



Proyecto Cupi2

## ISIS-1205 Algorítmica y Programación II

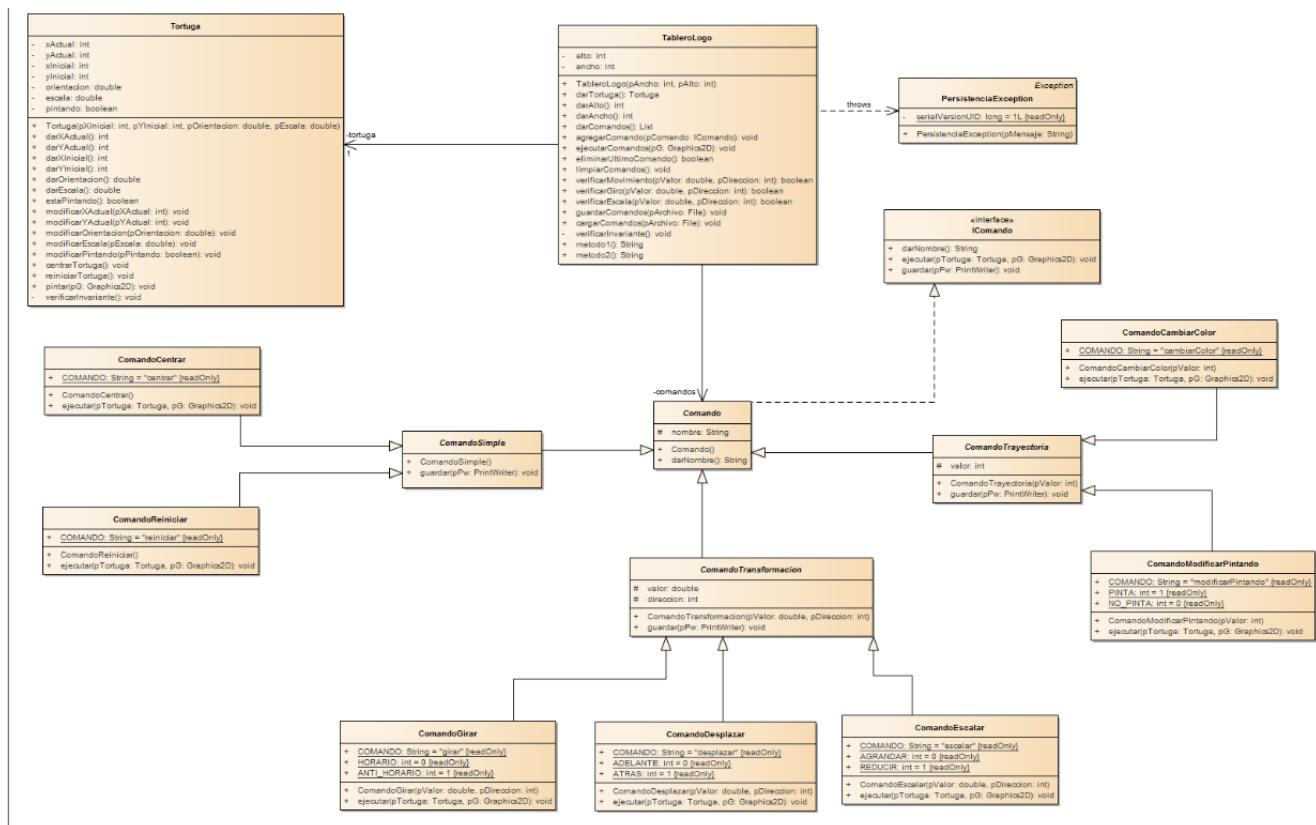
### Consideraciones adicionales de diseño

Ejercicio:	n10_cupiLogo
Autor:	Equipo Cupi2 2016
Semestre:	2016-1

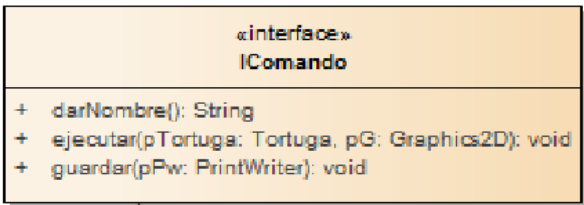
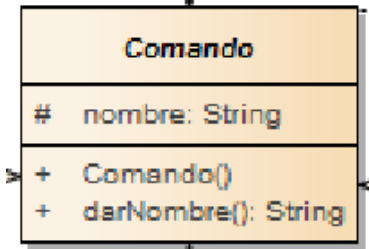
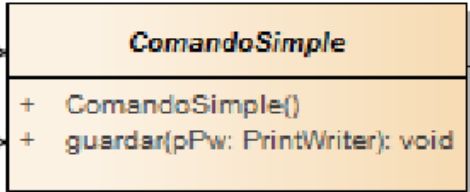
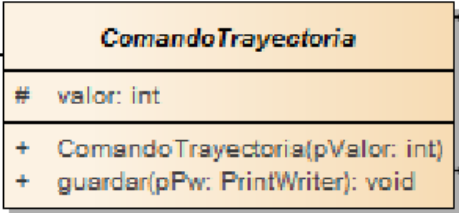
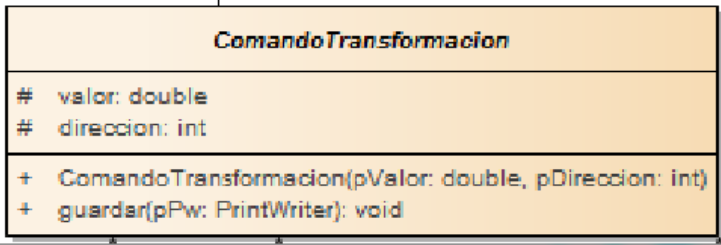
## Reutilización y desacoplamiento

La aplicación debe estar construida de manera que sea fácilmente extensible. Esto quiere decir que debe permitir agregar fácilmente nuevos tipos de comandos y nuevos requerimientos funcionales. Los elementos del modelo conceptual con los que se construye esta aplicación deben estar desacoplados entre sí y deben ser, en lo posible, reutilizables.

Considerando lo anterior, se propone el siguiente modelo del mundo:



En la siguiente tabla se describen algunas de las clases incluidas para que la aplicación cumpla con las características de desacoplamiento y reutilización esperadas:

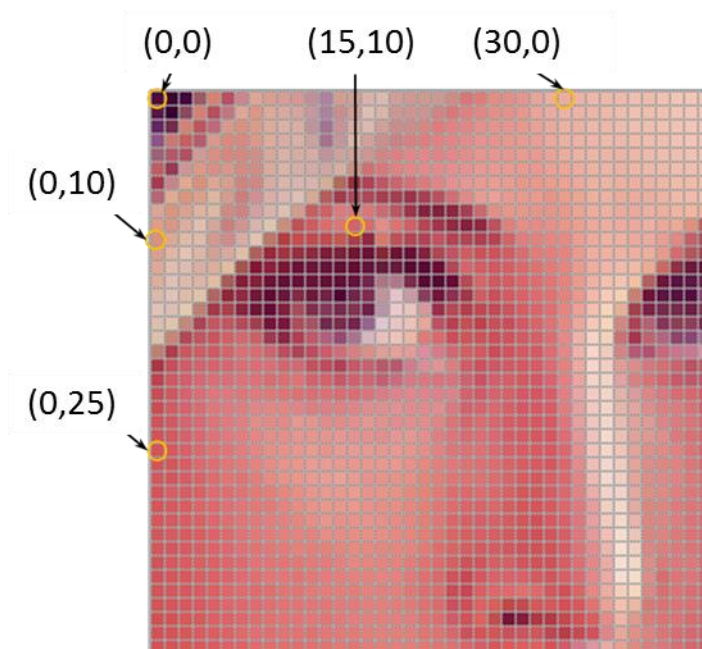
Clase	Descripción
 <pre> classDiagram     class IComando {         &lt;&lt;interface&gt;&gt;         + darNombre(): String         + ejecutar(pTortuga: Tortuga, pG: Graphics2D): void         + guardar(pPw: PrintWriter): void     }         </pre>	<p>IComando (Interface)</p> <p>Interface que define las funcionalidades que debe ofrecer cualquier tipo de comando.</p>
 <pre> classDiagram     class Comando {         # nombre: String         + Comando()         + darNombre(): String     }         </pre>	<p>Comando (Abstract)</p> <p>Clase abstracta que representa un comando. Define las propiedades (atributos y métodos) que comparten todos los comandos. Sirve como base para la implementación de las clases de todos los comandos. Esta clase se compromete con el contrato funcional de la interface <i>IComando</i>.</p>
 <pre> classDiagram     class ComandoSimple {         + ComandoSimple()         + guardar(pPw: PrintWriter): void     }         </pre>	<p>ComandoSimple (Abstract)</p> <p>Clase abstracta que representa un comando simple (comandos que no tienen parámetros). Define métodos que todo comando simple ejecuta de manera similar. Esta clase hereda de la clase Comando.</p>
 <pre> classDiagram     class ComandoTrayectoria {         # valor: int         + ComandoTrayectoria(pValor: int)         + guardar(pPw: PrintWriter): void     }         </pre>	<p>ComandoTrayectoria (Abstract)</p> <p>Clase abstracta que representa un comando que afecta la visualización de la trayectoria. Define las propiedades (atributos y métodos) que todo comando de trayectorias ejecuta de manera similar. Esta clase hereda de la clase Comando.</p>
 <pre> classDiagram     class ComandoTransformacion {         # valor: double         # direccion: int         + ComandoTransformacion(pValor: double, pDireccion: int)         + guardar(pPw: PrintWriter): void     }         </pre>	<p>ComandoTransformacion (Abstract)</p> <p>Clase abstracta que representa un comando que afecta las transformaciones de la tortuga. Define las propiedades (atributos y métodos) que todo comando de transformación ejecuta de manera similar. Esta clase hereda de la clase Comando.</p>

<div data-bbox="235 159 786 338"> <p><b>ComandoCentrar</b></p> <ul style="list-style-type: none"> <li>+ <u>COMANDO</u>: String = "centrar" {readOnly}</li> <li>+ ComandoCentrar()</li> <li>+ ejecutar(pTortuga: Tortuga, pG: Graphics2D): void</li> </ul> </div>	<p>ComandoCentrar</p> <p>Clase que extiende de ComandoSimple, se encarga de crear el comando que mueve la tortuga al centro del tablero. No requiere parámetros.</p>
<div data-bbox="256 384 764 546"> <p><b>ComandoReiniciar</b></p> <ul style="list-style-type: none"> <li>+ <u>COMANDO</u>: String = "reiniciar" {readOnly}</li> <li>+ ComandoReiniciar()</li> <li>+ ejecutar(pTortuga: Tortuga, pG: Graphics2D): void</li> </ul> </div>	<p>ComandoReiniciar</p> <p>Clase que extiende de ComandoSimple, se encarga de crear el comando que mueve la tortuga al centro del tablero y elimina la trayectoria pintada. No requiere parámetros.</p>
<div data-bbox="235 596 786 825"> <p><b>ComandoGirar</b></p> <ul style="list-style-type: none"> <li>+ <u>COMANDO</u>: String = "girar" {readOnly}</li> <li>+ <u>HORARIO</u>: int = 0 {readOnly}</li> <li>+ <u>ANTI_HORARIO</u>: int = 1 {readOnly}</li> <li>+ ComandoGirar(pValor: double, pDireccion: int)</li> <li>+ ejecutar(pTortuga: Tortuga, pG: Graphics2D): void</li> </ul> </div>	<p>ComandoGirar</p> <p>Clase que extiende de ComandoTransformacion, se encarga de cambiar la orientación de la tortuga con el valor y la dirección dadas por parámetro.</p>
<div data-bbox="235 867 786 1089"> <p><b>ComandoDesplazar</b></p> <ul style="list-style-type: none"> <li>+ <u>COMANDO</u>: String = "desplazar" {readOnly}</li> <li>+ <u>ADELANTE</u>: int = 0 {readOnly}</li> <li>+ <u>ATRAS</u>: int = 1 {readOnly}</li> <li>+ ComandoDesplazar(pValor: double, pDireccion: int)</li> <li>+ ejecutar(pTortuga: Tortuga, pG: Graphics2D): void</li> </ul> </div>	<p>ComandoDesplazar</p> <p>Clase que extiende de ComandoTransformacion, se encarga de cambiar la posición de la tortuga con el valor y la dirección dadas por parámetro.</p>
<div data-bbox="279 1134 742 1325"> <p><b>ComandoEscalar</b></p> <ul style="list-style-type: none"> <li>+ <u>COMANDO</u>: String = "escalar" {readOnly}</li> <li>+ <u>AGRANDAR</u>: int = 0 {readOnly}</li> <li>+ <u>REDUCIR</u>: int = 1 {readOnly}</li> <li>+ ComandoEscalar(pValor: double, pDireccion: int)</li> <li>+ ejecutar(pTortuga: Tortuga, pG: Graphics2D): void</li> </ul> </div>	<p>ComandoEscalar</p> <p>Clase que extiende de ComandoTransformacion, se encarga de cambiar el tamaño de la tortuga con el valor y la dirección (proporción) dadas por parámetro.</p>
<div data-bbox="224 1367 797 1549"> <p><b>ComandoCambiarColor</b></p> <ul style="list-style-type: none"> <li>+ <u>COMANDO</u>: String = "cambiarColor" {readOnly}</li> <li>+ ComandoCambiarColor(pValor: int)</li> <li>+ ejecutar(pTortuga: Tortuga, pG: Graphics2D): void</li> </ul> </div>	<p>ComandoCambiarColor</p> <p>Clase que extiende de ComandoTrayectoria, se encarga de cambiar el color de la trayectoria de la tortuga con el valor dado por parámetro.</p>
<div data-bbox="224 1589 797 1814"> <p><b>ComandoModificarPintando</b></p> <ul style="list-style-type: none"> <li>+ <u>COMANDO</u>: String = "modificarPintando" {readOnly}</li> <li>+ <u>PINTA</u>: int = 1 {readOnly}</li> <li>+ <u>NO_PINTA</u>: int = 0 {readOnly}</li> <li>+ ComandoModificarPintando(pValor: int)</li> <li>+ ejecutar(pTortuga: Tortuga, pG: Graphics2D): void</li> </ul> </div>	<p>ComandoModificarPintando</p> <p>Clase que extiende de ComandoTrayectoria, se encarga de cambiar el estado de pintando o no la trayectoria de la tortuga con el valor dado por parámetro.</p>

## Sistema de coordenadas

A continuación, se exponen varios detalles útiles para el desarrollo del proyecto:

- Las coordenadas de los símbolos gráficos del editor están dadas en píxeles.
- Java2D dibuja sobre una superficie de píxeles, cuya dimensión depende del panel en donde se esté dibujando.
- El origen de coordenadas se encuentra en la posición (0,0) que está situada en la esquina SUPERIOR IZQUIERDA de la superficie de dibujo (lienzo).
- La primera coordenada se refiere a la COLUMNA del píxel con el cual se está trabajando. Es creciente hacia la derecha.
- La segunda coordenada se refiere a la FILA del píxel con el cual se está trabajando. Es creciente HACIA ABAJO.
- Todo esto se ilustra en la siguiente figura:



Se debe tener esto en cuenta, pues en el código las variables de las coordenadas se llaman  $(x,y)$ , que “inducen” a pensar erróneamente en un plano cartesiano, que tiene el origen en la esquina inferior izquierda, donde la primera coordenada es creciente hacia la derecha y la segunda coordenada es creciente hacia arriba.

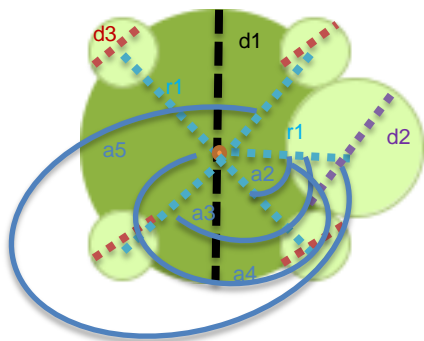
## Representación del Color

En Java2D es común representar el color utilizando el sistema RGB el cual utiliza 3 parámetros para representar un color: El rojo (Red), el verde (Green) y el azul (Blue). Sin embargo, el uso de 3 parámetros suele ser poco deseado en ciertos casos, por lo que en esos casos se utiliza un único parámetro en el modelo de color sRGB (Bits 24-31 representa Alfa –por defecto 11111111-, 16-23 representa Rojo, 8-15 representa Verde, 0-7 representa Azul).

## Sugerencias para pintar la tortuga de CupiLogo

La tortuga está compuesta por 6 círculos. 1 que representa el caparazón (con diámetro  $d1$ ), 1 que representa la cabeza (con diámetro  $d2$ ) y los otros 4 que representan las extremidades (con diámetro  $d3$ ).

Al cambiar la escala de la tortuga, estos diámetros se ven afectados por el valor “v” de la escala:  
Si la escala tiene el sentido de agrandar, el valor de escalar multiplica el diámetro. Si la escala tiene el sentido de reducir, el valor de escalar divide el diámetro.



<i>Variable (sin escalar)</i>	<i>Tamaño (píxeles)</i>
$d1$	40
$d2$	$\frac{d1}{2}=20$
$d3$	$\frac{d2}{2}=10$
$a1, a2, a3, a4, a5$	$0^\circ, 45^\circ, 135^\circ, 225^\circ, 315^\circ$

Para la orientación, se considera el ángulo  $0^\circ$  como el eje X positivo, hacia donde está mirando la tortuga. Para cada una de las partes de la tortuga se tienen las siguientes especificaciones:

- Caparazón: círculo de diámetro  $d1$ . El centro de este círculo corresponde a las coordenadas X y Y de la tortuga.
- Cabeza: círculo de diámetro  $d2$  con su centro ubicado a una distancia  $r1$  ( $d1/2$ ) del centro del caparazón de la tortuga con ángulo  $a1$  ( $0^\circ$ ).
- Pata 1 (Inferior derecha): círculo de diámetro  $d3$  con centro ubicado a una distancia  $r1$  ( $d1/2$ ) del centro del caparazón de la tortuga con un ángulo  $a2$  ( $45^\circ$ ).
- Pata 2 (Inferior izquierda): círculo de diámetro  $d3$  con centro ubicado a una distancia  $r1$  ( $d1/2$ ) del centro del caparazón de la tortuga con un ángulo  $a3$  ( $135^\circ$ ).
- Pata 3 (Superior izquierda): círculo de diámetro  $d3$  con centro ubicado a una distancia  $r1$  ( $d1/2$ ) del centro del caparazón de la tortuga con un ángulo  $a4$  ( $225^\circ$ ).
- Pata 4 (Superior derecha): círculo de diámetro  $d3$  con centro ubicado a una distancia  $r1$  ( $d1/2$ ) del centro del caparazón de la tortuga con un ángulo  $a5$  ( $315^\circ$ ).

Los colores de cada parte se muestran en el siguiente cuadro, los valores están representados usando el formato RGB.

<i>Color Caparazón</i>	142 , 179 , 65
<i>Color Cabeza y</i>	<i>Relleno:</i> 220 , 253 , 172
<i>Extremidades</i>	<i>Contorno:</i> 175 , 201 , 124