PROBLEMA 4:
DIAGRAMAS DE ESTADOS

María Salas Urbano
msurbano@us.es

Compilación

- Compilar un lenguaje → Traducirlo a otro lenguaje (sin cambiar el significado de las sentencias).
  - Lenguaje fuente: expresivo y sin intérprete
  - Lenguaje destino: menos expresivo y dispone de intérprete

- Ejemplo de lenguaje compilado: Java
  - El lenguaje destino es un lenguaje ensamblador interpretable

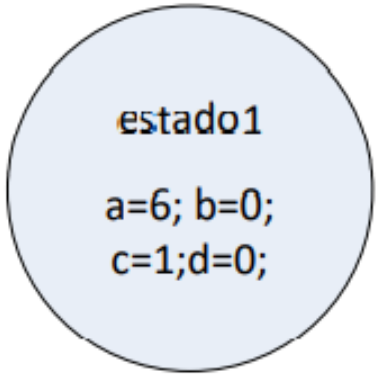
- En cualquier problema de compilación necesitamos: Criterio de corrección para decidir si es correcta la traducción

Ideas Decisión 1

- El diagrama de estado se traduce a un main() en Java.
- Hay que almacenar en memoria el diagrama // de estado y después generar el código.

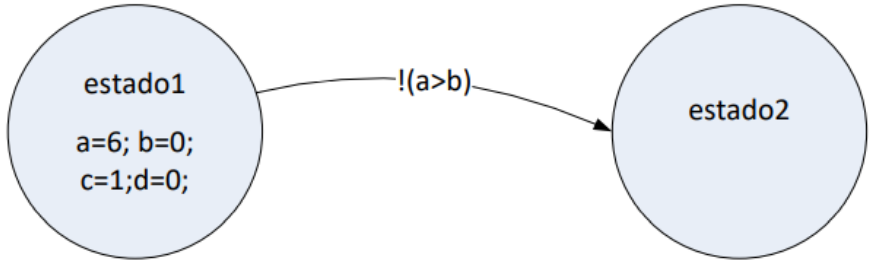
```
public class _Programa
{
    public static void
        main(String[] args)
    {
        //diagrama
    }
}
```

- El comienzo de main() coincidirá con la transición inicial del diagrama de estados.
- El estado se traduce como una secuencia de asignaciones Java:




```
a=6;
b=0;
c=1;
d=0;
```

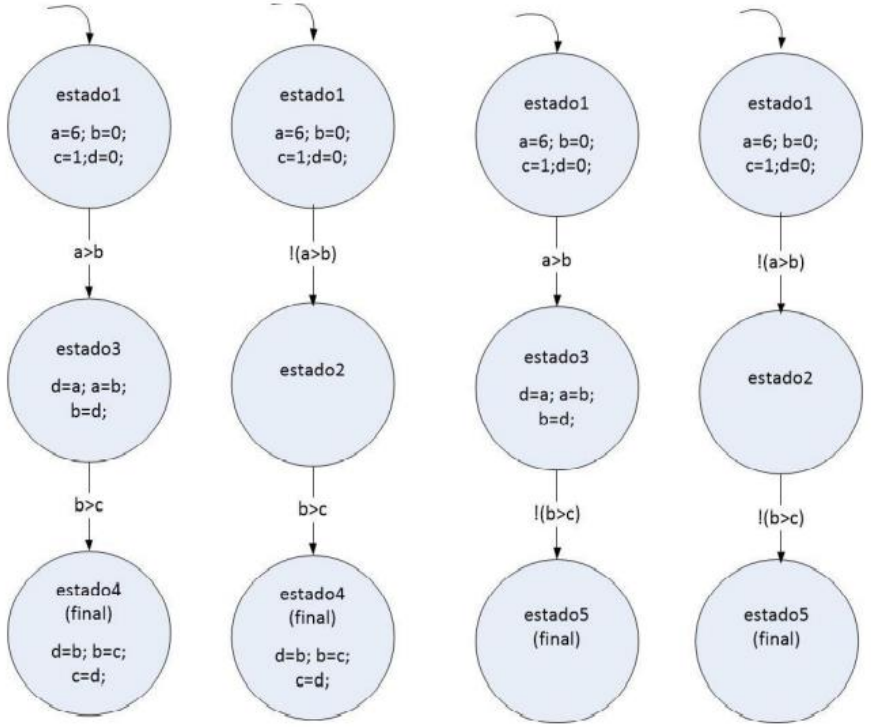
- Las transiciones se traduce a instrucciones condicionales. La condición de la transición se traduce como condición de la instrucción condicional



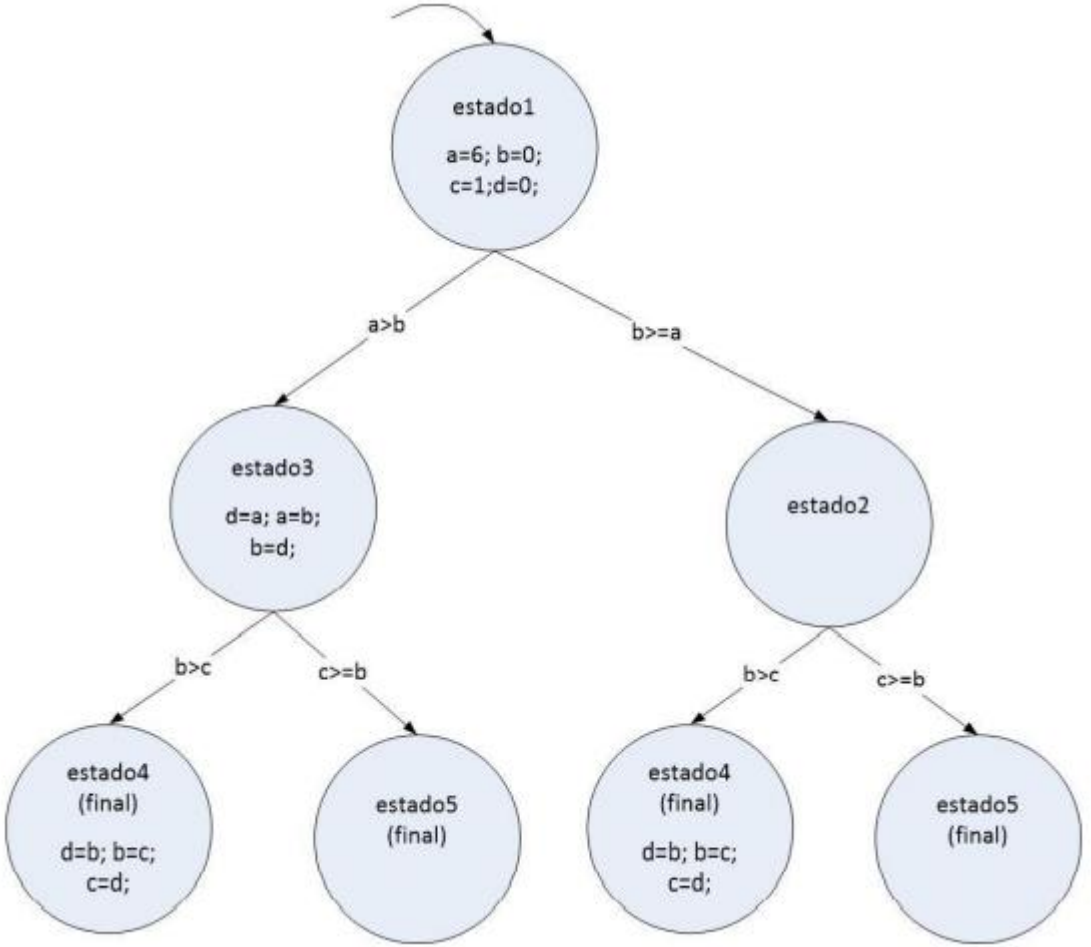

```
// estado1
if (!(a>b)){
     //estado2
}
```

Ideas Decisión 2

- Para generar código, el diagrama de estados debe transformarse en un conjunto de secuencias.
- Cada secuencia tendrá comienzo en el estado inicial y llegará hasta un estado final



- Las secuencias pueden organizarse de forma jerárquica aprovechando la existencia de estados comunes



Ideas Decisión 3

- Una vez organizadas las secuencias de forma jerárquica podemos traducirla a código secuencial haciendo uso de un centinela indicando si se ha alcanzado o no un estado final.

Resumen ideas

1. Decisión 1: El diagrama de estado se traduce al main() en Java.
   - Hay que almacenarlo en memoria y después generar el código.
   - La transición inicial es el comienzo del main().
   - Cada estado se traduce como una secuencia de asignaciones.
   - Las transiciones se traducen a instrucciones condicionales.
2. Decisión 2: El diagrama de estados debe transformarse en un conjunto de secuencias.
   - Cada secuencia comienza en el estado inicial y alcanza un estado final.
   - Las secuencias pueden organizarse de forma jerárquica.
3. Decisión 3: Utilizamos un centinela para indicar si se ha alcanzado o no un estado final.

1. Decisión 1: Memoria para almacenar el diagrama de estados
   - Memoria global para almacenar transiciones (se usará para generar las secuencias)
   - Memoria para almacenar estados (se usará para generar el código)
2. Decisión 2: Generar de forma recursiva las secuencias desde el contenido de la memoria transiciones.
3. Decisión 3:
   - El código del estado se genera consultado la memoria de estados.
   - El código de las transiciones se genera comprobando el valor del centinela y la condición de la transición.
   - El código de la condición se genera consultando la memoria transiciones y haciendo llamada recursiva.
   - Hay que memorizar el estado inicial del diagrama y las variables para generar el código.

```
generar_codigo_diagrama_desde(estado){
  escribir_codigo_estado(estado)
  si no hay ninguna transición desde(estado) entonces
     escribir(fin=true;)
  sino
     escribir(fin=false;)
     para cada transición a un estado_destino desde estado hacer
        escribir(if (fin==false && condición de la transición){)
        generar_codigo_diagrama_desde(estado_destino)
        escribir(})
     finpara
  finsi
}
```