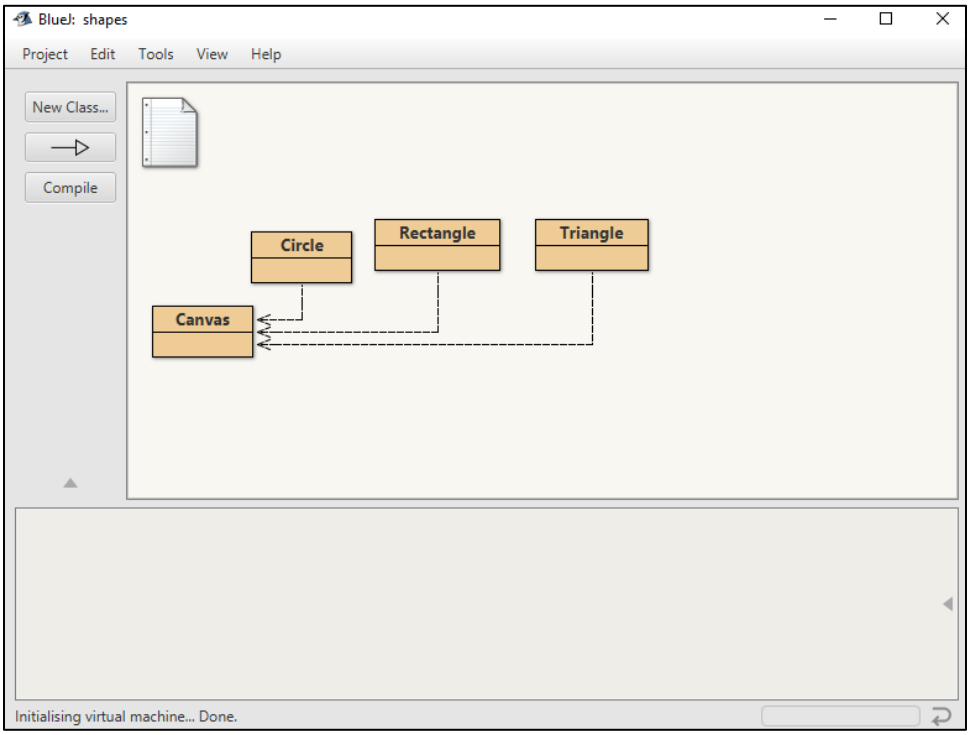
LABORATORIO #1: CONSTRUCCIÓN CLASES Y OBJETOS

A. Conociendo el proyecto shapes.

1. Captura de pantalla para proyecto `shapes`.



2. Diagrama de clases:

- ¿Qué clases ofrece?
  `Circle`
  `Rectangle`
  `Triangle`
  `Canvas`
- ¿Qué relaciones existen entre ellas?
  La flecha hace referencia a que una clase usa la otra, según la dirección de esta.
  `Circle → Canvas`
  `Triangle → Canvas`
  `Rectangle →  Canvas`

3. Documentación
- ¿Qué clases presenta el paquete `shapes`?
  `Circle – Rectangle – Triangle – Canvas`
- ¿Qué atributos ofrece la clase `Circle`?
  `PI`
- ¿Cuántos métodos ofrece la clase `Circle`?
  12
- ¿Cuáles métodos ofrece la clase `Circle` para que la figura cambie?
  `changeColor – changeSize`

4. Código:
- ¿Cuántos atributos realmente tiene?
  6
- ¿Cuáles atributos describen la forma de la figura?
  diameter
- ¿Cuántos métodos tiene en total?
  14
- ¿Quiénes usan los métodos privados?
  `draw – erase`

5. Comparando: Documentación – Código
- ¿Qué no se ve en la documentación?
  Atributos:
  `Diameter – xPosition – yPosition – color – isVisible`
  Métodos:
  `draw – erase`
- ¿Por qué debe ser así?
  No se ven en la documentación porque están privados.

6. Atributo PI:
- ¿Qué significa que sea `public`?
  Cualquier clase en cualquier paquete puede acceder al método.
- ¿Qué significa que sea `static`?
  La variable es única para todas las instancias de la clase.
- ¿Qué significa que sea `final`?
  Indica que la variable es de tipo constante y no admitirá cambios después de la declaración.
- ¿De qué tipo de datos debería ser? ¿Por qué?
  `Float`, porque este almacena números decimales con suficiente para contener de 6 a 7 dígitos decimales (32 bits).

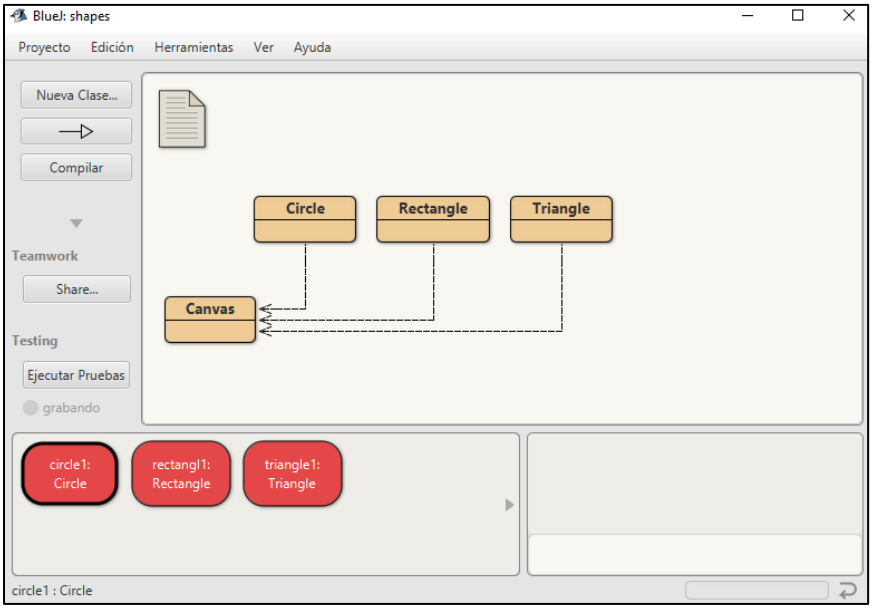
7. Tipo del atributo `diameter`:
   - ¿Qué se está indicando al decir que es `int`?
     Es un dato (32 bits) para almacenar valores numéricos.
   - Si sabemos todos círculos van a ser pequeños (diámetro menor a 100), ¿De qué tipo deberían ser este atributo?
     `Byte`, esto porque almacena números enteros de – 128 a 127.
   - Si son grandes, pero no tanto (diámetro menor a 30000), ¿De qué tipo deberían ser este atributo?
     `Short`, esto porque almacena números enteros de – 32768 a 32767.
   - ¿Qué restricción adicional tendría este atributo?
     Al ser una longitud esta debe ser positiva.

```
public static final float PI=3.1416f;

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

8. Propósito del Proyecto:
   Poder visualizar diferentes figuras geométricas.

B. Manipulando objetos: Usando un objeto.

1. Creación de un objeto:



- ¿Cuántas clases hay?, ¿Cuántos objetos crearon?

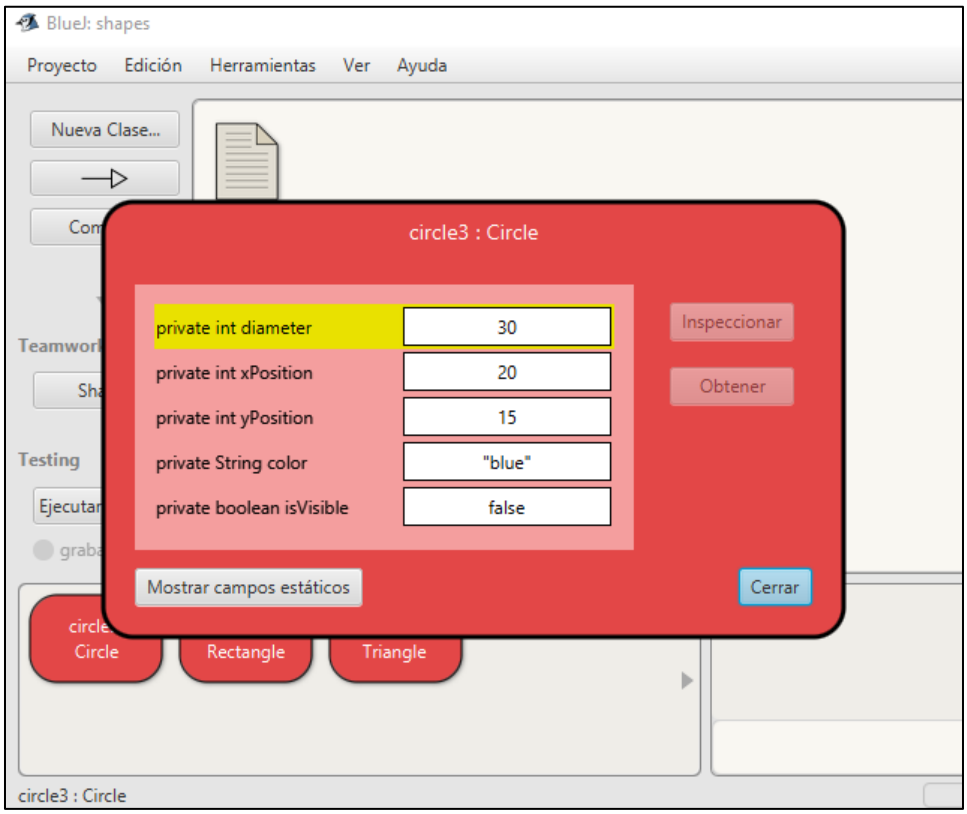
  Clases = 4
  Objetos = 3

- ¿Por qué?

  Porque las clases `circle`, `rectangle` y `triangle` usan la clase `canvas`

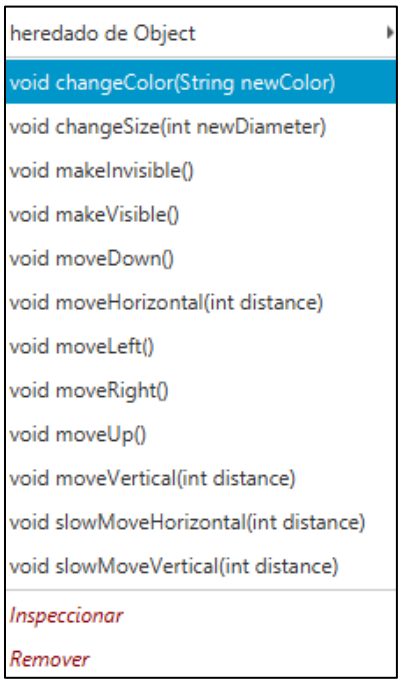
2. Inspección del estado del objeto, `circle`:

- ¿Cuáles son los valores de inicio de todos sus atributos?



3. Inspección del comportamiento del objeto, `circle`:

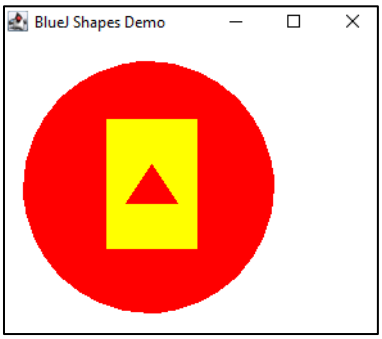
- Captura de pantalla:



- ¿Por qué no aparecen todos los que están en el código?
  Porque están privados y estos no pueden ser modificados.

4. Construcción de un logo con `shapes`:

- ¿Cuántas y cuáles clases se necesitan?
  Se necesitan 4 clases y estas son: `Canvas, Circle, Triangle, Rectangle`
- ¿Cuántos objetos se usan en total?
  3
- Captura de pantalla:

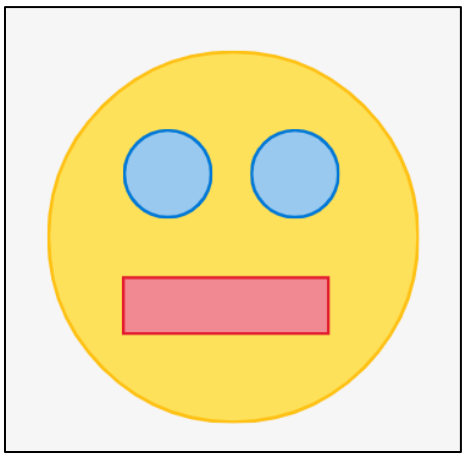


- Logo original:

C. Manipulando objetos: Analizando y escribiendo código.

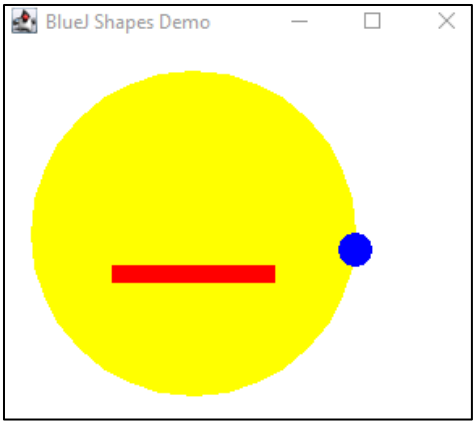
1. Leyendo código:

- ¿Cuál es la figura resultante?
  Una cara seria.
- Pitándola:



2. Escribiendo código:

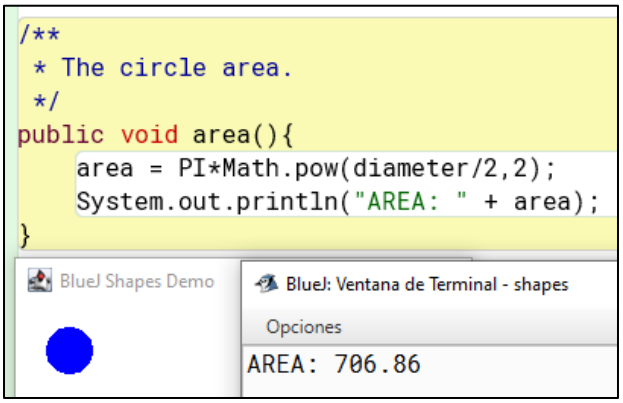
- ¿Cuántas variables existen?
  4: `face, rEye, lEye y mouth`
- ¿Cuántos objetos existen?
  3: Dado que se uso 2 veces el constructor de `circle` y 1 vez el de `rectangle.`
- ¿Qué color tiene cada uno de ellos?
  `face`: Yellow
  `mouth`: Red
  `rEye – lEye`: Blue
- ¿Cuántos objetos se ven?
  3: `2 circle, 1 rectangle`
- Capturando pantalla:



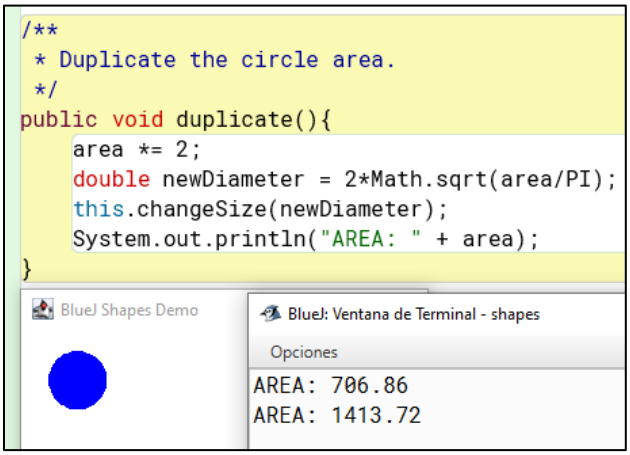
3. Comparando figuras:
   - ¿Son iguales?
     No.
   - ¿Por qué?
     Pesamos que `rEye` y `lEye` serian diferentes por lo que serían tratados como distintos objetos.

D. Extendiendo una clase: `Circle`.

1. Desarrollando método `area()`:
   Capturando pantalla:



2. Desarrollando método `duplicate()`:
Capturando pantallas:



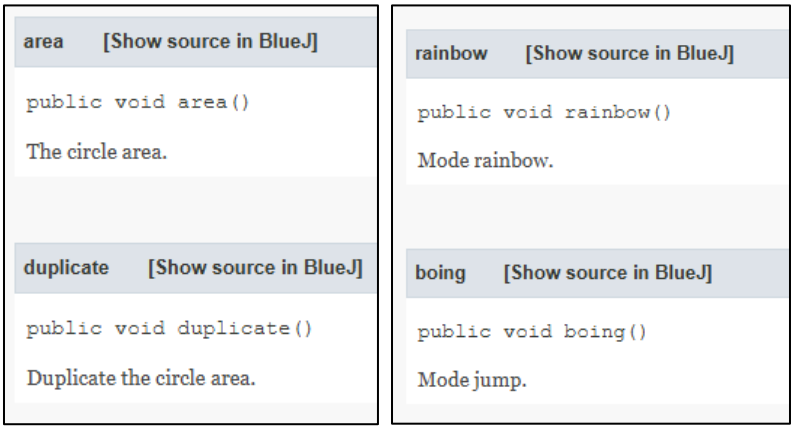
3. Desarrollando método `rainbow()`:
Capturando pantallas:

Ver video: [Rainbow prueba]

4. Proponiendo un nuevo método:
Método boing: Simula el rebote de una pelota.

Ver video: [Boing prueba]

5. Generando una documentación para nuevos métodos:
Capturando pantalla:



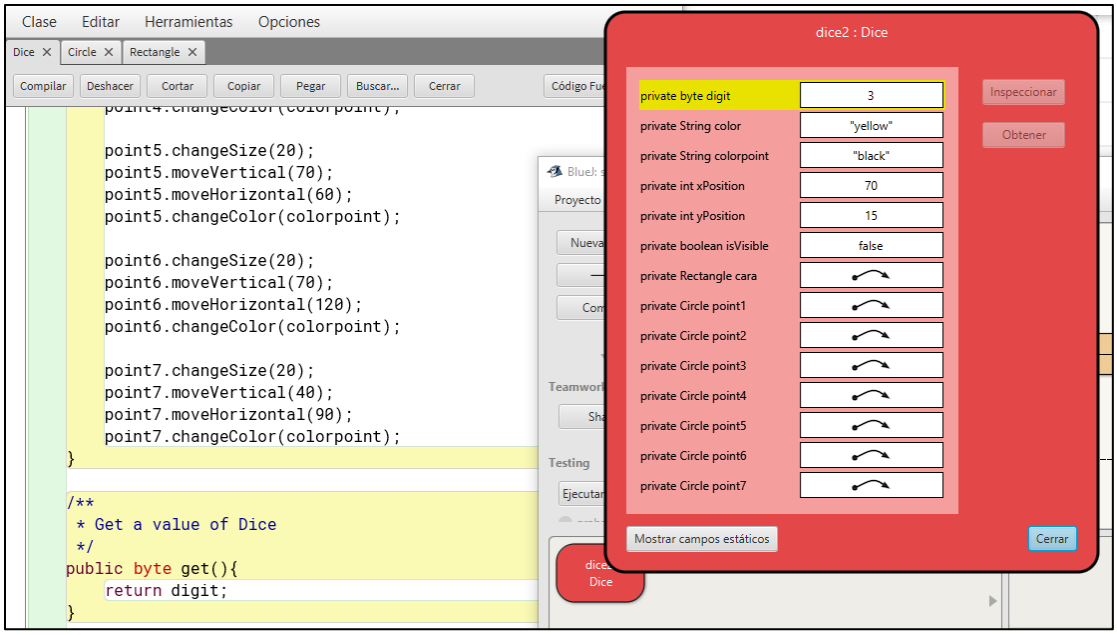
E. Codificando una nueva clase: *Dice*.

1. Clasificando los métodos:

```
_(digit:byte) :Dice -> Constructor

get() : byte -> Analizador

next(): void -> Analizador

change(digit:byte) : void -> Modificador

change(): void -> Analizador

moveTo(x:int, y:int): void -> Modificador

changeColor(color: String) : void -> Modificador

makeVisible(): void -> Modificador

makeInvisible(): void -> Modificador
```

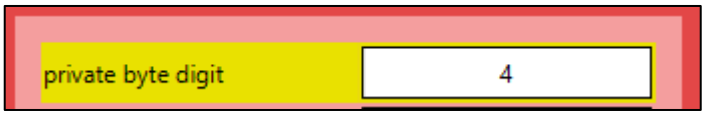
2. Desarrollando la clase Dice:
Capturando pantalla: Mini-ciclo #1 Constructor, inicializa los atributos del objeto.

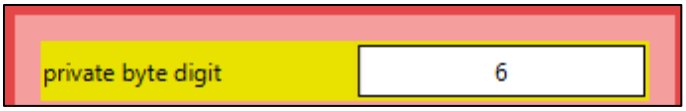
```
/**
 * Constructor for objects of class Dice
 * @param valor is between 1 and 6
 */
public Dice(byte valor) {
    digit = valor;
    color = "yellow";
    colorpoint = "black";
    xPosition = 70;
    yPosition = 15;
    isVisible = false;
    cara = new Rectangle();
    point1 = new Circle();
    point2 = new Circle();
    point3 = new Circle();
    point4 = new Circle();
    point5 = new Circle();
    point6 = new Circle();
    point7 = new Circle();
    cara.changeSize(100,100);
    cara.changeColor(color);
```



Mini-ciclo #2 El método next muestra el valor

```
/**
 * Add 1 to digit
 */
public void next(){
    if (digit != 6){
        digit += 1;
        draw();
    }
    else {
        digit = 1;
        draw();
    }
}

/**
 * Change the digit of Dice
 * @param valor is between 1 and 6
 */
public void change(byte newDigit){
    digit = newDigit;
    draw();
}
```

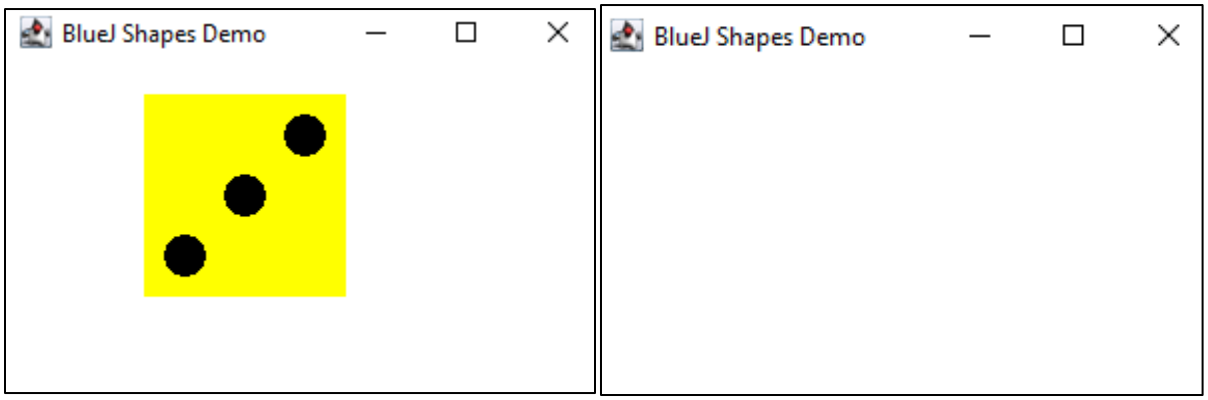




Mini-ciclo #3 Según el valor ingresado cambia el valor del dado.

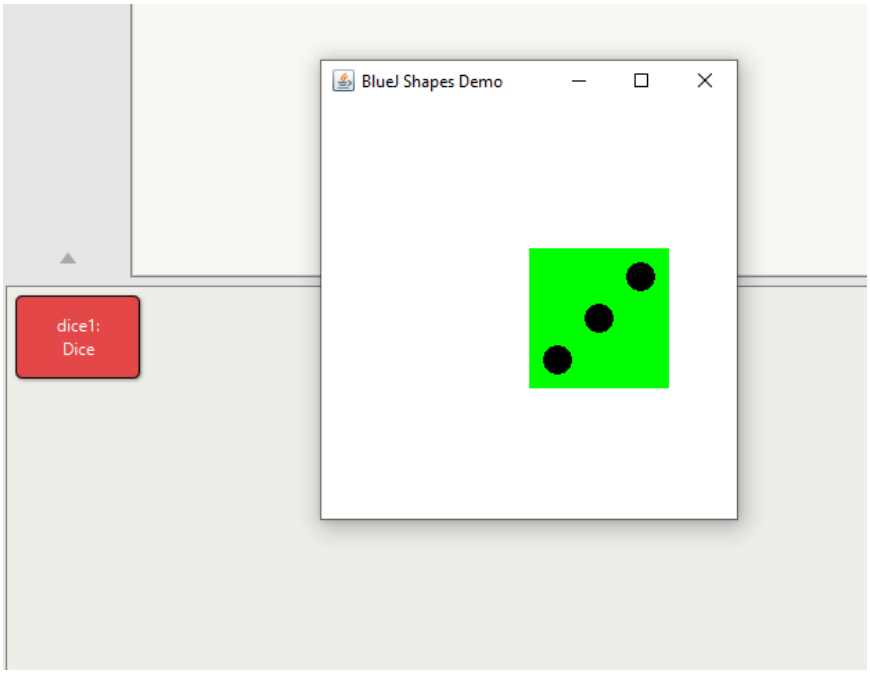
```
/**
 * Make the Dice visible
 */
public void makeVisible(){
    isVisible = true;
    draw();
}

/**
 * Make the Dice Invisible
 */
public void makeInvisible(){
    erase();
    isVisible = false;
}
```

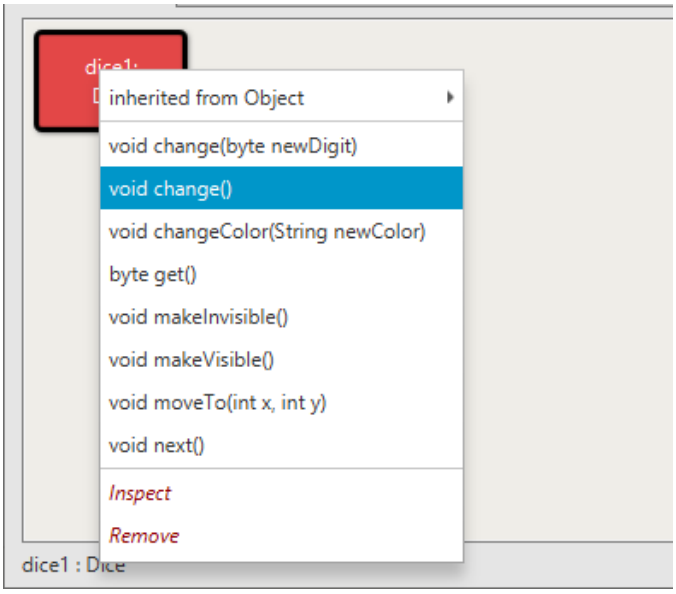
```
private void erase(){
    if(isVisible) {
        cara.makeInvisible();
        point1.makeInvisible();
        point2.makeInvisible();
        point3.makeInvisible();
        point4.makeInvisible();
        point5.makeInvisible();
        point6.makeInvisible();
        point7.makeInvisible();
    }
}
```

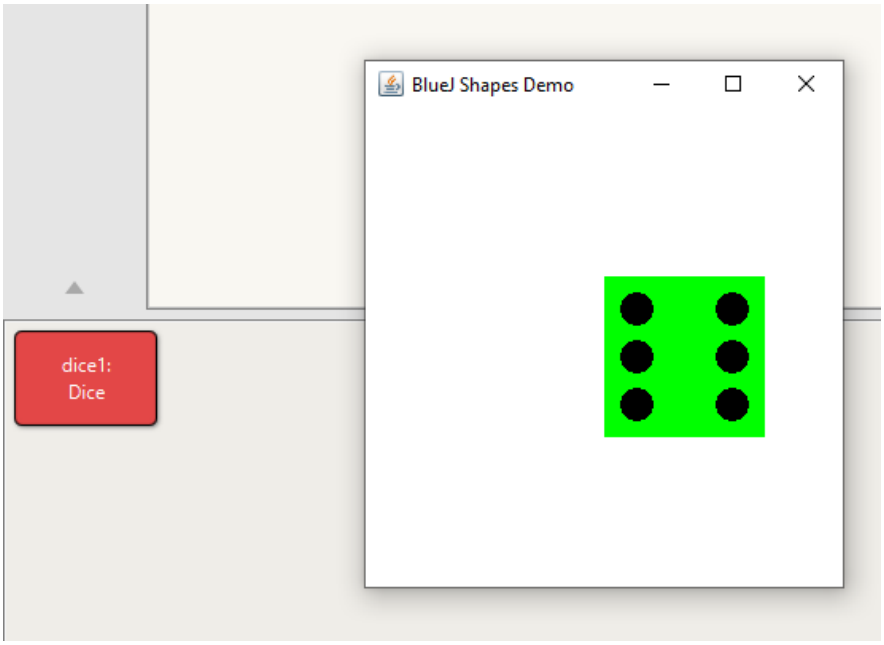


Mini-ciclo #6 Cambia el valor del dado de forma aleatoria.

```
/**
 * Change the digit random
 * @param valor is between 1 and 6
 */
public void change(){
    byte numDice = (byte)(Math.random()*6+1);
    change(numDice);
    draw();
}
```


BlueJ Shapes Demo
dice1:
Dice

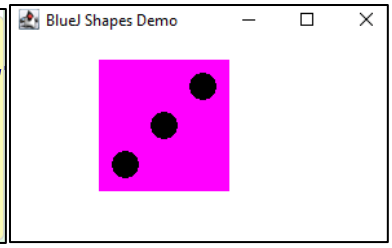

inherited from Object
void change(byte newDigit)
void change()
void changeColor(String newColor)
byte get()
void makeInvisible()
void makeVisible()
void moveTo(int x, int y)
void next()
Inspect
Remove
dice1 : Dice

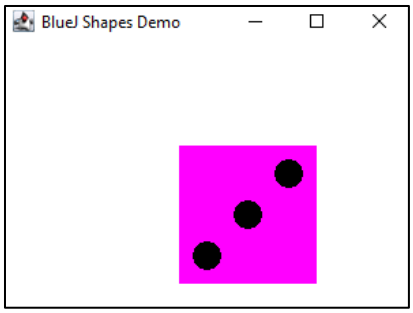


Mini-ciclo #5

```
/**
 * Move the Dice.
 * @param x, y distance the desired distance in pixels
 */
public void moveTo(int x, int y){
    erase();
    xPosition = x;
    yPosition = y;
    // mover todo dependiendo x
    cara.moveHorizontal(x);
    point1.moveHorizontal(x);
    point2.moveHorizontal(x);
    point3.moveHorizontal(x);
    point4.moveHorizontal(x);
    point5.moveHorizontal(x);
    point6.moveHorizontal(x);
    point7.moveHorizontal(x);
    // mover todo dependiendo y
    cara.moveVertical(y);
    point1.moveVertical(y);
    point2.moveVertical(y);
    point3.moveVertical(y);
    point4.moveVertical(y);
    point5.moveVertical(y);
```

```
/**
 * Change the color.
 * @param color the new color. Valid colors are "red", "yellow"
 * "magenta" and "black".
 */
public void changeColor(String newColor){
    color = newColor;
    cara.changeColor(color);
    draw();
}
```





F. Diseñando y codificando una nueva clase: *DiceTaken*.

1. Diseñando la clase DiceTaken:

DiceTaken
+ (digit: byte): DiceTaken
+ verificaTablaero(DiceTaken) : void
+ lanzar(): void
+ deslizar(): void
+ get() : void
+ delete() : void
+ makeVisible() : void
+ makeInvisible() : void
+ ganar(): void

2. Planificando la construcción:
Miniciclos

1. (digit: byte)
   get()
   delete()
2. Lanzar()
   Deslizar()
3. makeVisible()*
   makeInvisible()*

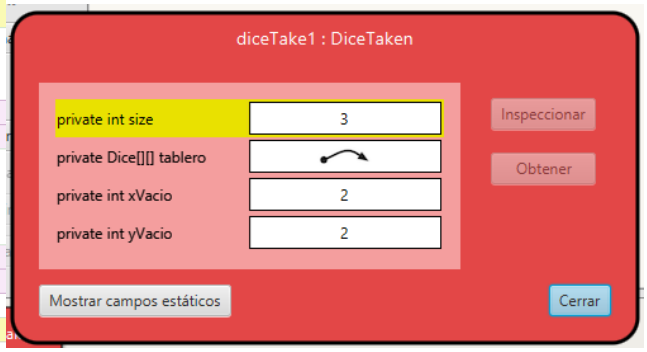
verificaTablero()*
ganar()*

3. Implementando la clase:

Mini ciclo #1

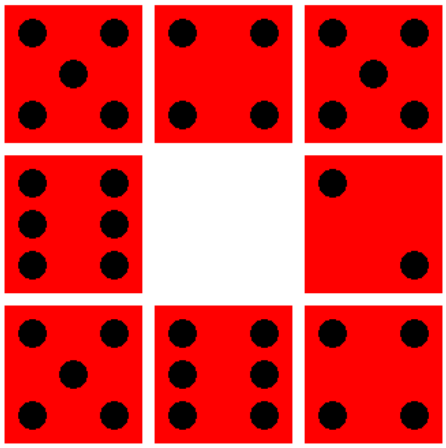
Atributos del tablero con tamaño 3

```
/**
 * Constructor para objetos de la clase DiceTaken
 */
public DiceTaken(int size)
{
    this.tablero = new Dice[size][size];
    this.size = size;

    for (int i = 0; i<this.tablero.length; i++){
        for (int j = 0; j<this.tablero[i].length;j++){
            byte digit_dice = (byte)(Math.random()*6+1);
            this.tablero[i][j] = new Dice(digit_dice);
            this.tablero[i][j].changeColor("red");
            this.tablero[i][j].moveTo( 110 * j,110 * i);
        }
    }
}
```



Obtención del tablero de juego

```
/**
 * Obtiene el tablero de juego
 */
public void get(){
    for (int i = 0; i<this.tablero.length; i++){
        for (int j = 0; j<this.tablero[i].length;j++){
            this.tablero[i][j].makeVisible();
        }
    }

    int i = (int)(Math.random()*this.size);
    int j = (int)(Math.random()*this.size);
    this.xVacio = i;
    this.yVacio = j;
    this.tablero[i][j].makeInvisible();
}
```

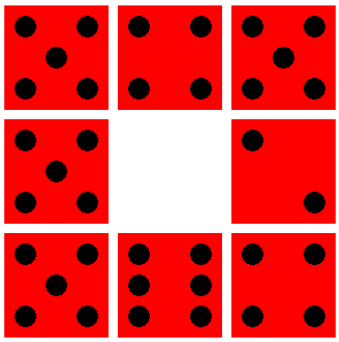


```java
/**
 * Borra el tablero de juego
 */
public void delete(){
    for (int i = 0; i<this.tablero.length; i++){
        for (int j = 0; j<this.tablero[i].length;j++){
            this.tablero[i][j].makeInvisible();
        }
    }
}
```

Mini ciclo #2

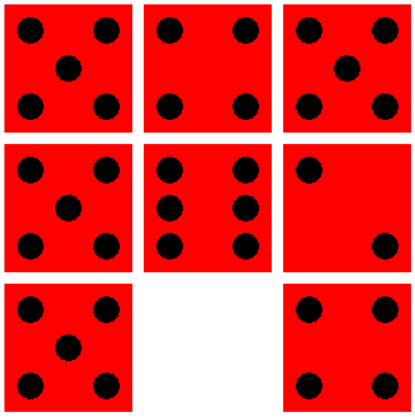
Se lanza la posición (2,1)

```java
/**
 * Lanzar un dado con posicion especifica
 */
public void lanzar(int x,int y){
    if (x-1 != this.xVacio && y-1 != this.yVacio){
        this.tablero[x-1][y-1].change();
    }
}
```



Se mueve la posición (3,2)

```java
/**
 * Desliza un dado que pueda moverse
 *
 * @param  x,y enteros y en el rango del tablero
 *
 */
public void deslizar(int x,int y)
{
    if (x-1 != this.xVacio && y-1 != this.yVacio){
        this.tablero[xVacio][yVacio].change(this.tablero[x-1][y-1].get());
        this.tablero[x-1][y-1].makeInvisible();
        this.tablero[xVacio][yVacio].makeVisible();
        this.xVacio = x-1;
        this.yVacio = y-1;
    }

}
```



Retrospectiva

1. ¿Cuál fue el tiempo total invertido en el laboratorio por cada uno de ustedes? (Horas/Nombre)

   (12/Angie Natalia Mojica)
   (12/Daniel Antonio Santanilla)

2. ¿Cuál es el estado actual del laboratorio? ¿Por qué?

   Queda pendiente la implementación que permite visualizar y verificar que el tablero cumple con los reglamentos del juego, esto hizo falta porque no se tenían los conocimientos claros para poder crear la interfaz.

3. Considerando las prácticas XP del laboratorio. ¿cuál fue la más útil? ¿por qué?

   Programación a pares: pues todo el trabajo se produjo de forma conjunta, donde simultáneamente estábamos analizando, diseñando, comprobando e intentando programar cada instrucción, consideramos que fue bastante útil, pues entre ambos fuimos aclarando conceptos y generando ideas.

4. ¿Cuál consideran fue el mayor logro? ¿Por qué?

   Aprender a implementar clases en java junto con sus métodos a partir de mini ciclos y pruebas. También se aprendió sobre la sintaxis básica de java; sus datos primitivos. Además, se aprendió a usar cosas básicas en BlueJ.

5. ¿Cuál consideran que fue el mayor problema técnico? ¿Qué hicieron para resolverlo?

   Poder implementar una matriz de objetos y obtener un numero aleatorio con la función random. Para resolverlo recurrimos a la documentación y las preguntas en el foro del laboratorio

6. ¿Qué hicieron bien como equipo? ¿Qué se comprometen a hacer para mejorar los resultados?

Como equipo se logró mantener una buena comunicación, haber destinado un tiempo específico para desarrollar el laboratorio y cumplirlo.

En vista de que no se logró el desarrollo completo del laboratorio debemos buscar información más clara que ayude a tener una mejor comprensión sobre el lenguaje, pues hubo temas que no dominamos en su totalidad lo que hizo que no pudiésemos visualizar la forma de resolverlo.