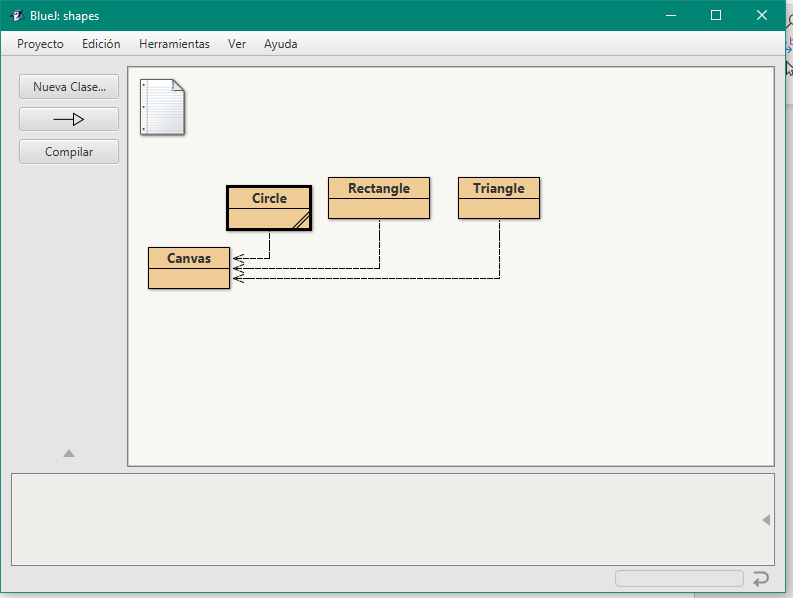
Tatiana Medina – Jose Pérez

POOB - 1

2020 - 2

Laboratorio 1/6 Introducción. Clases y objetos

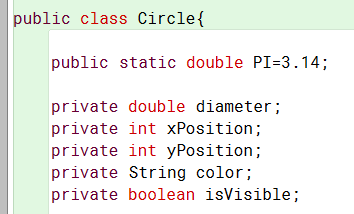
Conociendo el proyecto shapes



* 1. Nos ofrece 4 clases: *Canvas, Circle, Rectangle, Triangle*
  2. Las clases *Circle, Rectangle y Triangle son figuras geométricas.* En el diagrama que provee BlueJ podemos observar que la clase *Circle, Rectangle y Triangle* utilizan métodos de la clase *Canvas.*

1. 1. El paquete ofrece 4 clases: Canvas, Circle, Rectangle, Triangle
   2. Static double Pi
      * makeVisible ()
      * makeInvisible ()
      * moveRight ()
      * moveLeft ()
      * moveUp ()
      * moveDown ()
      * moveHorizontal (int distance)
      * moveVertical (int distance)
      * slowMoveHorizontal (int distance)
      * slowMoveVertical (int distance)
      * changeSize (int newDiameter)
      * changeColor (String newColor)
   3. changeSize y changeColor
   4. Tiene 6 atributos:

* static double Pi
* int Diameter
* int xPosition
* int yPosition
* string Color
* boolean isVisible
  1. La forma del circulo esta descrita por dos atributos: Diameter y Pi
  2. Tiene 14 métodos
* makeVisible ()
* makeInvisible ()
* moveRight ()
* moveLeft ()
* moveUp ()
* moveDown ()
* moveHorizontal (int distance)
* moveVertical (int distance)
* slowMoveHorizontal (int distance)
* slowMoveVertical (int distance)
* changeSize (int newDiameter)
* changeColor (String newColor)
* draw ()
* erase ()
  1. Los objetos que se instancian a partir de la clase
  2. Los atributos y los métodos privados
  3. Porque los atributos y los métodos suelen dejar bastante claro sus propósitos
  4. Que se puede acceder a esta sin necesidad de instanciar un objeto de clase *circle.*
  5. Se utiliza la directiva *final.*

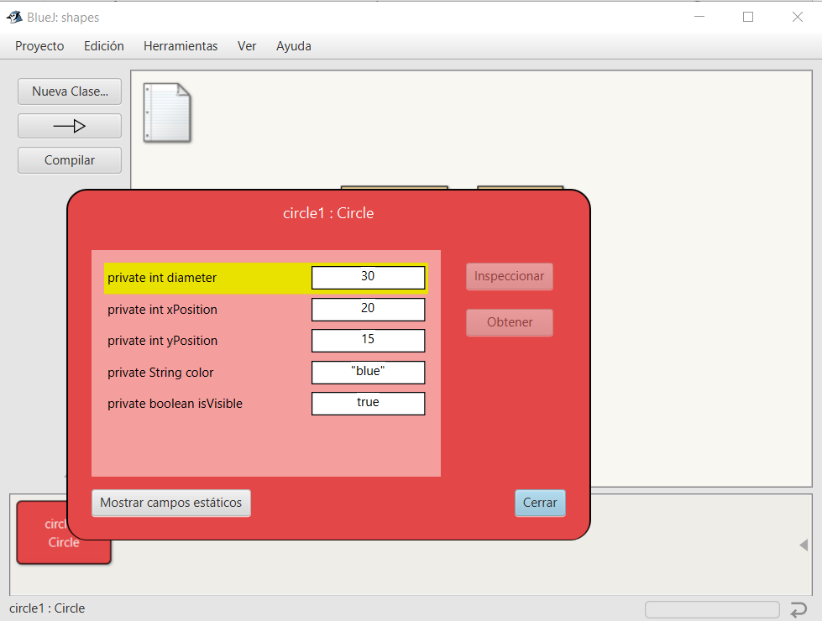


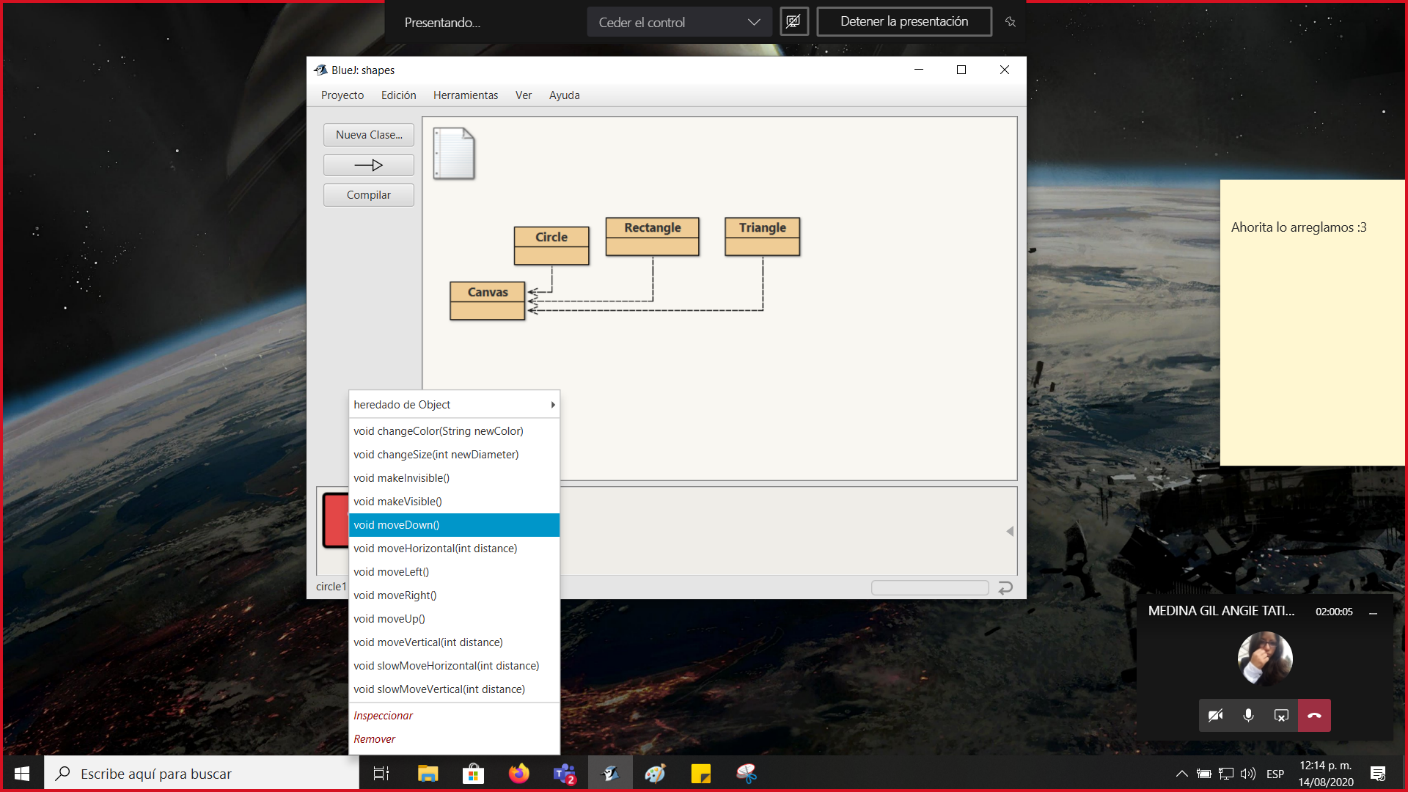
* 1. Que la variable es de tipo entero cuyo rango eta entre -216 y 216 -1
  2. Podría ser tipo *byte*, este es un tipo de dato primitivo, numérico entero con signo que se mueve en un rango de -128 a 127.
  3. Si se tienen diámetros mayores a 220000000, *diameter* como mínimo debería ser de tipo *int (-*231 y 231 -1*)*. Como restricciones, no tendría sentido que se le asigne a una figura un diámetro negativo.

1. Demostrar algunas características de objetos

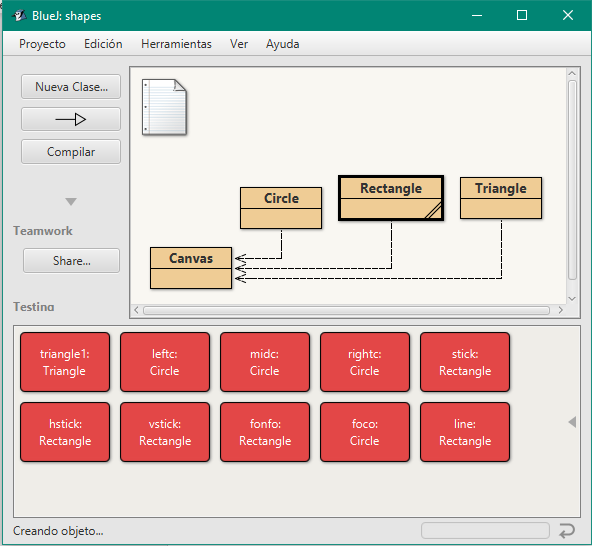
Manipulando objetos. Usando opciones.

* 1. Existen 4 clases: *Triangle, Square, Circle.*
  2. Se crearon 3 objetos: *Triangle1 = new Triangle (), Square1 = new Square (), Circle1 = new Circle ()*
  3. Porque la clase *canvas* no es instanciable.

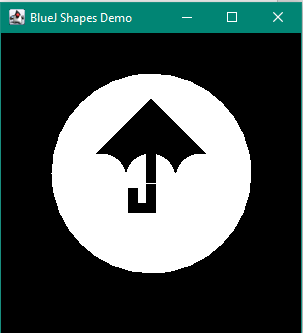
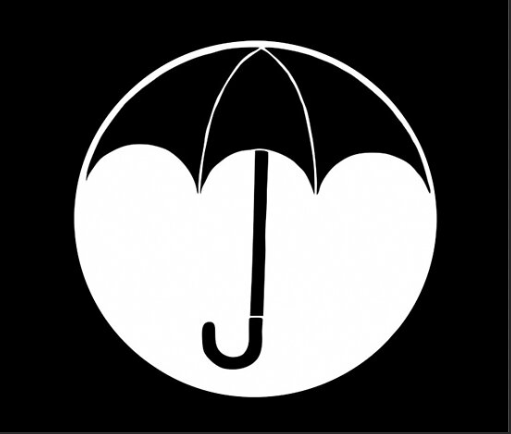




* 1. No aparecen los métodos privados ya que estos no son modificables para el programador
  2. Se usaron las tres clases de figuras (*Triangle, Square, Circle*) y el *canvas.*
  3. Se usaron 10 objetos

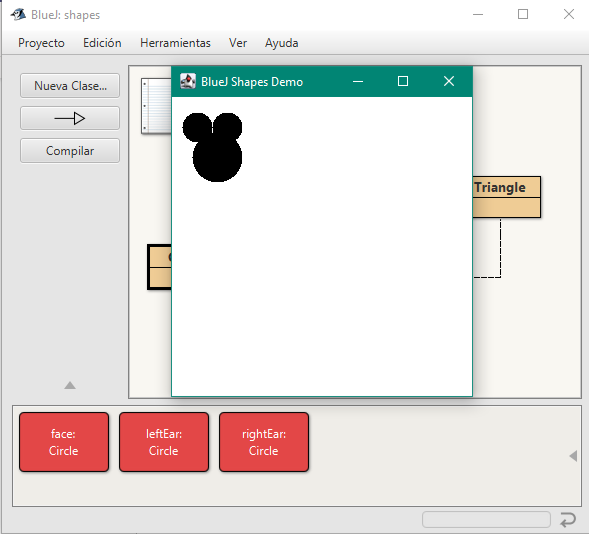


Versión BlueJ Original

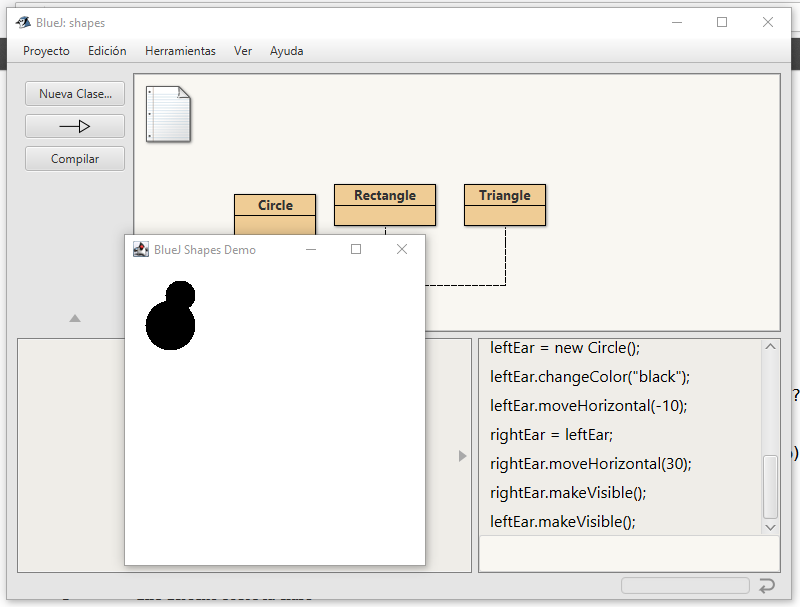
 

Manipulando objetos. Analizando y escribiendo código.

* 1. La silueta de la cabeza de Mickey Mouse

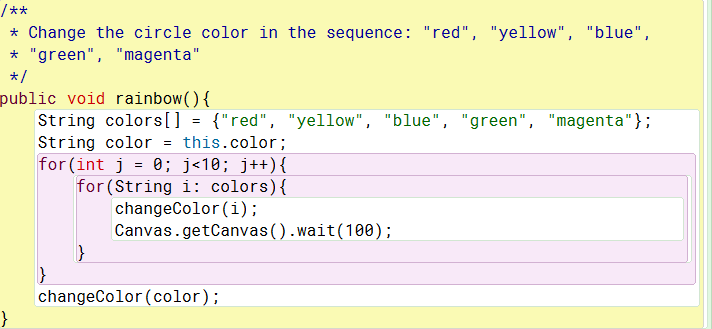


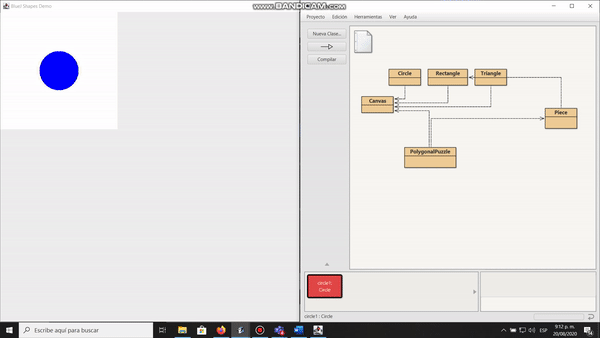
* 1. Existen 3 variables: *face, leftEar, rightEar*
  2. Existen 2 objetos: *face y leftEar*
  3. Cada objeto es de color negro
  4. Son visibles los dos objetos: *face* y *leftEar*, ya que *rightEar* no se instancio, se le fue asignado *leftEar*

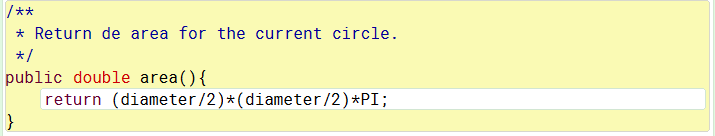


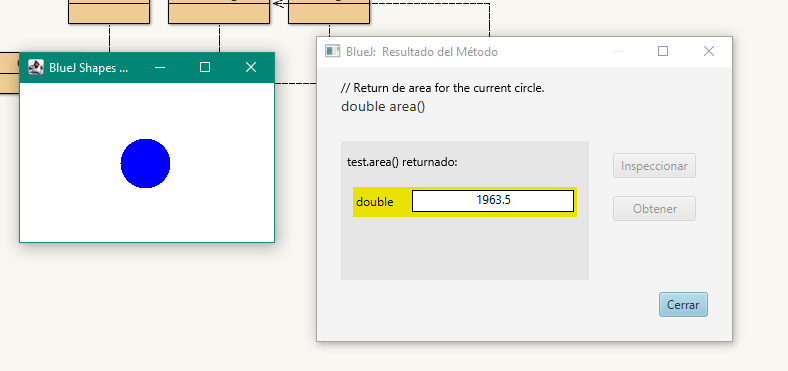
* 1. No son iguales
  2. Como se dijo en el punto anterior, la segunda figura tendría dos objetos visibles y en la figura 1 se ven 3 objetos.

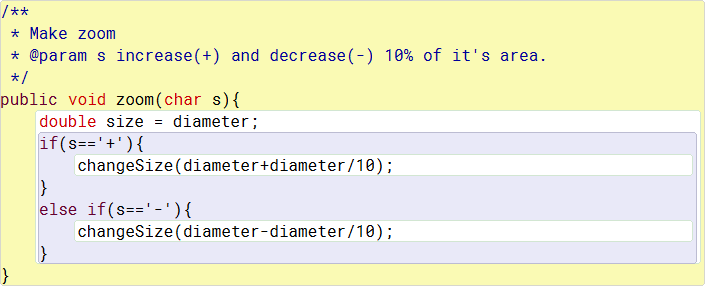
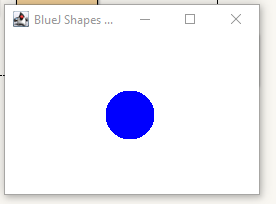
Extendiendo clases.

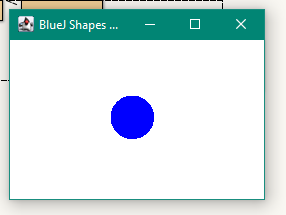
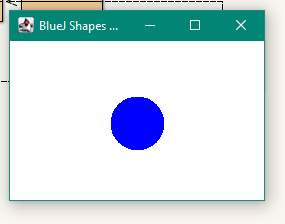


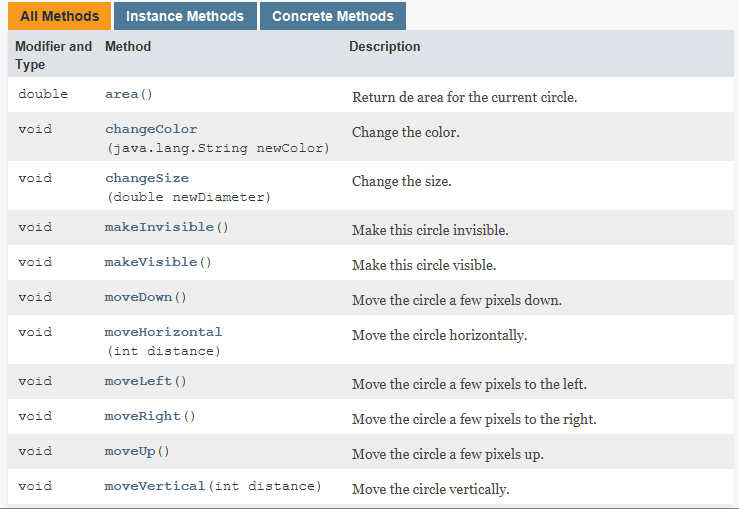


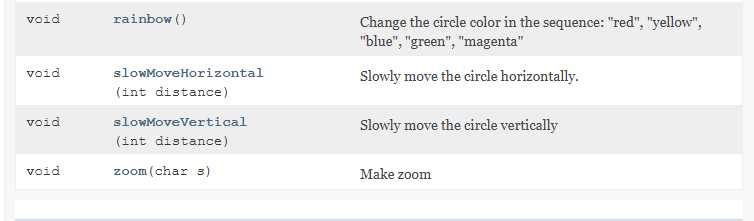










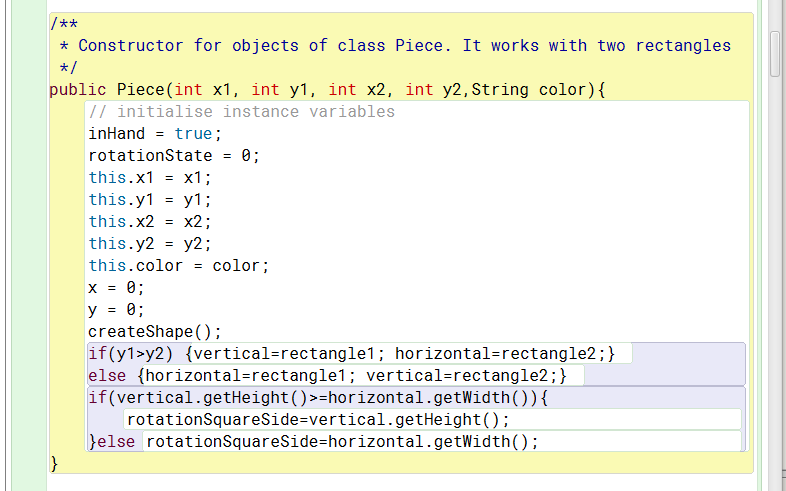
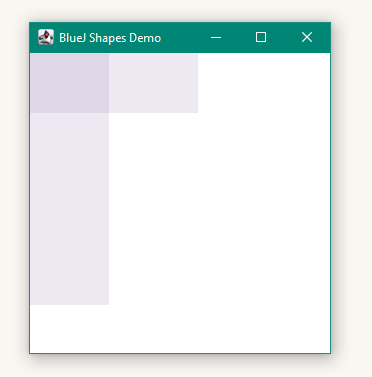
Implementando una nueva clase. Piece

1. * Constructores:
     + - \_(color: String, point: int[][])
   * Analizadores:
   * Modificadores:
     + - Take ()
       - Put ()
       - reflect (d: char)
       - rotate ()
       - resize(percentage:int)
       - translate (dx:int, dy:int)

* Mini-ciclo 1

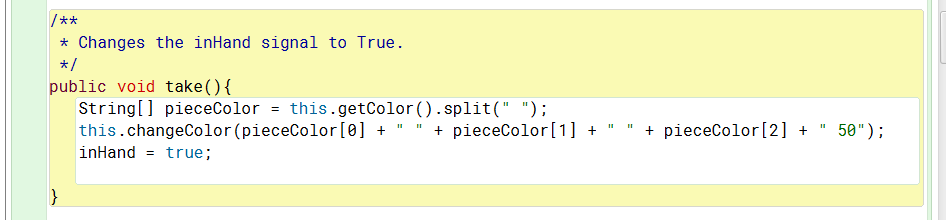
Se utilizó el constructor para crear una pieza con valores:

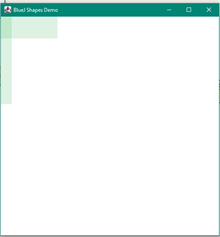
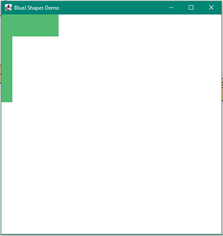
x1 = 79, y1 = 252, x2 = 168, y2 = 60, color = “162 137 190”

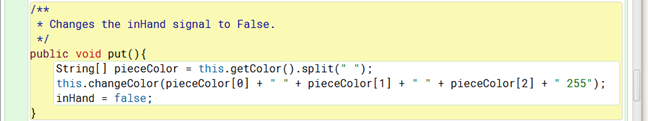
 

* Mini-ciclo 2

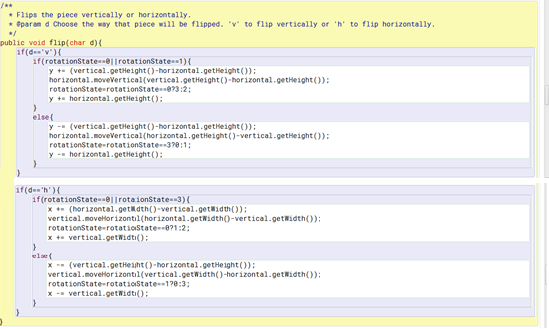
Se creo una pieza, como esta comienza en la mano podemos observar cierto nivel de transparencia y al ponerla vemos un color mucho más nítido.

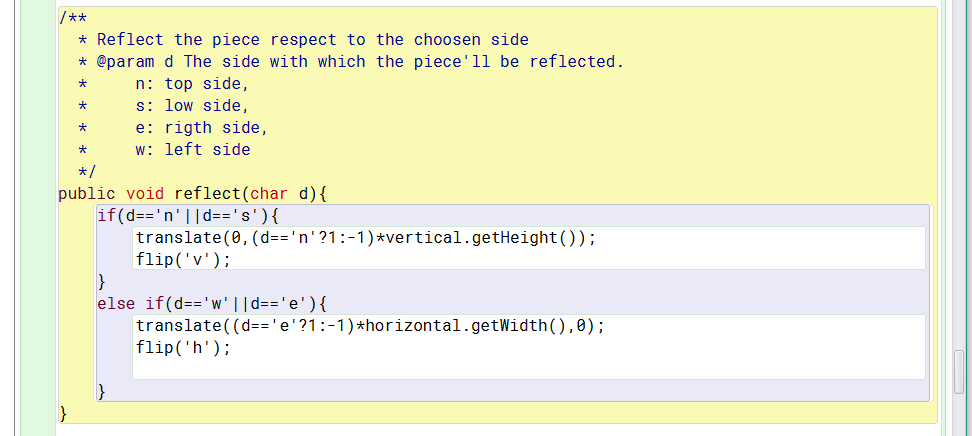


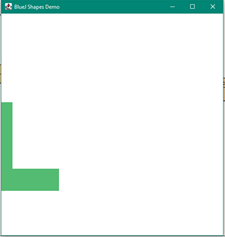
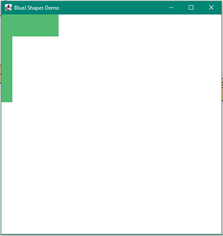


* Mini-ciclo 3
  + reflect (d: char)

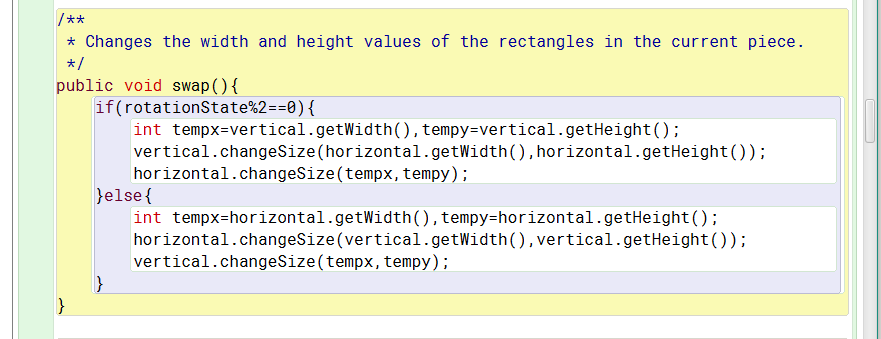


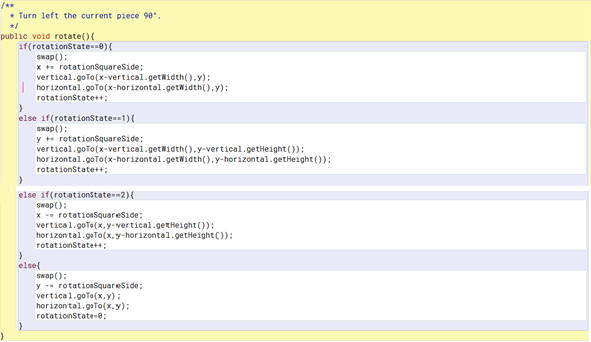


Reflejo sur Original

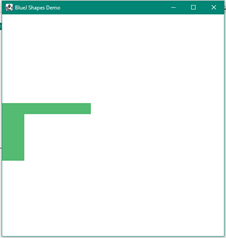
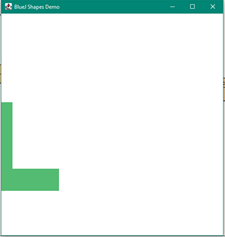
 

* + rotate ()



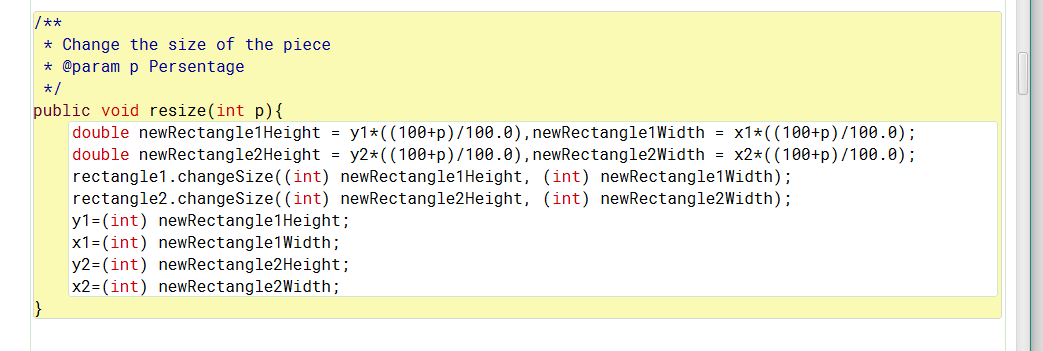


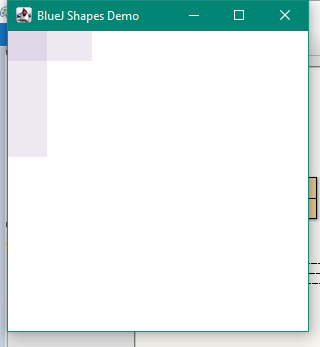
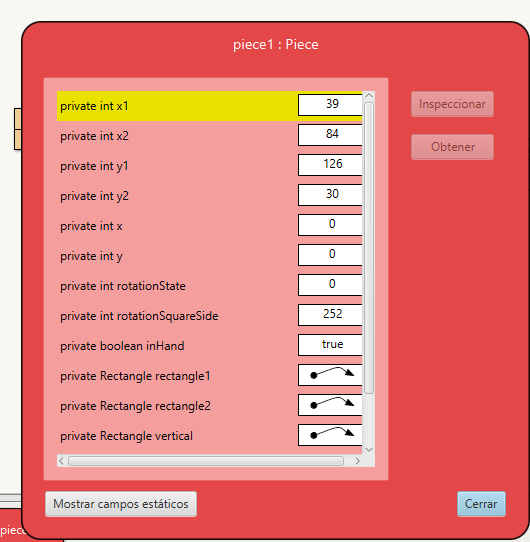
Rotación Original

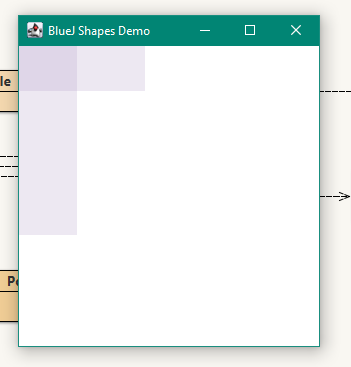
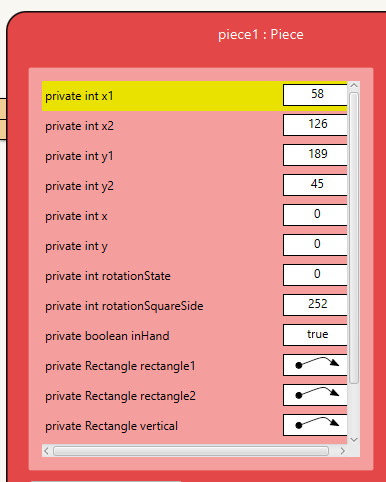
 

* Mini-ciclo 4
  + resize (percentage:int)

A la ficha se le aplico un aumento del 5%. Vemos en la tabla de valores que son los que esperábamos

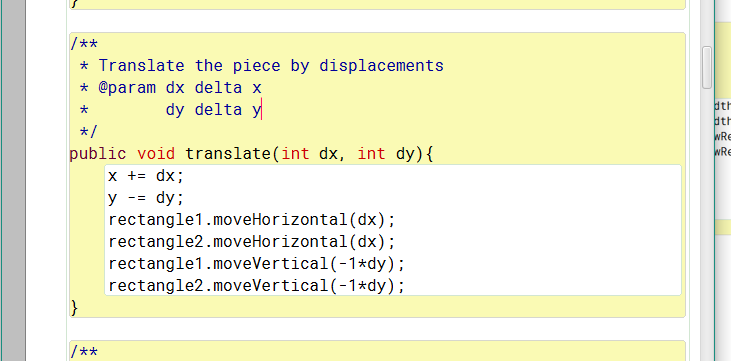


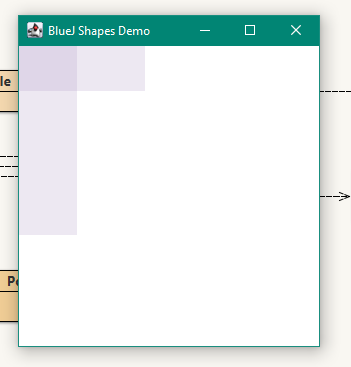
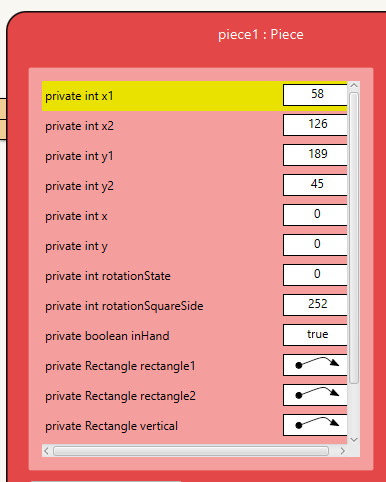
 

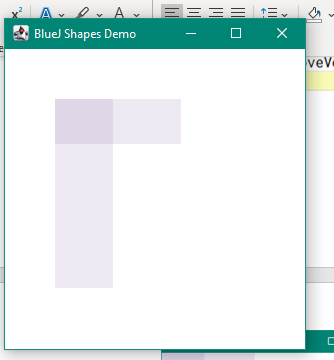
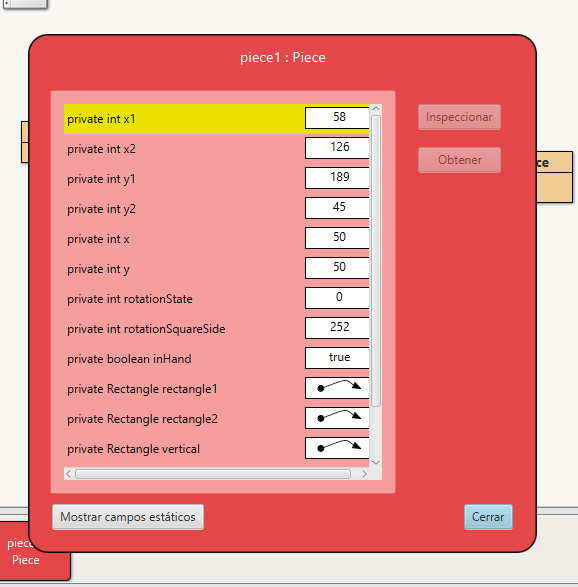
 

* + translate (dx: int, dy: int)

Se movio la ficha dx = 50 y dy = 50. Podemos ver tanto en las gráficas como en las tablas de datos de que se alteró correctamente

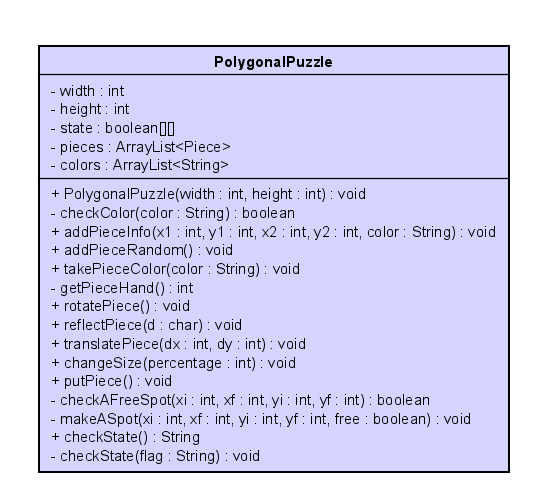


Definiendo y creando una nueva clase. Polygonal Puzzle

* Requisitos de interfaz



* 1. Mini-Ciclo 1: Constructor

Funciones:

* + - PolygonalPuzzle(int width, int height)

Mini-Ciclo 2: Checks

Descripción: Necesarios para las acciones de take y put, los métodos checks retornan condiciones necesarias para saber si el juego está terminado, si un color ya se encuentra en uso e incluso para poner una pieza en el tablero.

Funciones:

* + - checkColor(String color)
    - checkAFreeSpot(int xi, int xf, int yi, int yf)
    - checkState()

Mini-Ciclo 3: Adds

Descripción: Necesarias para la organización y manipulación de nuevas piezas, los métodos add permiten generar nuevas fichas ya sea de manera aleatoria o con características específicas.

Funciones:

* + - addPieceInfo(int x1, int y1, int x2, int y2, String color)
    - addPieceRandom()

Mini-Ciclo 4: Take, Put, Make

Descripción: Fundamentales para el proceso de completar el puzzle las funciones take y put permiten al usuario escoger o colocar una pieza respectivamente, mientras que makeASpot internamente controla las posiciones para generar colisiones

Funciones:

* + - takePieceColor(String color)
    - putPiece()
    - makeASpot(int xi, int xf, int yi, int yf, boolean free)

Mini-Ciclo 5: Gets

Descripción: Obtiene la pieza que se encuentra actualmente en la mano, esto es importante pues habilita las funciones de transformación

Funciones:

* + - getPieceHand()

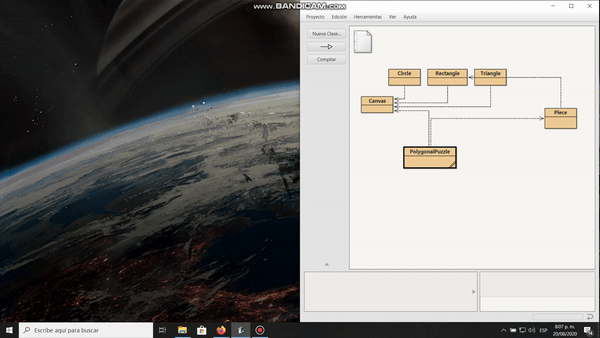
Mini-Ciclo 6: Transformation

Descripción: Estos métodos permiten transformar la pieza en mano a voluntad, ya sea cambiando su tamaño, rotando reflejando o moviéndola.

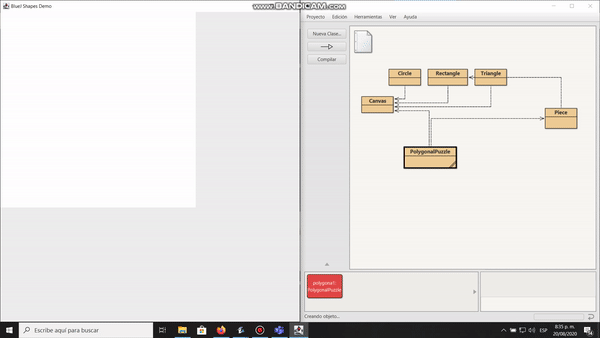
Funciones:

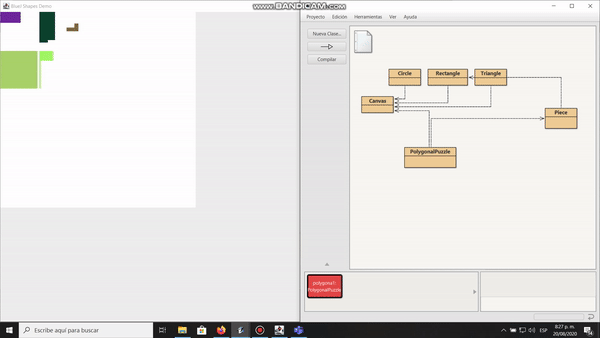
* + - rotatePiece()
    - reflectPiece(char d)
    - translatePiece(int dx, int dy)
    - changeSize(int percentage)
  1. Implementen la clase por mini-ciclos. Al final de cada mini-ciclo realicen una prueba indicando su propósito. Capturen las pantallas relevantes

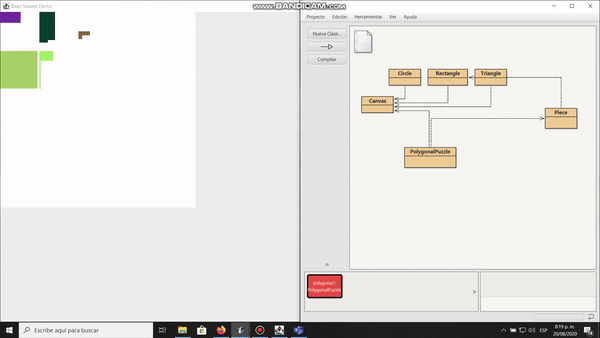
Mini-Ciclo 1: Constructor

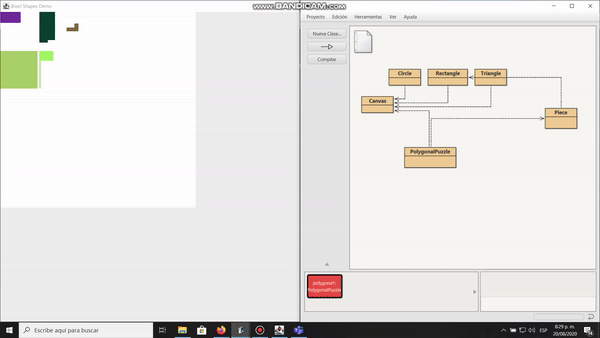


Mini-Ciclo 2: Checks

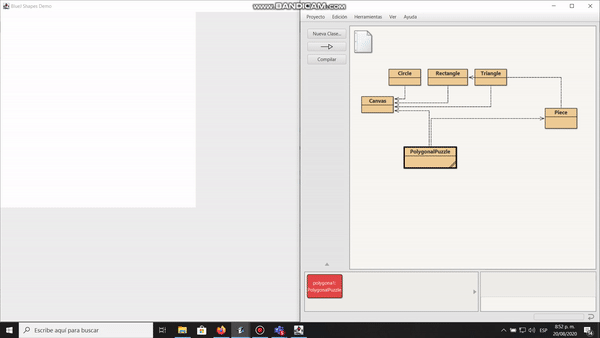


Mini-Ciclo 3: Adds



Mini-Ciclo 4: Take, Put, Make

Mini-Ciclo 6: Transformation



1. Se cumplieron todos los requisitos funcionales propuestos, en términos de mini-ciclos, todo fue realizado de manera satisfactoria.
2. Fueron necesarios los métodos getRectanglesPosition(), getRectanglesDimensions(), getState(), getColor(), getWidth(), getHeiht() pues los retornos de estos métodos son necesarios para calcular colisiones, activar métodos de PolygonalPuzzle y corroborar en la agregación de una nueva pieza.
3. Un keyListener para controlar todas las acciones vía teclado.

Extendiendo una clase. Polygonal Puzzle

Nuevos requisitos funcionales

1. Se genera una pieza que cumpla con el área que se da y se corre en el método que se creó en el siguiente punto, como este devuelve un booleano sabremos cuando se podrá parar de otra manera si no hay lugar donde poner ninguna pieza generada con esa área

Por otra parte, consideramos que los métodos creados son extensibles para que se trabaje con más rectángulos, en este caso al tener más rectángulos con los que trabajar se podría hacer un *floodfill* modificado sobre la matriz del tablero e ir contando el área.

1. Se recorre el tablero con un *for* anidado calculando que la celda en la que se encuentre no exceda los límites del tablero, cuando se encuentra una celda en la que es posible ubicar una parte de la pieza (es decir la celda tiene valor *true*) se intenta poner toda la ficha si esto no es posible en un principio se rota en la cuatro posibilidades y en dado caso de que todavía no funcione se traslada en un principio una unidad a la derecha se vuelve a repetir el proceso y si falta se traslada una posición abajo y se vuelve a hacer todo el proceso hasta que se pueda ubicar.

Retrospectiva

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

20 horas / Tatiana Medina – Jose Perez

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

El laboratorio está totalmente terminado, pero no nos sentimos satisfechos con el desarrollo del último punto pues nuestras soluciones no son eficaces temporalmente hablando

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

Nos resultó más útil la programación a pares, se podría decir que el 98% del laboratorio lo hicimos trabajando de esta manera ya que las ideas que nos surgen las complementamos, podemos evitar posibles errores futuros de mejor manera, etc.

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

Consideramos como mayor logro haber terminado el laboratorio, ya que era nuestro primer acercamiento a Java y la programación orientada a objetos

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

Los computadores y el internet a veces representaban problemas técnicos, nos tocaba esperar un tiempo para volver a trabajar o reiniciar los computadores.

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

Consideramos que trabajamos muy bien como equipo, nos comprometemos a implementar de mejor manera la otra practica XP vista.