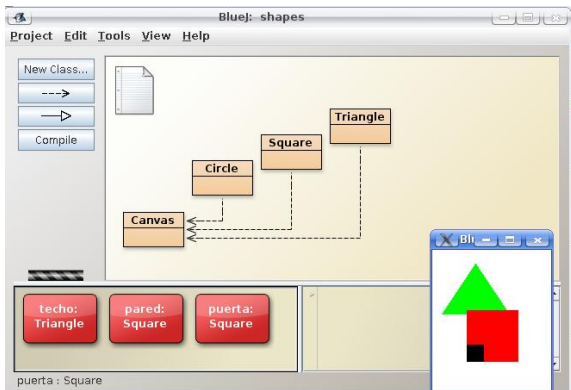
Clases y objetos

```
public class Square {  
    private int size;  
    private int xPosition;  
    private int yPosition;  
    private String color;  
    private boolean isVisible;  
  
    ...  
  
    /**  
     * Move the square horizontally by 'distance' pixels.  
     */  
    public void moveHorizontal(int distance) {  
        erase();  
        xPosition += distance;  
        draw();  
    }  
}
```

Clases y objetos

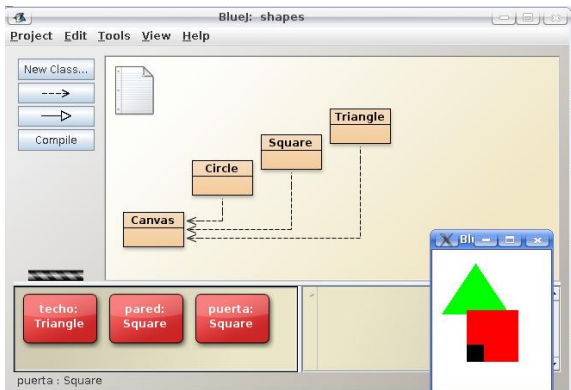
```
public class Square {  
    private int size;  
    private int xPosition;  
    private int yPosition;  
    private String color;  
    private boolean isVisible;  
    ...  
  
    /**  
     * Slowly move the square horizontally by 'distance' pixels.  
     */  
    public void slowMoveHorizontal(int distance) {  
        int delta;  
  
        if(distance < 0) {  
            delta = -1;  
            distance = -distance;  
        } else {  
            delta = 1;  
        }  
  
        for(int i = 0; i < distance; i++) {  
            xPosition += delta;  
            draw();  
        }  
    }  
    ...  
}
```

Clases y objetos



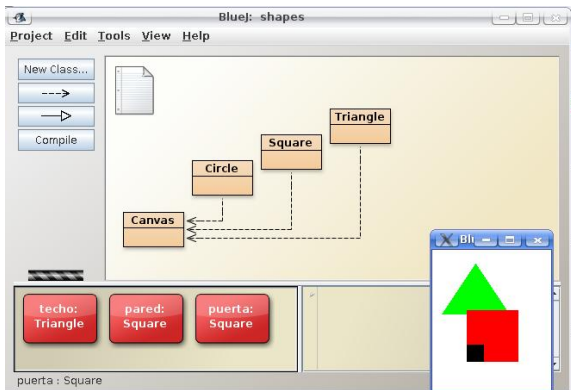
¿Qué pasó?

Clases y objetos



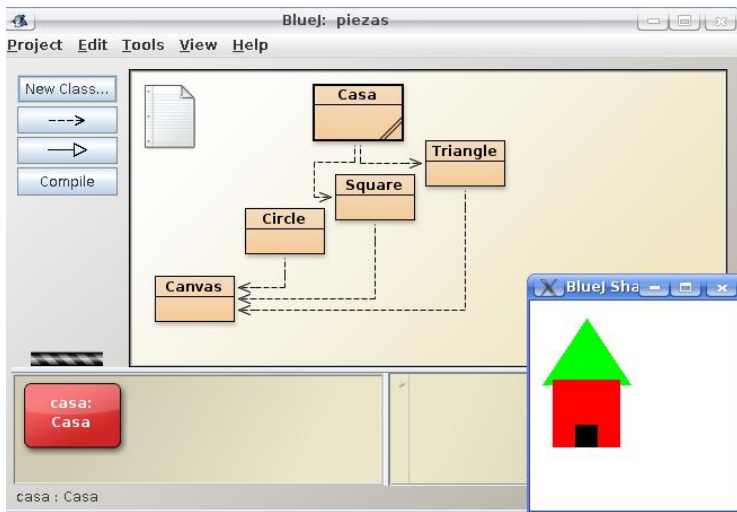
¿Quién no existe? Aunque la vemos ...

Clases y objetos



¿Qué hacemos?

Clases y objetos



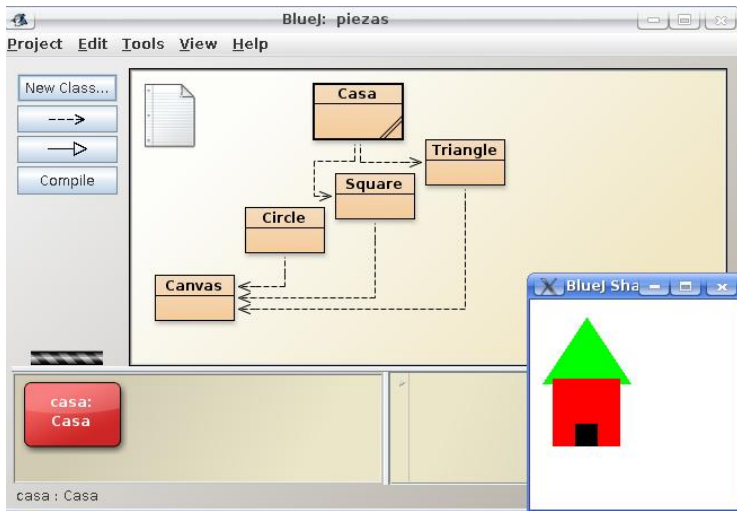
¿Atributos?

Clases y objetos

```
public class Casa {  
    private Triangle techo;  
    private Square pared;  
    private Square puerta;
```

Ingeniería reversa: ¿Diseño?

Clases y objetos



¿Creador? (No considerar tamaño ni ubicación)

Clases y objetos

```
public class Casa {  
  
    private Triangle techo;  
    private Square pared;  
    private Square puerta;  
  
    public Casa() {  
        techo=new Triangle();  
        techo.changeSize(60,80);  
        pared=new Square();  
        pared.changeSize(60);  
        pared.moveDown();  
        pared.moveLeft();  
        pared.moveLeft();  
        puerta=new Square();  
        puerta.changeSize(20);  
        puerta.changeColor("black");  
        puerta.moveDown();  
        puerta.moveDown();  
        puerta.moveDown();  
        puerta.moveLeft();  
    }  
}
```

Clases y objetos

BlueJ: piezas

Project Edit Tools View Help

New Class...
--->
->
Compile

```
classDiagram
    class Casa
    class Triangle
    class Square
    class Circle
    Casa <|-- Triangle
    Casa <|-- Square
    Square <|-- Circle
```

```
void makeVisible()
void makeInvisible()
void moveRight()
void moveLeft()
void moveUp()
void moveDown()
void moveVertical(int distance)
void moveHorizontal(int distance)
void slowMoveHorizontal(int distance)
void slowMoveVertical(int distance)
void changeSize(int newSize)
void changeColor(String newColor)
```

casa:
Casa

casa : Casa

BlueJ Sha

¿Hacer visible la casa?

Clases y objetos

```
public void makeVisible(){  
    techo.makeVisible();  
    pared.makeVisible();  
    puerta.makeVisible();  
}  
}
```

Ingeniería reversa: ¿Diseño?

Clases y objetos

BlueJ: piezas

Project Edit Tools View Help

New Class...
-->
->
Compile

```
classDiagram
    class Casa
    class Triangle
    class Square
    class Circle
    Casa <|-- Triangle
    Casa <|-- Square
    Circle --> Square
```

void makeVisible()
void makeInvisible()
void moveRight()
void moveLeft()
void moveUp()
void moveDown()
void moveVertical(int distance)
void moveHorizontal(int distance)
void slowMoveHorizontal(int distance)
void slowMoveVertical(int distance)
void changeSize(int newSize)
void changeColor(String newColor)

casas: Casa

casas : Casa

BlueJ Sha

¿Mover despacio la casa? Diseño - Código

Agenda

Iniciando

Las tres P

Investigación

POOB-Curso

Descripción

Prácticas, lenguajes y herramientas

Orientación a objetos

Materia prima

Clases y objetos

Atributos

Métodos

Casa

Herramienta. BlueJ

General

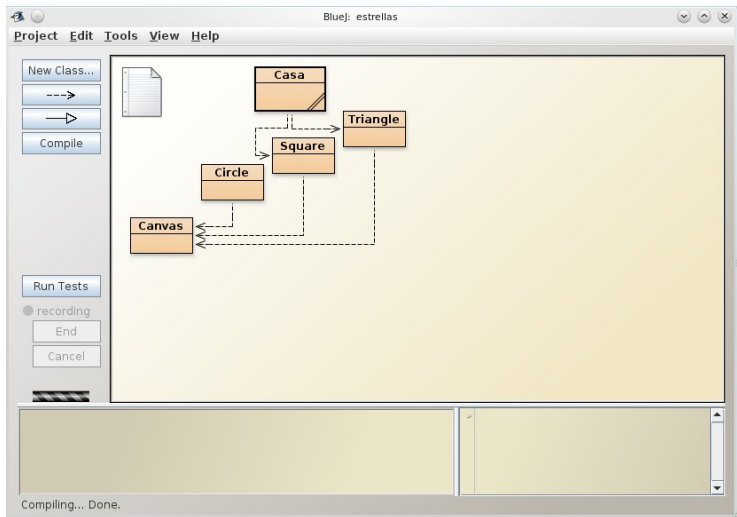
Editar

Compilar

Ejecutar

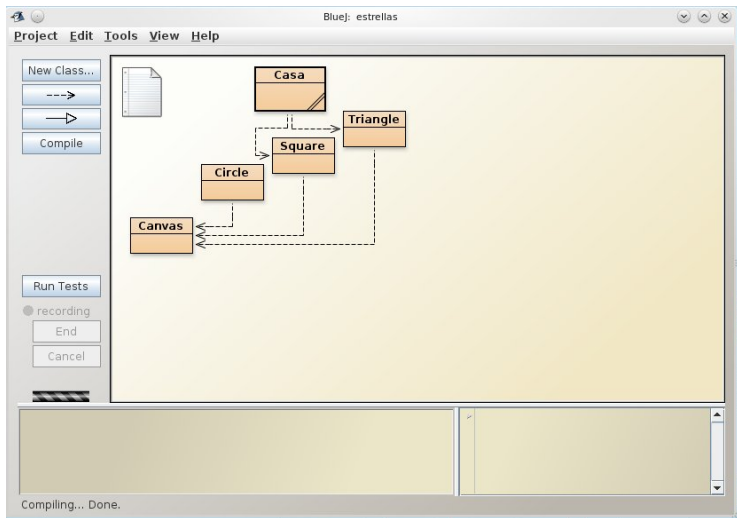
Documentar

General



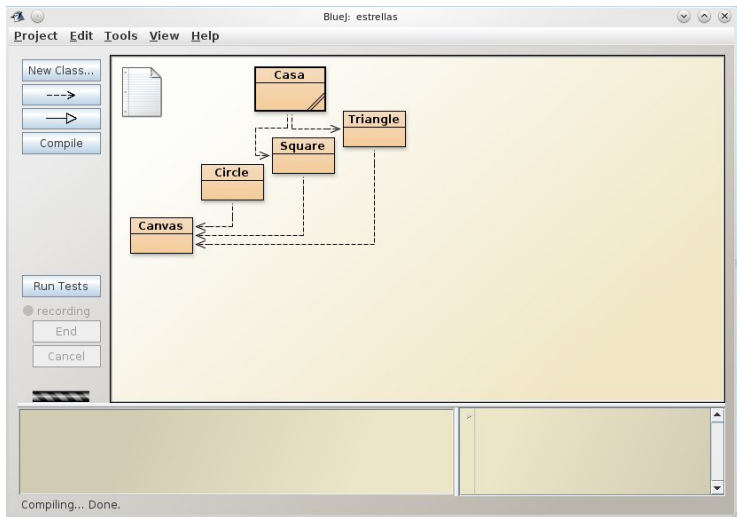
¿Qué debería permitir?

General



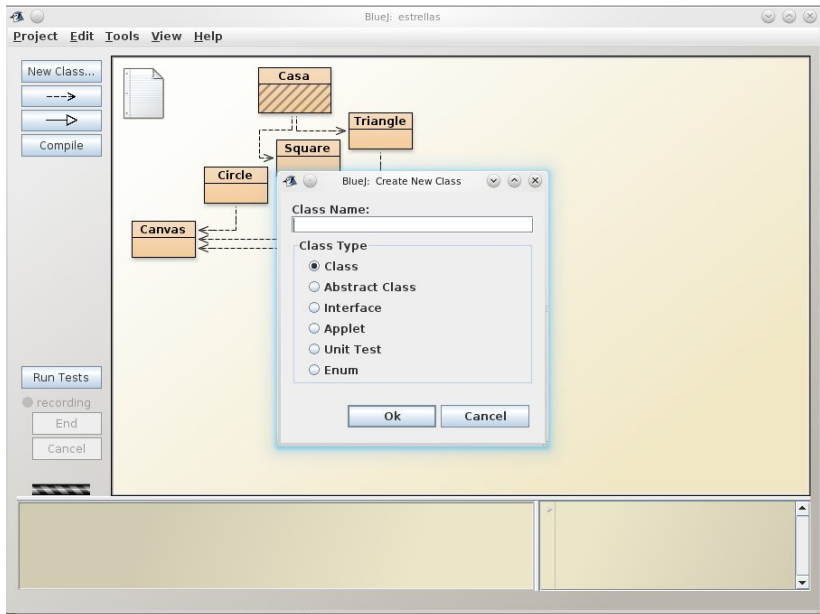
¿Qué usa?

General



¿En qué está desarrollado?

Editor



Editar

```
import java.awt.*;
import java.awt.geom.*;

/**
 * A circle that can be manipulated and that draws itself on a canvas.
 *
 * @author Michael Kolling and David J. Barnes
 * @version 1.0 (15 July 2000)
 */
public class Circle {
    public static final double PI=3.1416;

    private int diameter;
    private int xPosition;
    private int yPosition;
    private String color;
    private boolean isVisible;

    /**
     * Create a new circle at default position with default color.
     */
    public Circle() {
        diameter = 30;
        xPosition = 20;
        yPosition = 60;
        color = "blue";
        isVisible = false;
    }

    /**
     * Make this circle visible. If it was already visible, do nothing.
     */
    public void makeVisible() {
        isVisible = true;
        draw();
    }
}
```

¿Cómo se llama la clase?

Editor

```
import java.awt.*;
import java.awt.geom.*;

/**
 * A circle that can be manipulated and that draws itself on a canvas.
 *
 * @author Michael Kolling and David J. Barnes
 * @version 1.0 (15 July 2000)
 */
public class Circle {
    public static final double PI=3.1416;

    private int diameter;
    private int xPosition;
    private int yPosition;
    private String color;
    private boolean isVisible;

    /**
     * Create a new circle at default position with default color.
     */
    public Circle() {
        diameter = 30;
        xPosition = 20;
        yPosition = 60;
        color = "blue";
        isVisible = false;
    }

    /**
     * Make this circle visible. If it was already visible, do nothing.
     */
    public void makeVisible() {
        isVisible = true;
        draw();
    }
}
```

¿Cuántos atributos tiene?

Editor

```
import java.awt.*;
import java.awt.geom.*;

/**
 * A circle that can be manipulated and that draws itself on a canvas.
 *
 * @author Michael Kolling and David J. Barnes
 * @version 1.0 (15 July 2000)
 */
public class Circle {
    public static final double PI=3.1416;

    private int diameter;
    private int xPosition;
    private int yPosition;
    private String color;
    private boolean isVisible;

    /**
     * Create a new circle at default position with default color.
     */
    public Circle() {
        diameter = 30;
        xPosition = 20;
        yPosition = 60;
        color = "blue";
        isVisible = false;
    }

    /**
     * Make this circle visible. If it was already visible, do nothing.
     */
    public void makeVisible() {
        isVisible = true;
        draw();
    }
}
```

¿Qué diferencia existe entre PI y diameter?

Editor

```
import java.awt.*;
import java.awt.geom.*;

/**
 * A circle that can be manipulated and that draws itself on a canvas.
 *
 * @author Michael Kolling and David J. Barnes
 * @version 1.0 (15 July 2000)
 */
public class Circle {
    public static final double PI=3.1416;

    private int diameter;
    private int xPosition;
    private int yPosition;
    private String color;
    private boolean isVisible;

    /**
     * Create a new circle at default position with default color.
     */
    public Circle() {
        diameter = 30;
        xPosition = 20;
        yPosition = 60;
        color = "blue";
        isVisible = false;
    }

    /**
     * Make this circle visible. If it was already visible, do nothing.
     */
    public void makeVisible() {
        isVisible = true;
        draw();
    }
}
```

¿Qué otras clases se usan?

Editor

```
import java.awt.*;
import java.awt.geom.*;

/**
 * A circle that can be manipulated and that draws itself on a canvas.
 *
 * @author Michael Kolling and David J. Barnes
 * @version 1.0 (15 July 2000)
 */
public class Circle {
    public static final double PI=3.1416;

    private int diameter;
    private int xPosition;
    private int yPosition;
    private String color;
    private boolean isVisible;

    /**
     * Create a new circle at default position with default color.
     */
    public Circle() {
        diameter = 30;
        xPosition = 20;
        yPosition = 60;
        color = "blue";
        isVisible = false;
    }

    /**
     * Make this circle visible. If it was already visible, do nothing.
     */
    public void makeVisible() {
        isVisible = true;
        draw();
    }
}
```

¿Cuántos métodos vemos?

Editor

```
import java.awt.*;
import java.awt.geom.*;

/**
 * A circle that can be manipulated and that draws itself on a canvas.
 *
 * @author Michael Kolling and David J. Barnes
 * @version 1.0 (15 July 2000)
 */
public class Circle {
    public static final double PI=3.1416;

    private int diameter;
    private int xPosition;
    private int yPosition;
    private String color;
    private boolean isVisible;

    /**
     * Create a new circle at default position with default color.
     */
    public Circle() {
        diameter = 30;
        xPosition = 20;
        yPosition = 60;
        color = "blue";
        isVisible = false;
    }

    /**
     * Make this circle visible. If it was already visible, do nothing.
     */
    public void makeVisible() {
        isVisible = true;
        draw();
    }
}
```

¿Qué diferencia existe entre Circle y makeVisible?

Editor

```
import java.awt.*;
import java.awt.geom.*;

/**
 * A circle that can be manipulated and that draws itself on a canvas.
 *
 * @author Michael Kolling and David J. Barnes
 * @version 1.0 (15 July 2000)
 */
public class Circle {
    public static final double PI=3.1416;

    private int diameter;
    private int xPosition;
    private int yPosition;
    private String color;
    private boolean isVisible;

    /**
     * Create a new circle at default position with default color.
     */
    public Circle() {
        diameter = 30;
        xPosition = 20;
        yPosition = 60;
        color = "blue";
        isVisible = false;
    }

    /**
     * Make this circle visible. If it was already visible, do nothing.
     */
    public void makeVisible() {
        isVisible = true;
        draw();
    }
}
```

¿Cómo se crea un círculo? (Llámelo point)

Editar

```
import java.awt.*;
import java.awt.geom.*;

/**
 * A circle that can be manipulated and that draws itself on a canvas.
 *
 * @author Michael Kolling and David J. Barnes
 * @version 1.0 (15 July 2000)
 */
public class Circle {
    public static final double PI=3.1416;

    private int diameter;
    private int xPosition;
    private int yPosition;
    private String color;
    private boolean isVisible;

    /**
     * Create a new circle at default position with default color.
     */
    public Circle() {
        diameter = 30;
        xPosition = 20;
        yPosition = 60;
        color = "blue";
        isVisible = false;
    }

    /**
     * Make this circle visible. If it was already visible, do nothing.
     */
    public void makeVisible() {
        isVisible = true;
        draw();
    }
}
```

¿Cómo se hace visible el círculo anterior?

Editar

```
import java.awt.*;
import java.awt.geom.*;

/**
 * A circle that can be manipulated and that draws itself on a canvas.
 *
 * @author Michael Kolling and David J. Barnes
 * @version 1.0 (15 July 2000)
 */
public class Circle {
    public static final double PI=3.1416;

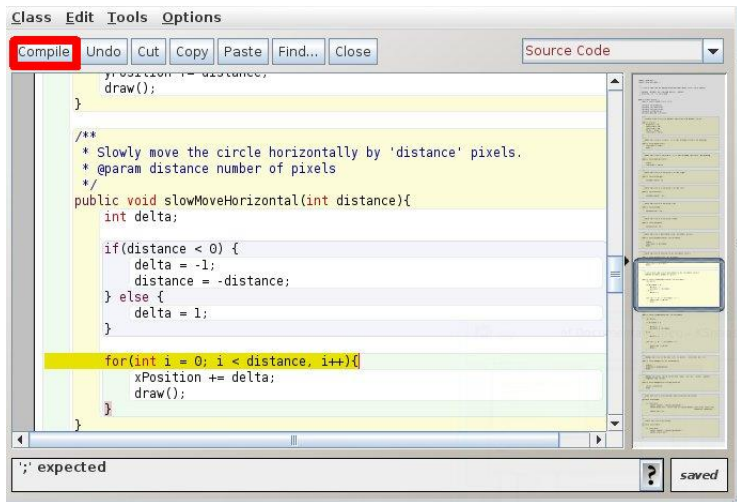
    private int diameter;
    private int xPosition;
    private int yPosition;
    private String color;
    private boolean isVisible;

    /**
     * Create a new circle at default position with default color.
     */
    public Circle() {
        diameter = 30;
        xPosition = 20;
        yPosition = 60;
        color = "blue";
        isVisible = false;
    }

    /**
     * Make this circle visible. If it was already visible, do nothing.
     */
    public void makeVisible() {
        isVisible = true;
        draw();
    }
}
```

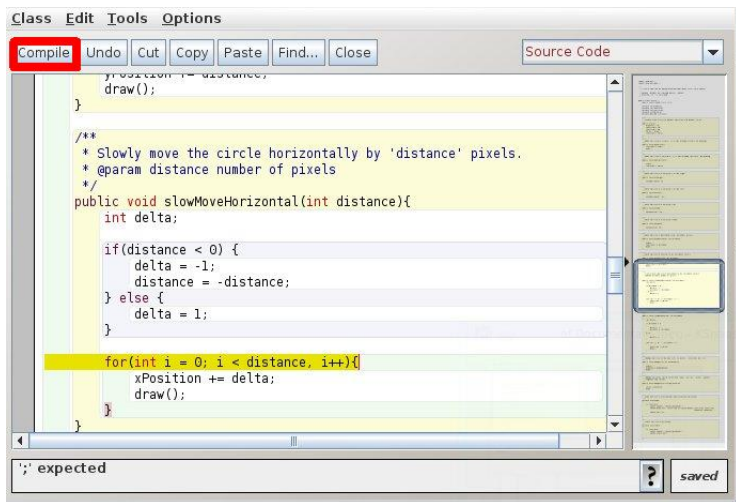
¿Cuál es el nombre del archivo?

Compilar



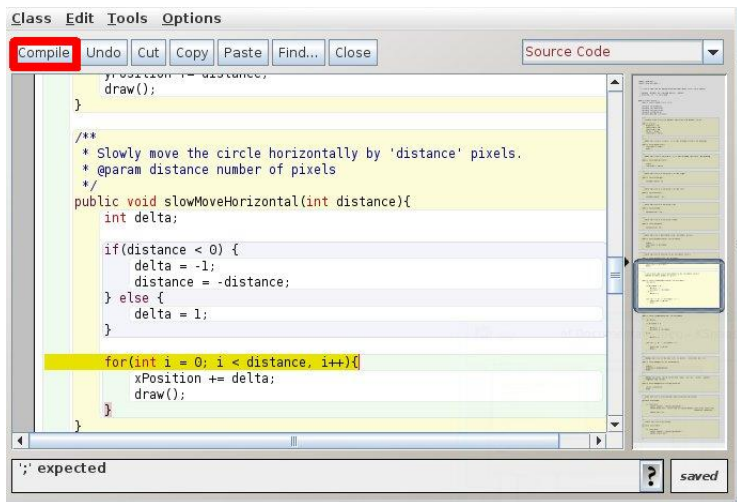
¿Cuál es la herramienta básica JDK?

Compilar



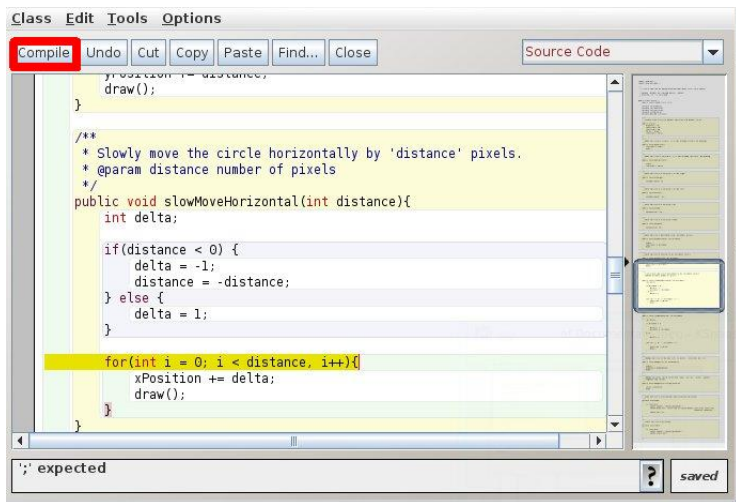
¿Qué error tiene el código?

Compilar



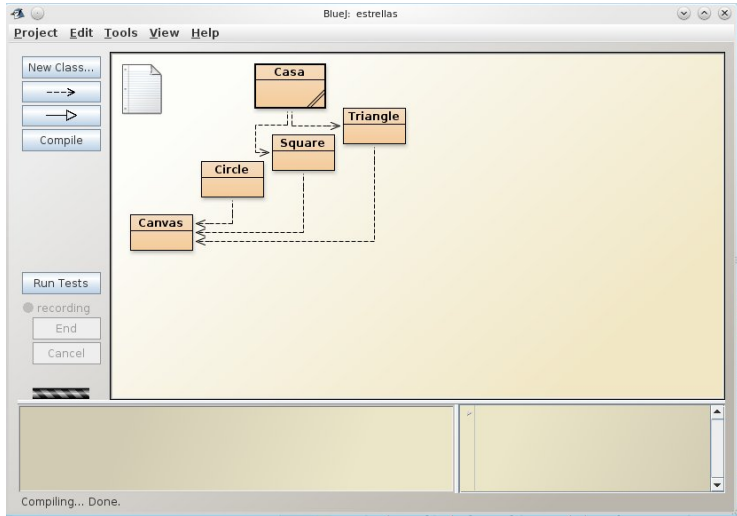
¿Qué hace el método?

Compiler



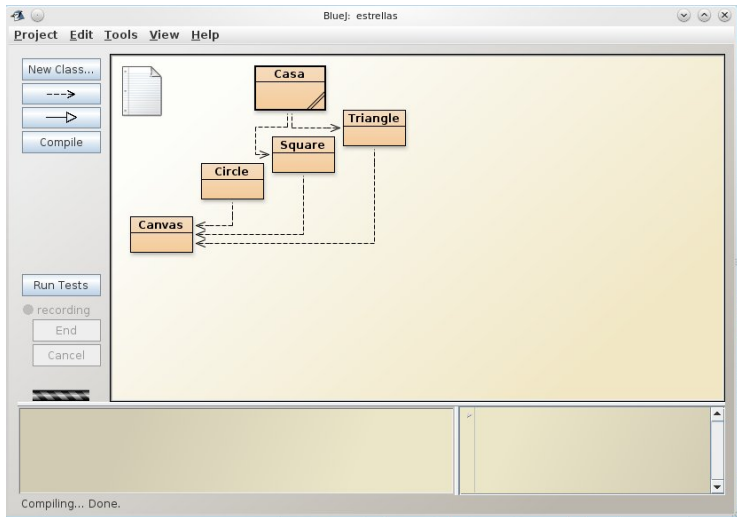
¿Cómo lo hace?

Ejecutar



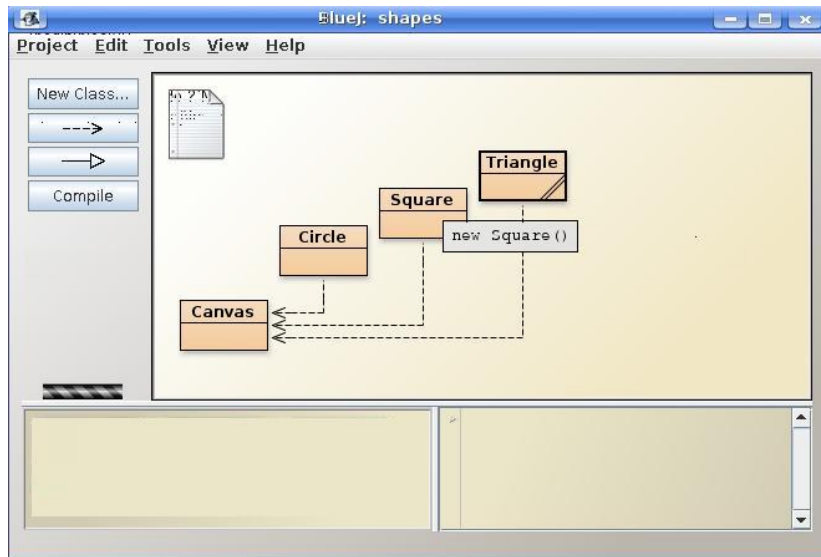
¿Cuál es la herramienta básica JDK?

Ejecutar



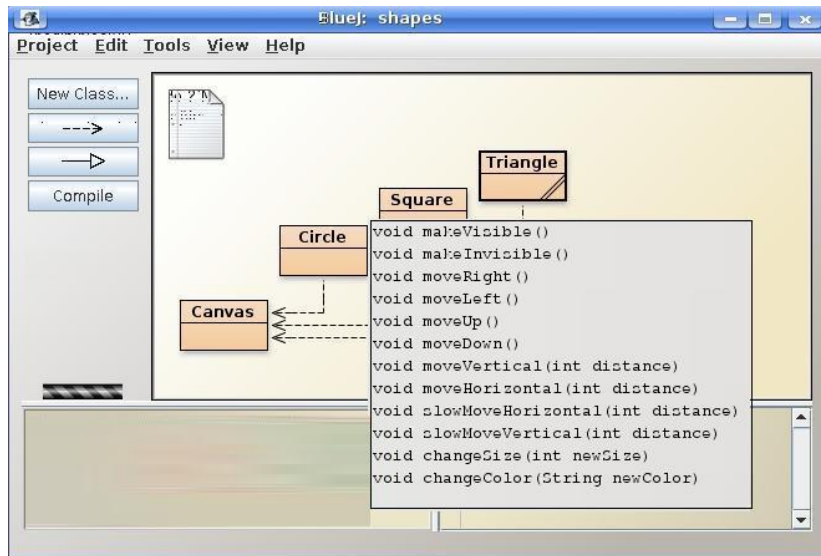
¿A quiénes podemos ejecutar?

Ejecutar



En Square, ¿Cuántos métodos de clase?

Ejecutar



En Square, ¿Cuántos métodos de objeto?

Ejecutar

```
import java.awt.*;
import java.awt.geom.*;

/**
 * A circle that can be manipulated and that draws itself on a canvas.
 *
 * @author Michael Kolling and David J. Barnes
 * @version 1.0. (15 July 2000)
 */
public class Circle{

    public static final double PI=3.1416;

    private int diameter;
    private int xPosition;
    private int yPosition;
    private String color;
    private boolean isVisible;

    /**
     * Create a new circle at default position with default color.
     */
    public Circle(){
        diameter = 30;
        xPosition = 20;
        yPosition = 15;
        color = "blue";
        isVisible = false;
    }
}
```

En Circle, ¿Cuáles atributos de clase?

Ejecutar

The screenshot shows the BlueJ IDE interface. The main window displays a class hierarchy diagram with the following classes and relationships:

- Casa** (base class)
- Circle** (subclass of Casa)
- Square** (subclass of Casa)
- Triangle** (subclass of Casa)
- Canvas** (class with associations to Circle, Square, and Triangle)

A red inspection window titled "bomba : Circle" is open, showing the following attributes and values:

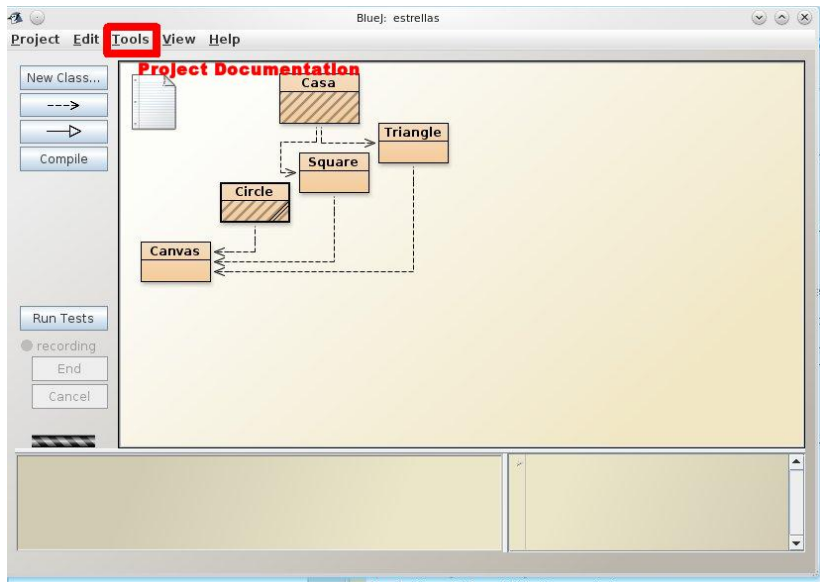
Attribute	Value
private int diameter	30
private int xPosition	20
private int yPosition	60
private String color	"blue"
private boolean isVisible	false

The window also includes buttons for "Inspect", "Get", "Show static fields", and "Close".

At the bottom left, a red button labeled "bomba: Circle" is visible, and the status bar shows "bomba : Circle".

En Circle, ¿Cuáles atributos de objeto?

Documentar



¿Cuál es la herramienta básica JDK?

Documentar

All Classes

- [Canvas](#)
- [Casa](#)
- [Circle](#)
- [Square](#)
- [Triangle](#)

Class Circle

[java.lang.Object](#)
└ **circle**

```
public class Circle
extends Object
```

A circle that can be manipulated and that draws itself on a canvas.

Version:
1.0 (15 July 2000)

Author:
Michael Kolling and David J. Barnes

Field Summary

static double	PI
---------------	--------------------

Constructor Summary

Circle()	Create a new circle at default position with default color.
--------------------------	-------------------------------------------------------------

Method Summary

void	changeColor(String newColor)	Change the color
------	----------------------------------------------	------------------

¿Qué se ve? ¿Qué no se vio?

Documentar

```
import java.awt.*;
import java.awt.geom.*;

/**
 * A circle that can be manipulated and that draws itself on a canvas.
 *
 * @author Michael Kolling and David J. Barnes
 * @version 1.0 (15 July 2000)
 */
```

```
public class Circle {
    public static final double PI=3.1416;

    private int diameter;
    private int xPosition;
    private int yPosition;
    private String color;
    private boolean isVisible;
```

```
    /**
     * Create a new circle at default position with default color.
     */
```

```
    public Circle() {
        diameter = 30;
        xPosition = 20;
        yPosition = 60;
        color = "blue";
        isVisible = false;
    }
```

```
    /**
     * Make this circle visible. If it was already visible, do nothing.
     */
    public void makeVisible() {
        isVisible = true;
        draw();
    }
```