

Programación Orientada a Objetos

Introducción

CIS

2024-1

Agenda

Iniciando

Las 3 P

Investigación: Guías + Lenguajes

POOB-Curso

Descripción

Prácticas, lenguajes y herramientas

Orientación a objetos

Materia prima

Clases y objetos

Atributos y métodos

Casa

Herramienta. BlueJ

General

Editar+Compilar+Ejecutar+Documentar

Agenda

Iniciando

Las 3 P

Investigación: Guías + Lenguajes

POOB-Curso

Descripción

Prácticas, lenguajes y herramientas

Orientación a objetos

Materia prima

Clases y objetos

Atributos y métodos

Casa

Herramienta. BlueJ

General

Editar+Compilar+Ejecutar+Documentar

Programa

Componentes

Calidad

¿?

- ▶ ¿Cuáles son los componentes de un programa?
- ▶ ¿Cuáles son los criterios de calidad de un programa?

Programa

Componentes

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

Calidad

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

- ▶ Portabilidad

Proceso

Etapas

Calidad

¿?

- ▶ ¿Cuáles son las fases de un proceso de desarrollo?
- ▶ ¿Cuáles son los criterios de calidad de un proceso de desarrollo?

Proceso

Etapas

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

Calidad

- ▶ Cronograma
- ▶ Alcance
- ▶ Presupuesto

Proceso

Etapas

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

¿ Qué necesita el cliente?

¿ Qué vamos a hacer?

¿ Cómo lo vamos a hacer?

¡ Hacerlo!

¿ Lo hicimos bien?

Calidad

- ▶ Cronograma
- ▶ Alcance
- ▶ Presupuesto

Personas

Técnicos

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

¿?

- ▶ ¿Cuáles son los roles de los desarrolladores?
- ▶ ¿Cuáles son los más reconocidos actualmente?

Personas

Técnicos

- ▶ Requisitos

- ▶ Análisis

Analista

- ▶ Diseño

Diseñador

¡ ARQUITECTO !

- ▶ Construcción

- ▶ Pruebas de unidad

Programador

¡ EQUIPO DE CALIDAD !

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:

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 of Extreme Programming

Planning

- User stories are written.
- Release planning creates the release schedule.
- Make frequent small releases.
- The project is divided into iterations.
- Iteration planning starts each iteration.

Managing

- Give the team a dedicated open work space.
- Set a sustainable pace.
- A stand up meeting starts each day.
- The Project Velocity is measured.
- Move people around.
- Fix XP when it breaks.

Designing

- Simplicity.
- 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.

Coding

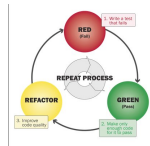
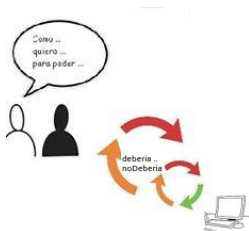
- The customer is always available.
- Code must be written to agreed standards.
- Code the unit test first.
- All production code is pair programmed.
- Only one pair integrates code at a time.
- Integrate often.
- Set up a dedicated integration computer.
- Use collective ownership.

Testing

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

Guías

BDD: Behavior Driven Development



Incremental

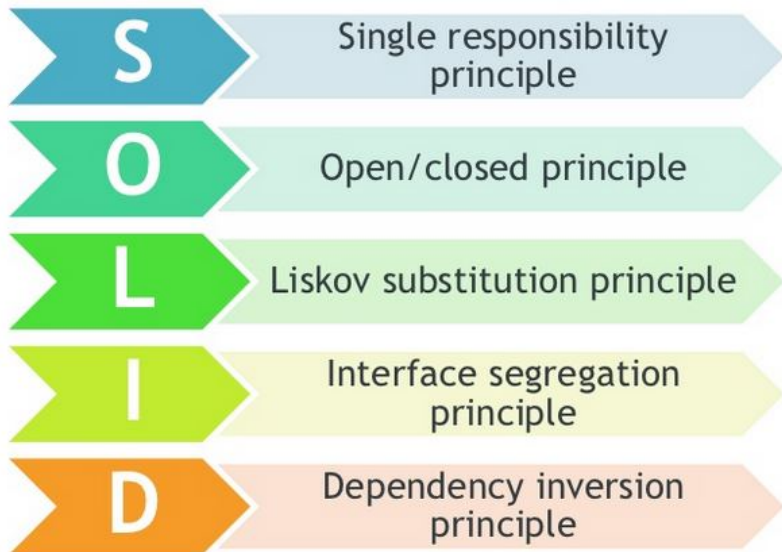


Iterative



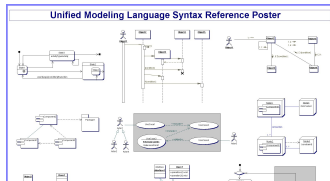
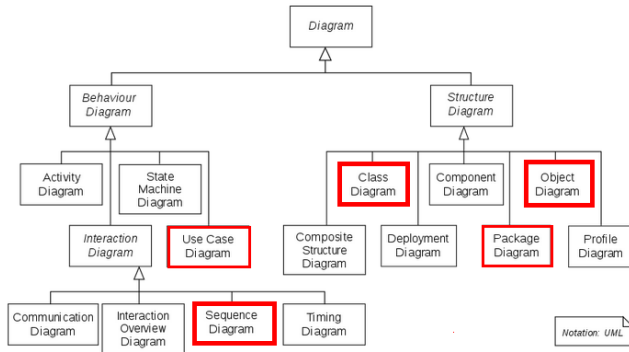
MDD: Model-Driven Development





Lenguajes

UML: Unified Modeling Language



Lenguajes

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 3 P

Investigación: Guías + Lenguajes

POOB-Curso

Descripción

Prácticas, lenguajes y herramientas

Orientación a objetos

Materia prima

Clases y objetos

Atributos y 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



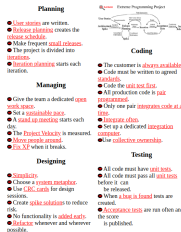
SOLID



XP



The Rules of Extreme Programming



UML



Java



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: 50% Exámen 50% Sustentación]

- ▶ 10% Quices y trabajos en clase

- ▶ 20% Laboratorio

Maratón *HackerRank Java* BONO 0.5 Laboratorio 3er tercio

Inicio: semana 1 Cierre: semana 16

- ▶ 20% Proyecto

T1 y T2 [100% Sustentación], T3 [50% Sustentación 50% Exámen]

Prácticas



The Rules of Extreme Programming

L06

Planning

L06

- **User stories** are written.
- **Release planning** creates the **release schedule**.
- Make frequent **small releases**.
- The project is divided into **iterations**. L01
- **Iteration planning** starts each iteration.

Managing

- Give the team a dedicated **open work space**.
- Set a **sustainable pace**.
- A **stand up meeting** starts each day.
- The **Project Velocity** is measured.
- **Move people around**.
- **Fix XP** when it breaks.

Designing

- **Simplicity**. L04
- Choose a **system metaphor**.
- Use **CRC cards** for design sessions. L02
- Create **spike solutions** to reduce risk.
- No functionality is **added early**.
- **Refactor** whenever and wherever possible. L04



Coding

- The customer is **always available**.
- Code must be written to agreed **standards**. L03
- Code the **unit test first**. L03
- All production code is **pair programmed**. L01
- Only one pair **integrates code at a time**. L06
- **Integrate often**.
- Set up a dedicated **integration computer**.
- Use **collective ownership**. L06

Testing

- All code must have **unit tests**. L02
- All code must pass all **unit tests** before it can be released.
- When a **bug is found** tests are created. L05
- **Acceptance tests** are run often and the score is published. L05

¿?

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

¿?

- ▶ ¿Cuáles son las herramientas JDK?

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

Herramientas

Moodle

505. Patrones de software

514. Entrada-Salida

513. Colecciones

512. GUI

511. Cierre: Evaluación

10. Cierre: Práctica

509. Excepciones

508. Lo abstracto y las interf...

507. Relaciones entre objetos

506. Cierre: evaluación

505. Cierre: práctica

504. Objetos e interacciones

503. Conceptos finales

502. Clases y o... Destacado

501. Introducción

PROGRAMACION ORIENTA A OBJETOS 2024-1

Curso

Configuración

Participantes

Calificaciones

Informes

Más

AL FINALIZAR EL CURSO EL ESTUDIANTE ESTÁ EN CAPACIDAD DE DESARROLLAR APLICACIONES SOFTWARE DE TAMAÑO PEQUEÑO EN EL PARADIGMA ORIENTADO A OBJETOS APLICANDO METODOLOGÍAS, TÉCNICAS Y HERRAMIENTAS ADECUADAS

PROFESORES: G01. DAVID ALEJANDRO VASQUEZ CARREÑO G03. RODRICO HUMBERTO GUALTERO G04. JUAN SEBASTIAN GARCIA HINCAPIE

MONITORES: G01. G02. G03

 CONTENIDO PROGRAMÁTICO

 NOTICIAS

 EVALUACIÓN

100% - Comprobación de lectura y trabajos en clase
20% - Laboratorio (Reglas de inasistencia: entrega individual y 75% de la nota. Regla incumplimiento de entrega 20% de penalización diaria. Límite de originalidad: 70%)
20% - Proyecto Inicial (1 y 2 tercios)
50% - Parcial 1 y 2 tercios. (T1: VI+1 MAR, T2: VI+2 ABR)
70% - Parcial 3 tercios. Proyecto final (3 tercios) 50% evaluación de equipo (sustentación) 20% evaluación individual (examen final)

Agregar un bloque

PROCESOS

 MANIFIESTO ÁGIL

 ESQUINA XP

LENGUAJES

 UML

 JAVA

 LANGUAGE SPECIFICATION

 CODE CONVENTIONS

 API

 W3SCHOOLS

HERRAMIENTAS



Herramientas

Arenas

BANCO DE CASOS

ARENA JAVA

PROYECTO FINAL

PROYECTO INICIAL

S16. Cierre

S15. Patrones de software

S14. Entrada-Salida

S13. Colecciones

12. GUI

11. Cierre: Evaluación

10. Cierre: Práctica

S09. Excepciones

S08. Lo abstracto y las interfaces

S07. Relaciones entre objetos

S06. Cierre: evaluación

S05. Cierre: práctica

S04. Objetos e interacciones

S03. Conceptos finales

S02. Clases y objetos

S01. Introducción



Java
Challenge Solver

HACKERRANK JAVA

De S01 a S16 Maratón individual.

Publicar imagen de avances al final de los tercios (S05 S10 S16)



Avance HackerRank



PROBLEMA DE MARATÓN INTERNACIONAL

S16 PUBLICAR FUENTES Y RESULTADOS EN ARENA



icpc. Resultados en arena



icpc. Fuentes



Agenda

Iniciando

Las 3 P

Investigación: Guías + Lenguajes

POOB-Curso

Descripción

Prácticas, lenguajes y herramientas

Orientación a objetos

Materia prima

Clases y objetos

Atributos y métodos

Casa

Herramienta. BlueJ

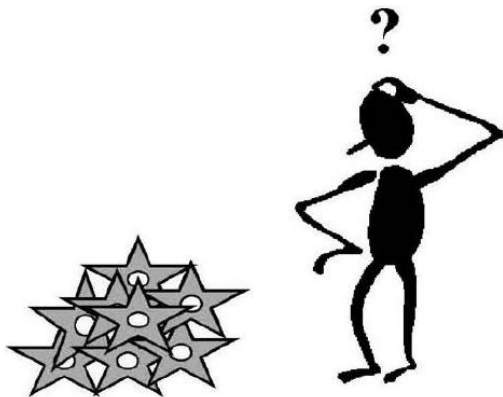
General

Editar+Compilar+Ejecutar+Documentar

Historia



Historia



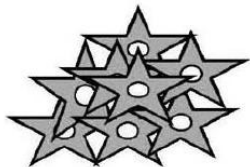
Historia



Historia



Clases y objetos



¿?

► ¿Qué son las estrellas?

Objeto

Objeto (en el mundo real)

Objeto (en software)

¿?

- ▶ ¿Qué es un objeto en el mundo real?
- ▶ ¿Qué es un 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)

¿?

- ▶ ¿Qué es una clase en el mundo real?
- ▶ ¿Qué es una 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)

Una **clase** define las características - atributos y métodos - que cada objeto que pertenece a la clase posee. Puede ser visto como un molde.

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)

Una **clase** define las características - atributos y métodos - que cada objeto que pertenece a la clase posee. Puede ser visto como un molde.



- ▶ Atributos
- ▶ Métodos

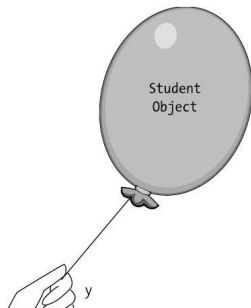
Objetos

Objeto (en software)

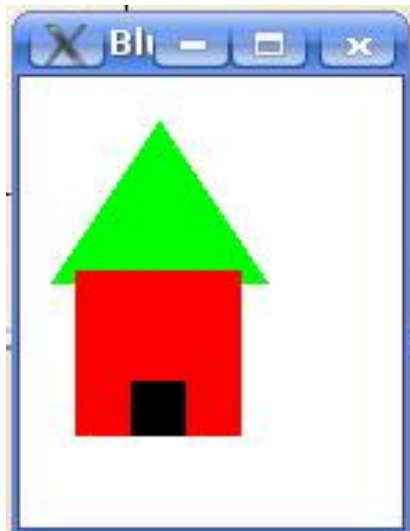
Un **objeto** es un artefacto software que representa una abstracción de un objeto del mundo real por medio de su estado (datos) y comportamiento (funciones).

Objeto

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



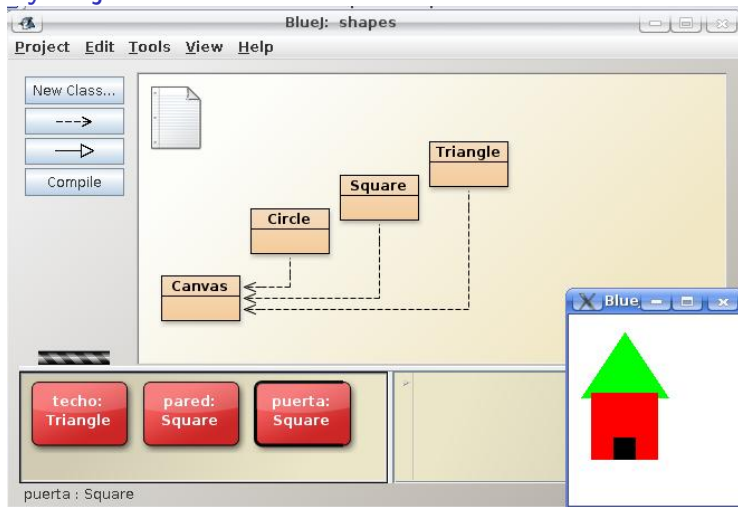
Clases y objetos



¿?

- ▶ ¿Cuántas clases?
- ▶ ¿Cuántos objetos?

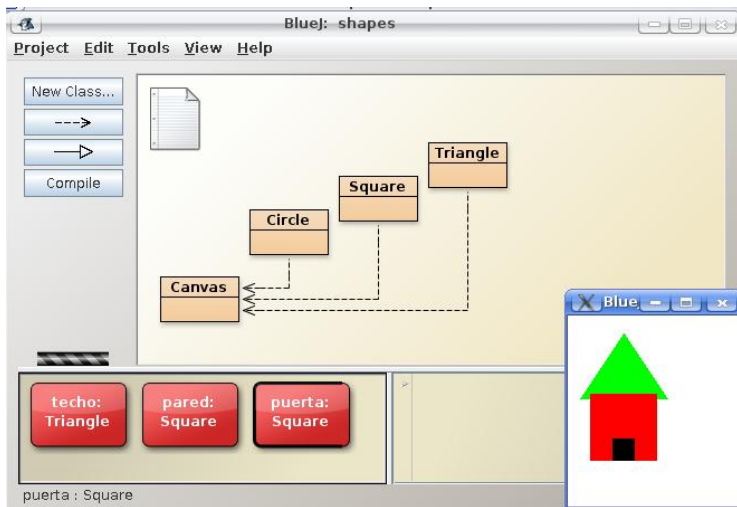
Clases y objetos



¿?

- ▶ ¿Cuántas clases?
- ▶ ¿Cuántos objetos?

Clases y objetos



Clase cuadrado

- ¿Qué atributos debe tener?

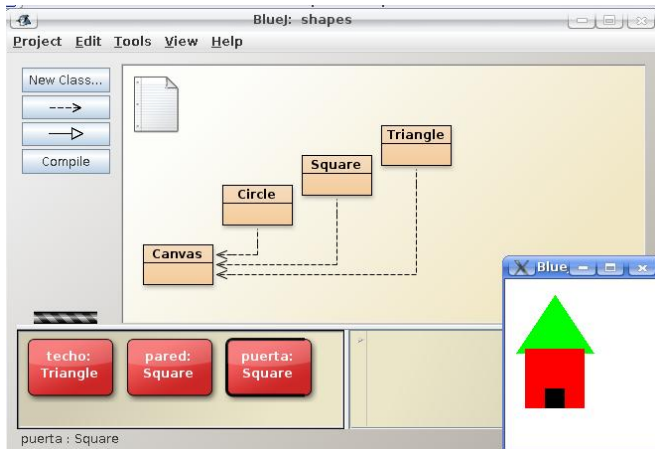
Clases y objetos

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

Clase cuadrado

- Atributos

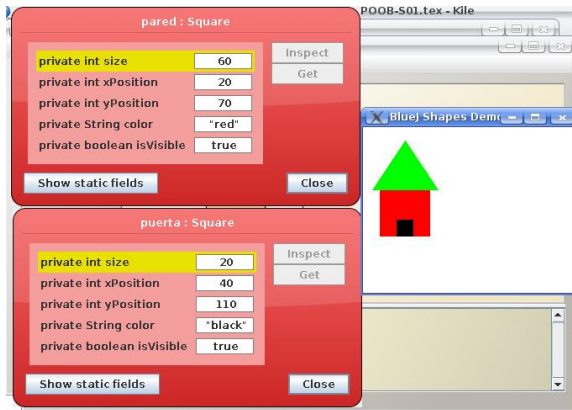
Clases y objetos



Objetos cuadrados

- ¿Estado?

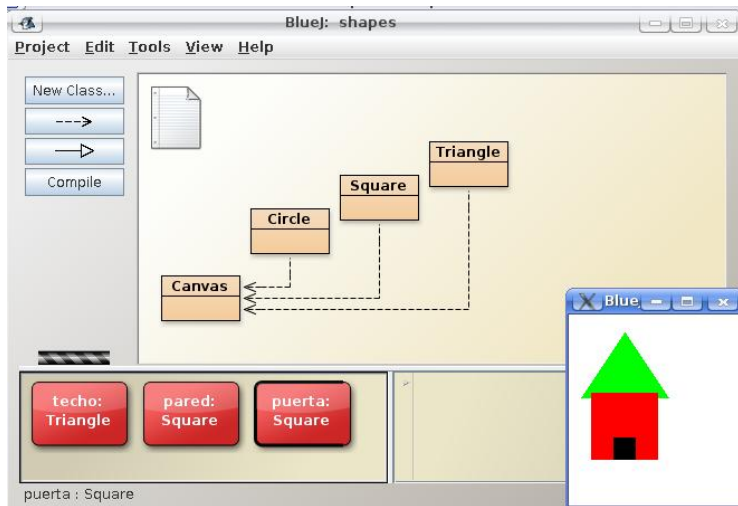
Clases y objetos



Objetos cuadrados

- Estado

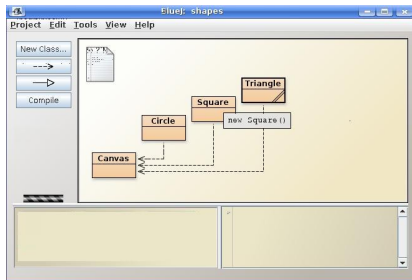
Clases y objetos



Clase cuadrado

- ¿Qué métodos debe ofrecer?

Clases y objetos



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

¡ Crear un cuadrado !

► ¿Método para crearlo?

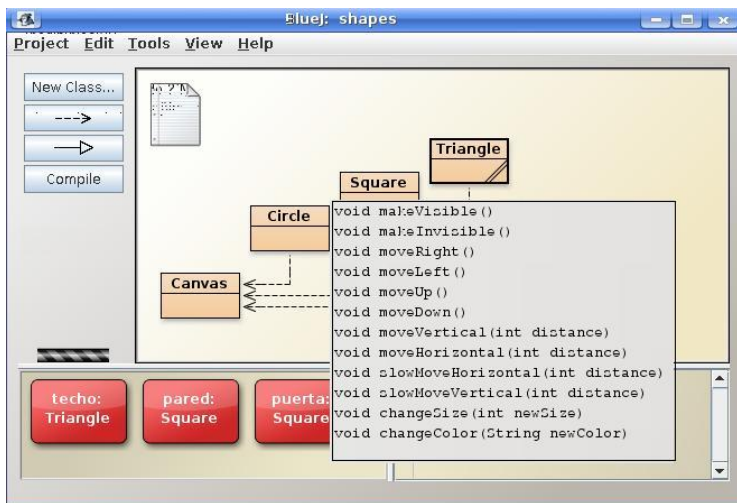
Clases y objetos

```
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;  
    }  
    ...  
}
```

¡ Crear un cuadrado !

- Método para crearlo

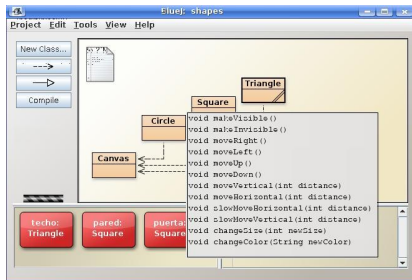
Clases y objetos



Clase Cuadrado

- ▶ Otros métodos

Clases y objetos



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

¡ Mover un cuadrado !

- ¿Método para mover horizontalmente?

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();  
    }  
  
    ...  
}
```

¡ Mover un cuadrado !

- Método para mover horizontalmente

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();  
        }  
    }  
    ...  
}
```

Mover lentamente un cuadrado

- TIPOS DE DATOS: Entero - Boolean - Cadena

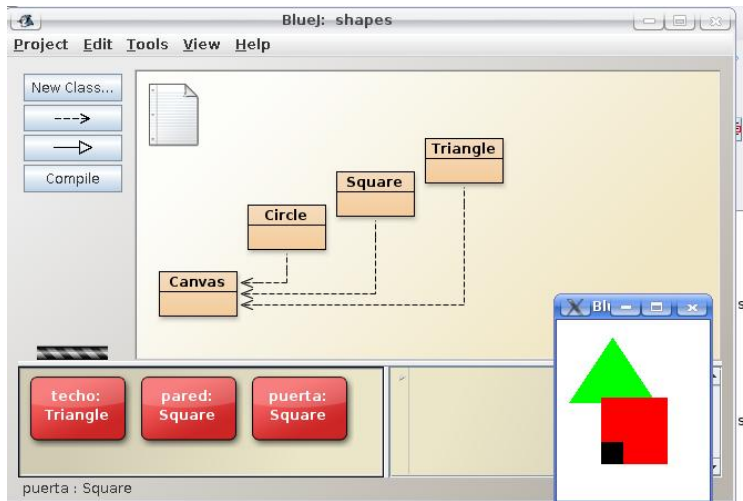
Clases y objetos

```
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();  
        }  
    }  
    ...  
}
```

Mover lentamente un cuadrado

- INSTRUCCIONES : Asignación - Condicional - Iteración

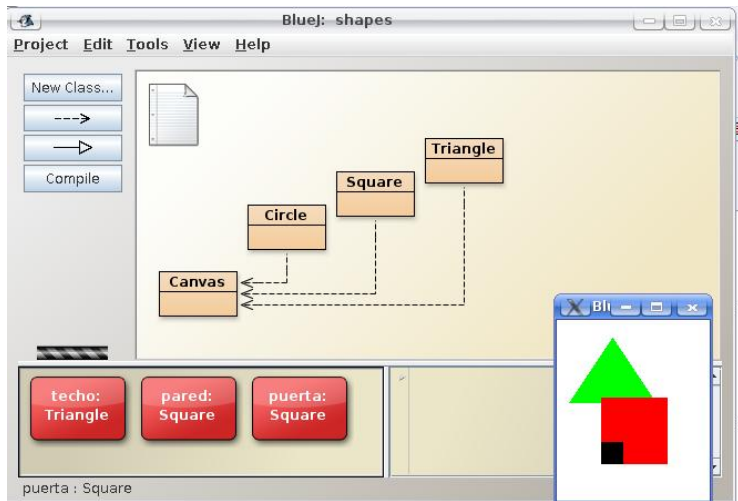
Clases y objetos



¿?

► ¿Qué pasó?

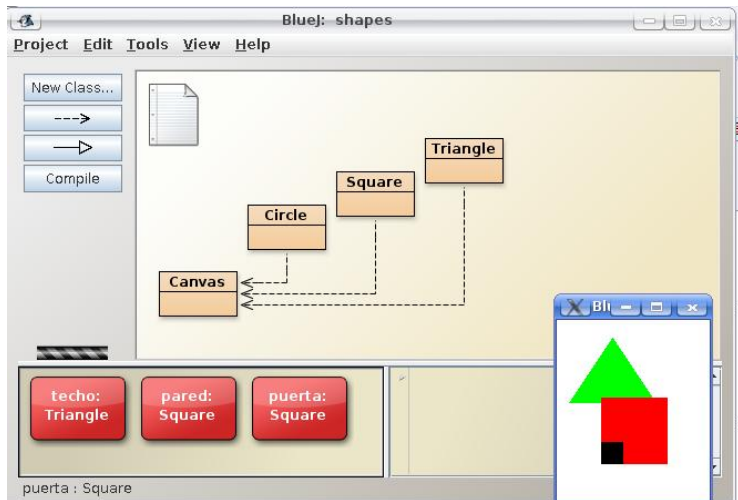
Clases y objetos



¿?

- ¿Quién no existe? Aunque la vemos ...

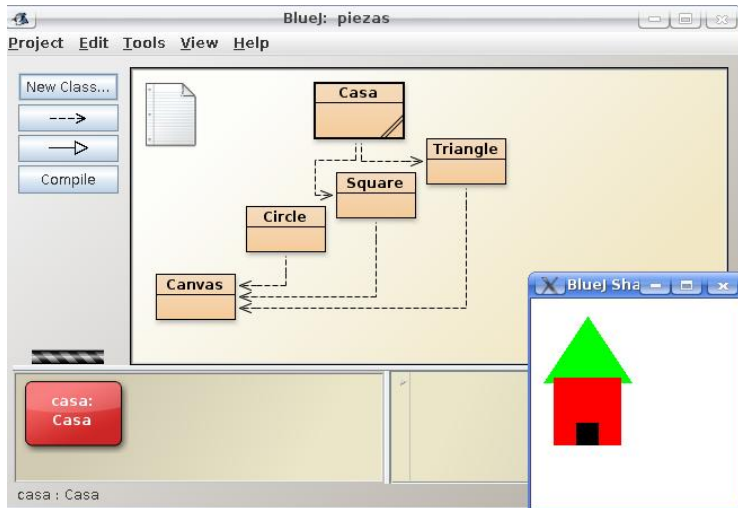
Clases y objetos



¿?

► ¿Qué hacemos?

Clases y objetos



Casa

► ¿Atributos?

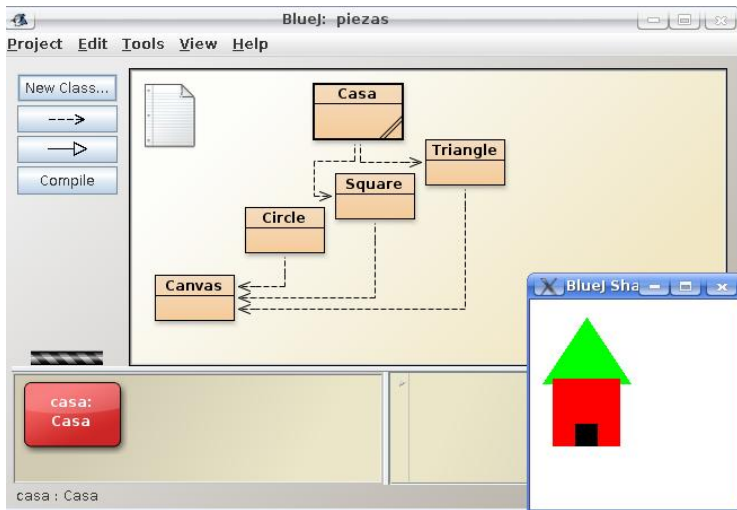
Clases y objetos

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

Casa. Estructura.

- ▶ ¿Diseño? (Ingeniería Reversa)

Clases y objetos



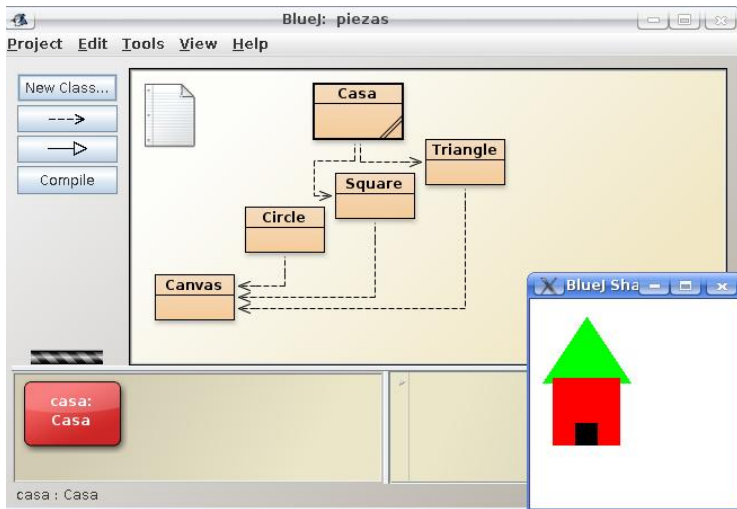
Casa

- ¿Creador? (No considerar ubicación ni tamaño)

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



Casa

► ¿Otros métodos?

Clases y objetos

The screenshot shows the BlueJ IDE window titled "BlueJ: piezas". The menu bar includes "Project", "Edit", "Tools", "View", and "Help". On the left, there are buttons for "New Class...", a dashed arrow, a solid arrow, and "Compile". Below these is a small icon of a film strip and a red button labeled "casa: Casa". At the bottom left, it says "casa : Casa".

The main workspace displays a class hierarchy diagram with four classes: "Casa", "Triangle", "Square", and "Circle". "Casa" is at the top, with dashed arrows pointing down to "Triangle" and "Square". "Square" has a dashed arrow pointing down to "Circle".

A code editor window is open, showing the following methods for the "Casa" class:

```
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)
```

On the right, a small window titled "BlueJ Sha" displays a graphical representation of a house: a red rectangle for the body, a green triangle for the roof, and a small black square for the door.

Casa

- ▶ Método para hacer visible la casa

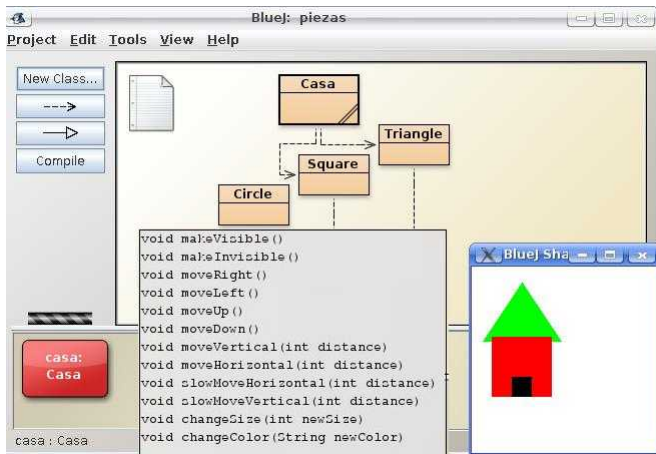
Clases y objetos

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

Casa. Estructura.

- ¿Diseño? (Ingeniería Reversa)

Clases y objetos



Casa: mover espacio

- ▶ ¿Código?
- ▶ ¿Diseño?

Agenda

Iniciando

Las 3 P

Investigación: Guías + Lenguajes

POOB-Curso

Descripción

Prácticas, lenguajes y herramientas

Orientación a objetos

Materia prima

Clases y objetos

Atributos y métodos

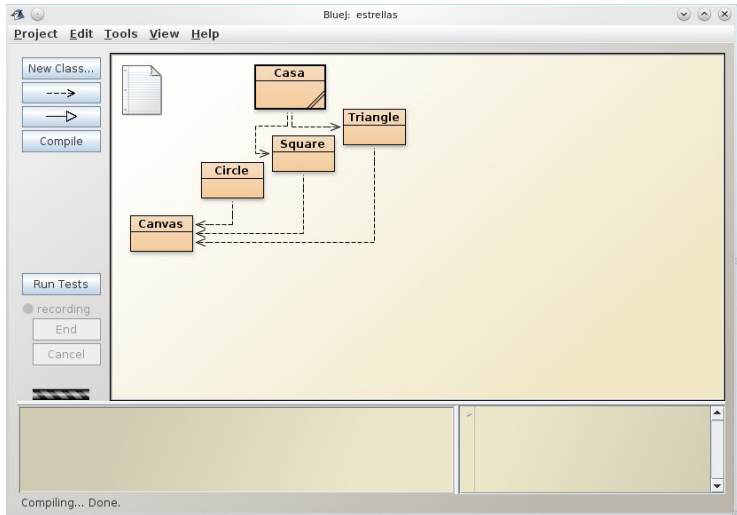
Casa

Herramienta. BlueJ

General

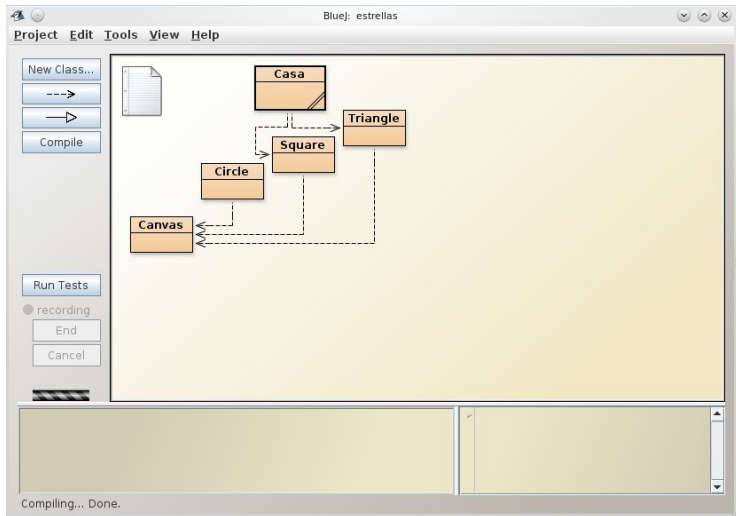
Editar+Compilar+Ejecutar+Documentar

General



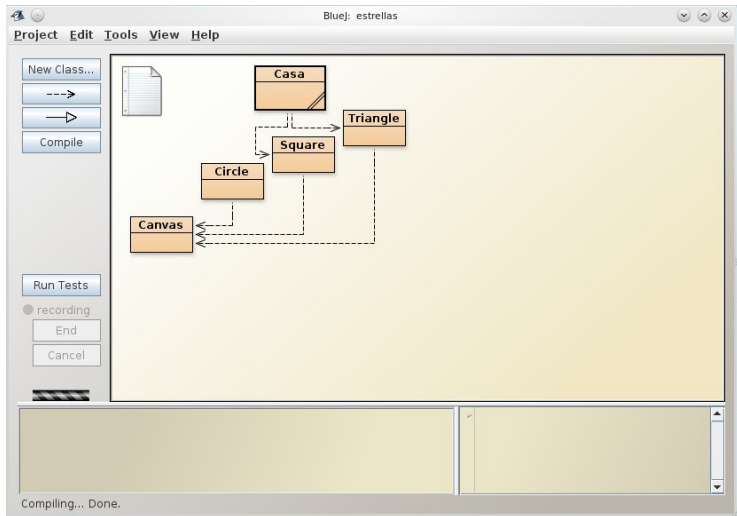
¿Qué debería permitir?

General



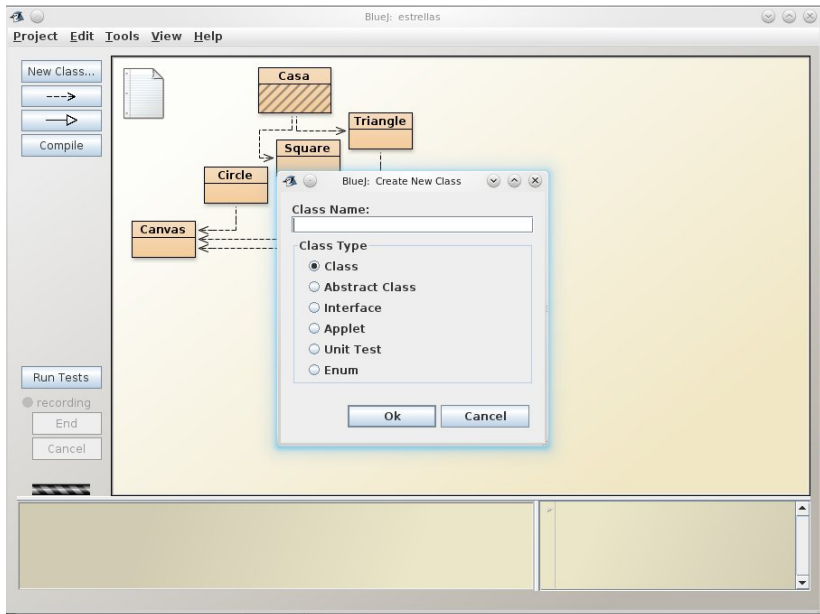
¿Qué usa?

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?

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 crea un círculo? (Llámelo point)

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 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;

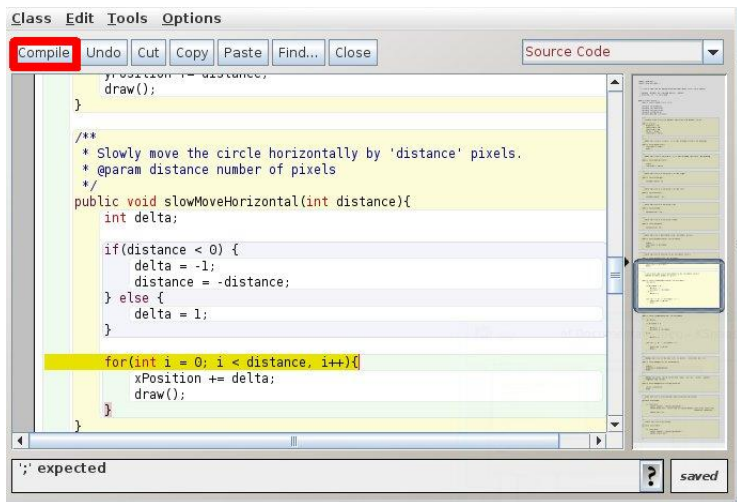
    private int diameter;
    private int xPosition;
    private int yPosition;
    private String color;
    private boolean isVisible;

    /**
     * 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();
    }
}
```

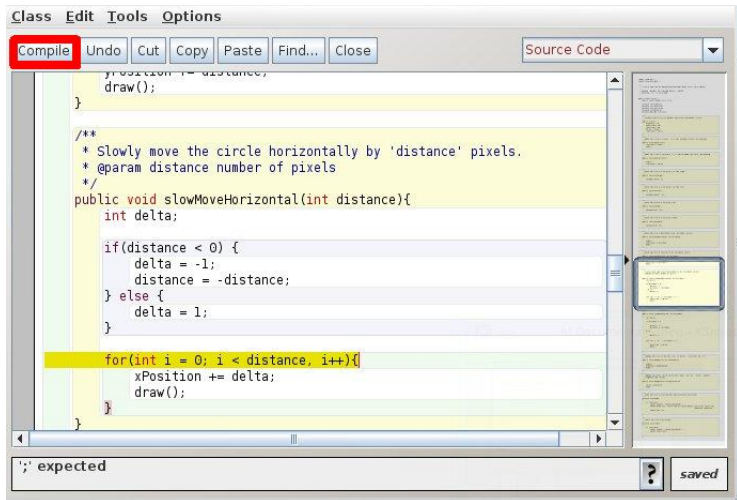
¿Cuál es el nombre del archivo?

Compilar



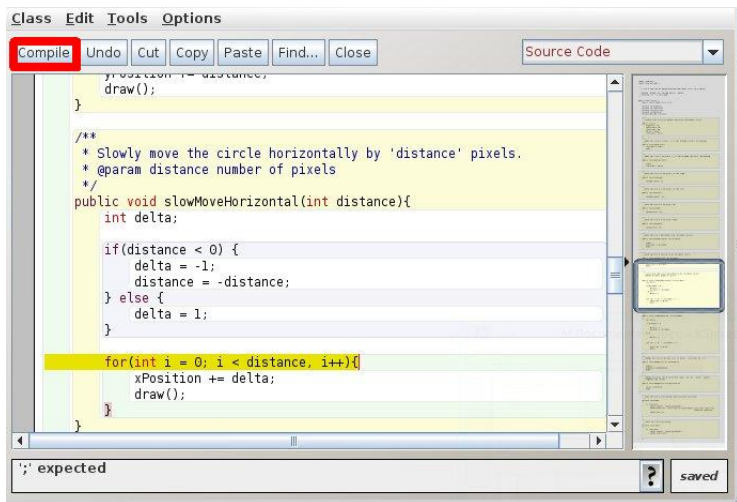
¿Cuál es la herramienta básica JDK?

Compilar



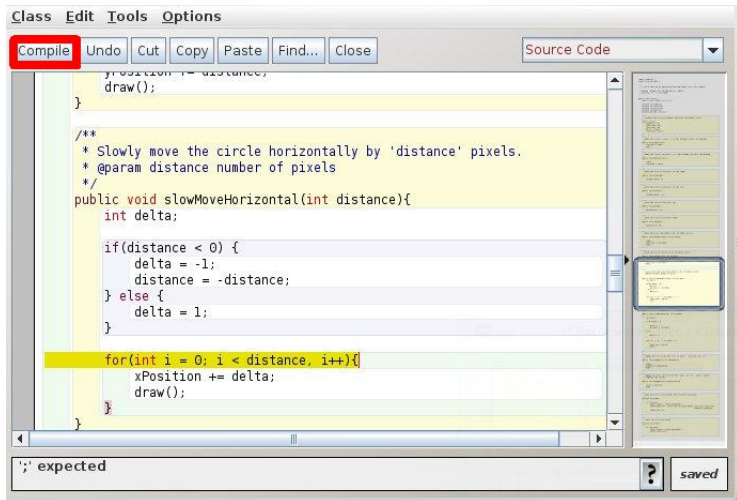
¿Qué error tiene el código?

Compilar



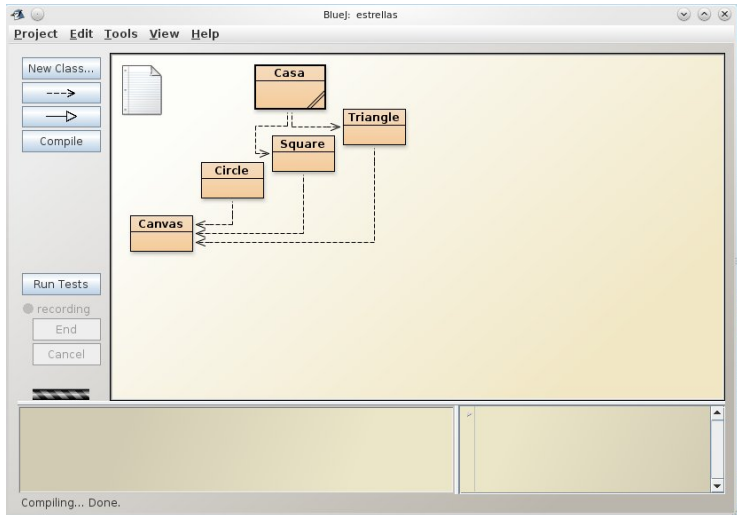
¿Qué hace el método?

Compiler



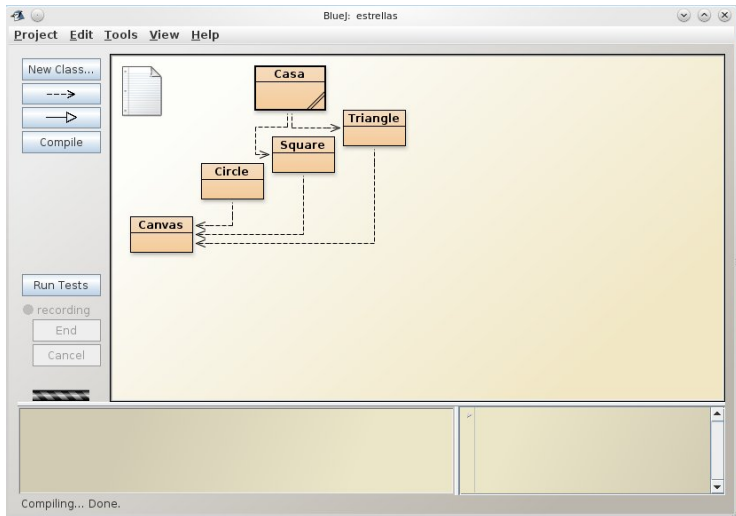
¿Cómo lo hace?

Ejecutar



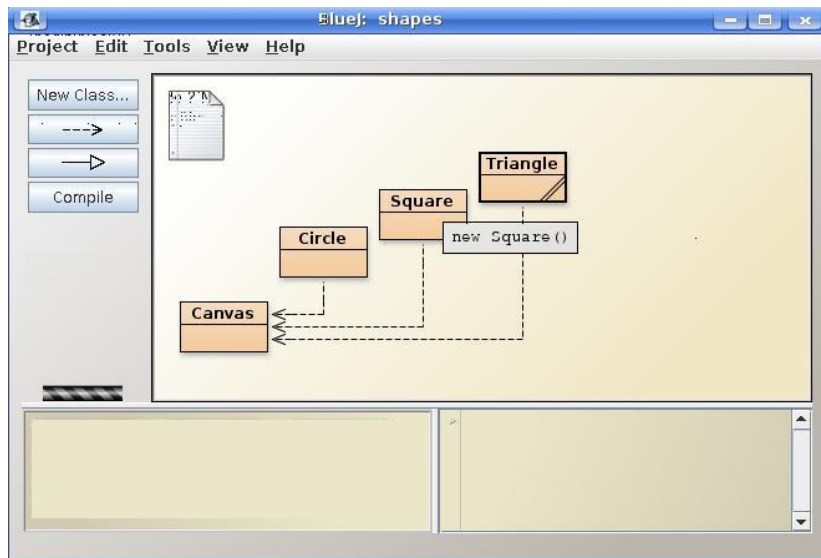
¿Cuál es la herramienta básica JDK?

Ejecutar



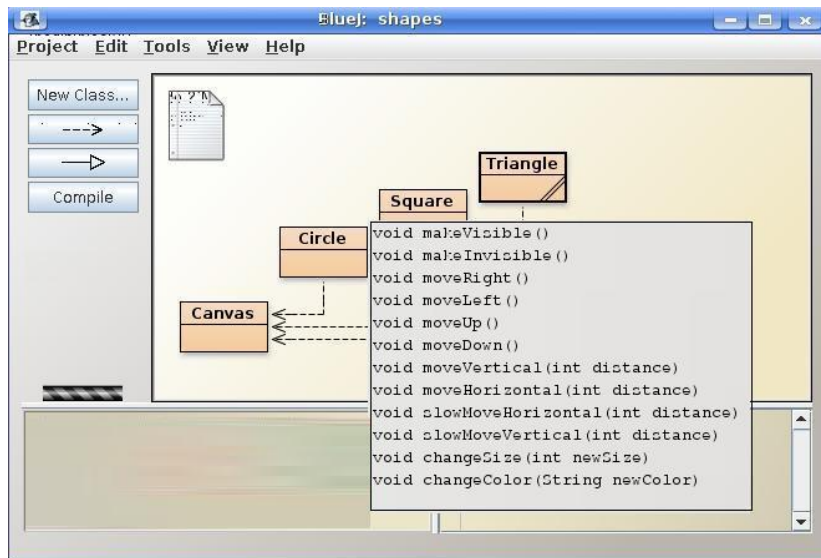
¿A quiénes podemos ejecutar?

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 workspace 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** (base class)

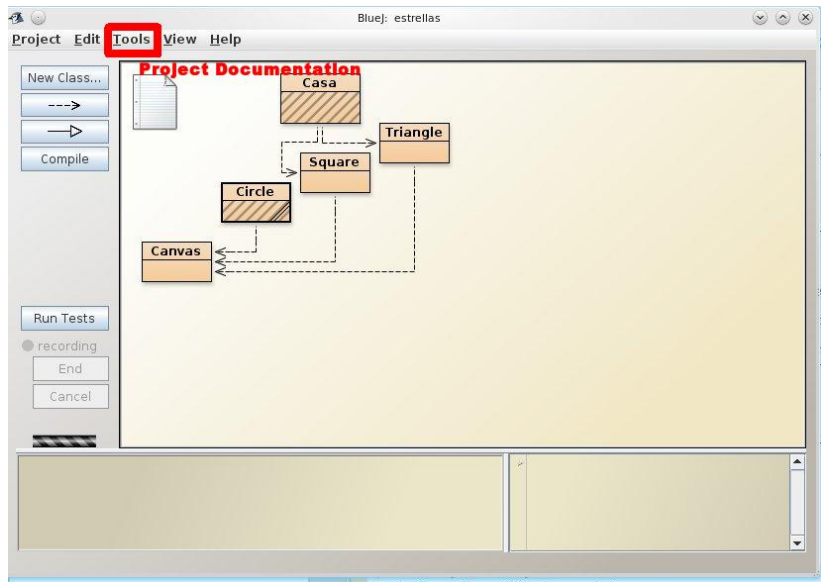
A red dialog box titled "bomba : Circle" is open, displaying the object's attributes and their values:

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

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

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 ve?

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();
    }
}
```