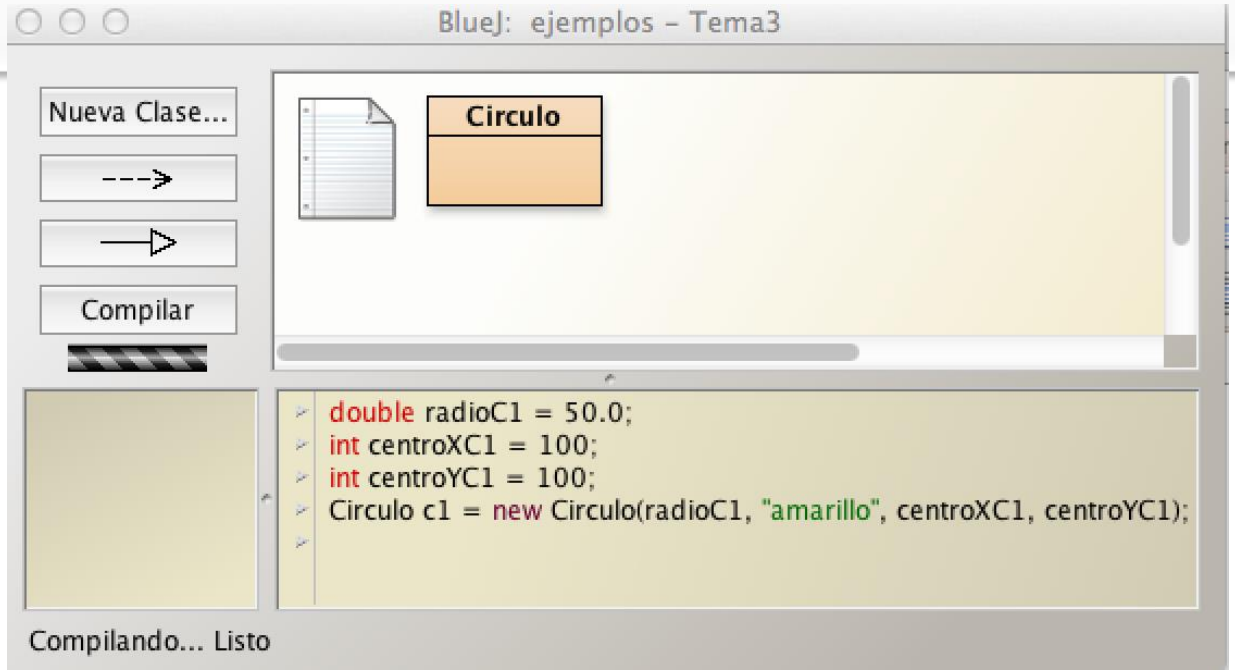
Del Punto 1: Memoria y tipo de variable

El **tipo** de una variable (Primitivo o Referencia) determina…

- el conjunto de **valores** que puede almacenar
- el conjunto de **operaciones** que se le aplican

Escribe en el **CodePad de Bluej** las siguientes instrucciones y, siguiendo las indicaciones del profes@r, responde: ¿cuál es el **estado de la memoria** tras su ejecución?

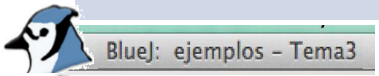
```
double radioC1 = 50.0;
int centroXC1 = 100, centroYC1 = 100;
Circulo c1 = new Circulo(radioC1, “amarillo”, centroXC1, centroYC1);
```



Del Punto 2 - Parte 1: Asignación

Inicialización según su tipo: valores para variables de tipo Primitivo

```
// Declaración e inicialización
// Inicialización a un valor del tipo o compatible, por asignación
char c = 'A';  // el valor asignado a c es un literal de tipo char
double d1 = 2; // el valor asignado a d1 es un literal int,
               // de tipo COMPATIBLE con double
double d2 = 3.0 + d1; // el valor asignado a d2 es el resultado de
                      // evaluar la expresión 3.0 + d1, i.e. 5.0
```



Copia las instrucciones anteriores –no los comentarios- en el ***CodePad*** de Bluej y luego "**traduce**" a Java los siguientes enunciados:

- **Mostrar** el **valor** de las variables c, d1 y d2 (en el *CodePad*)
- En la misma línea, **declarar** de tipo `int` **e inicializar** la variable d1Truncada a 2 **Mostrar** su **valor** ¿En qué se diferencia del mostrado para la variable d1? ¿Por qué?
- **Evaluar** (la expresión) `3.5 + d1`, para mostrar su valor y tipo
- En la misma línea, **declarar** de tipo `int` **e inicializar** la variable d2Truncada al valor `3.5 + d1`. **Mostrar** su **valor** ¿En qué se diferencia del mostrado para la variable d1? ¿Por qué?