# Sesión 3: Condiciones de carrera (*Race Conditions*)

## Concurrencia

Ángel Herranz

2019-2020

Universidad Politécnica de Madrid

# En capítulos anteriores. . .

## Concurrencia:

👍 Simultaneidad

\+

Sincronización + Comunicación

# Hoy: comunicación y sus problemas

## Concurrencia:

👍 Simultaneidad

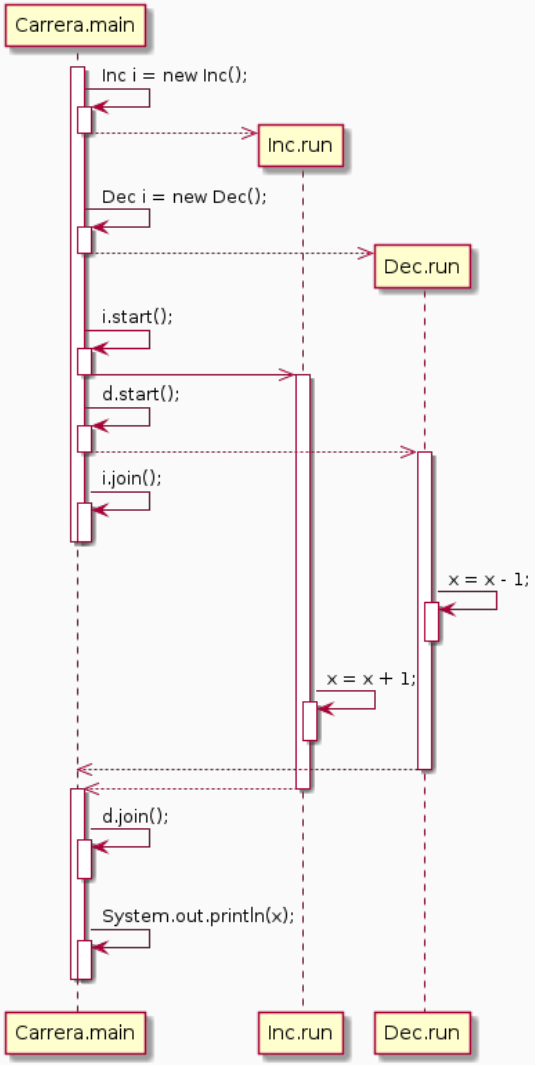
+

Sincronización + Comunicación

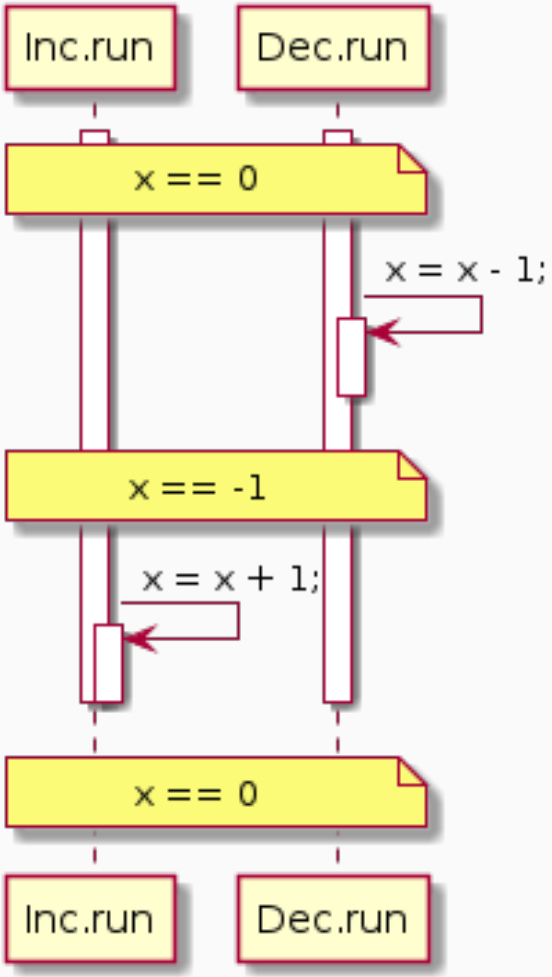
# ¡Ya podemos comunicar procesos!

```
public class Carrera {
  public static int x = 0;
  public class Inc
    extends Thread {
    public void run() {
      x = x + 1;
    }
  }
  public class Dec
    extends Thread {
    public void run() {
      x = x - 1;
    }
  }
  public static
    void main(String[] a)
    throws Exception {
    Inc i = new Inc();
    Dec d = new Dec();
    i.start();
    d.start();
    i.join();
    d.join();
    System.out.println(x);
  }
}
```

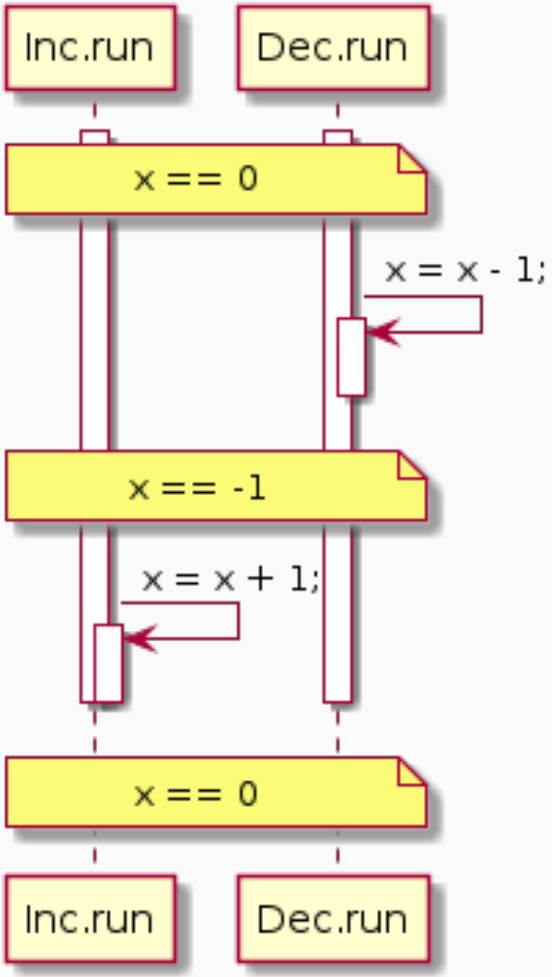
# Ejecutar paso a paso

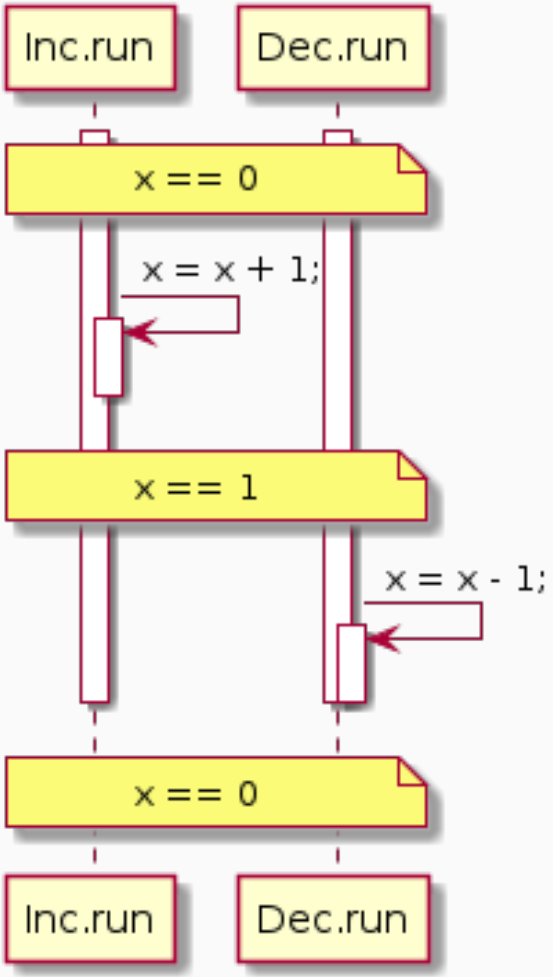


# Semántica: todos los *entrelazados*



# Semántica: todos los *entrelazados*

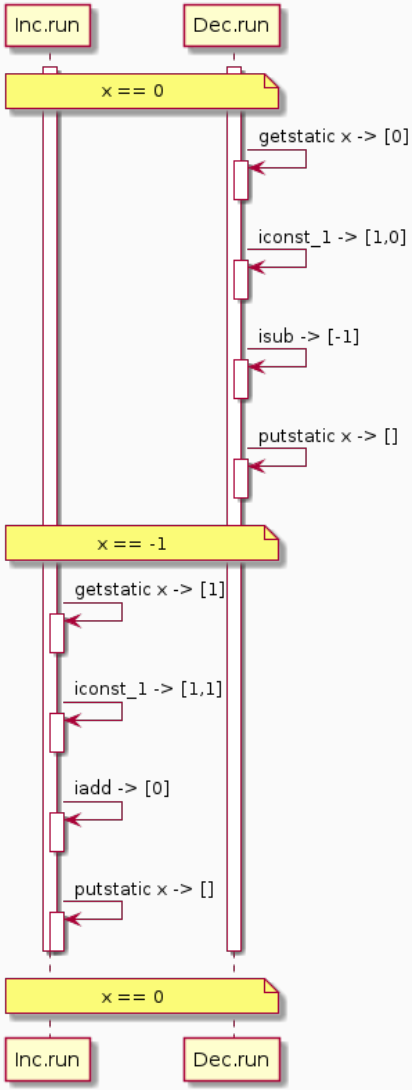


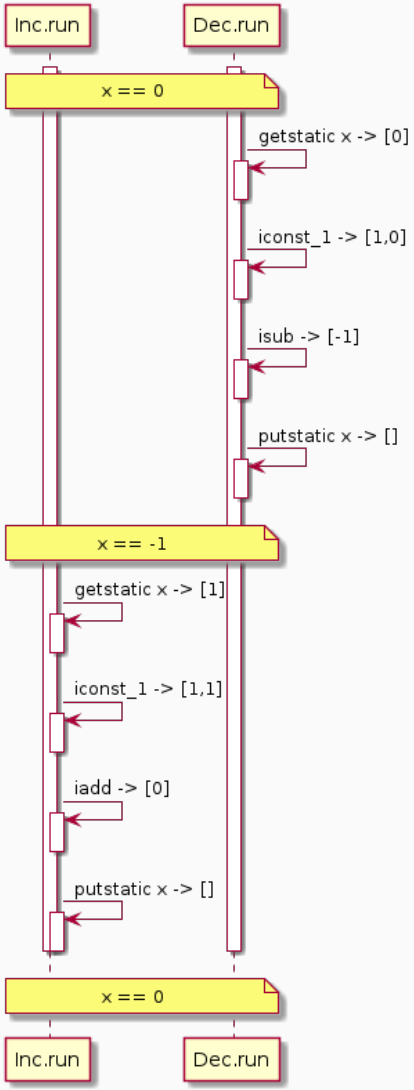
# Acciones atómicas

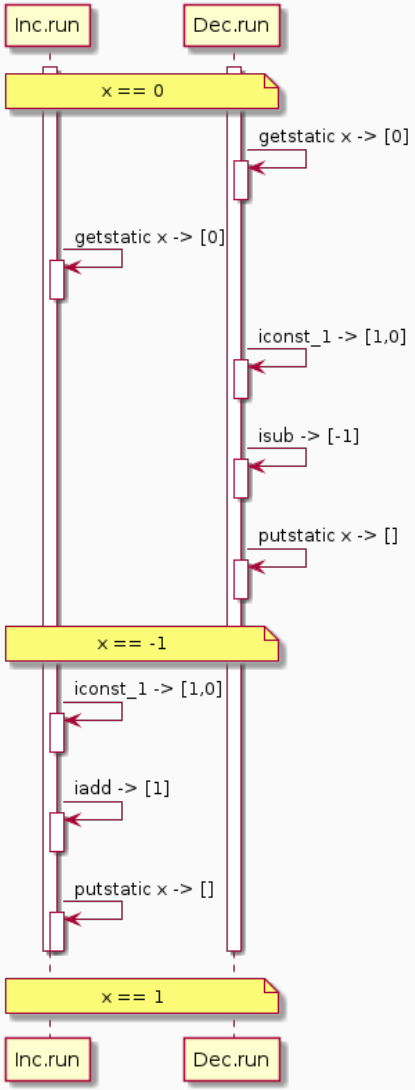
```
$ javap -c Carrera$Inc
public class Carrera$Inc {
  ...
  public void run();
    Code:
       0: getstatic     #2                  // Field x:I
       3: iconst_1
       4: iadd
       5: putstatic     #2                  // Field x:I
       8: return
  ...
}
```

# Entrelazados de acciones atómicas

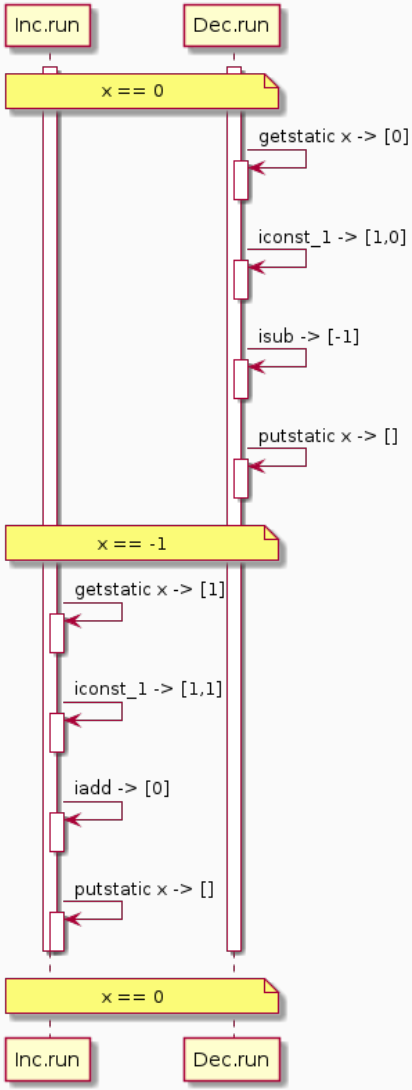


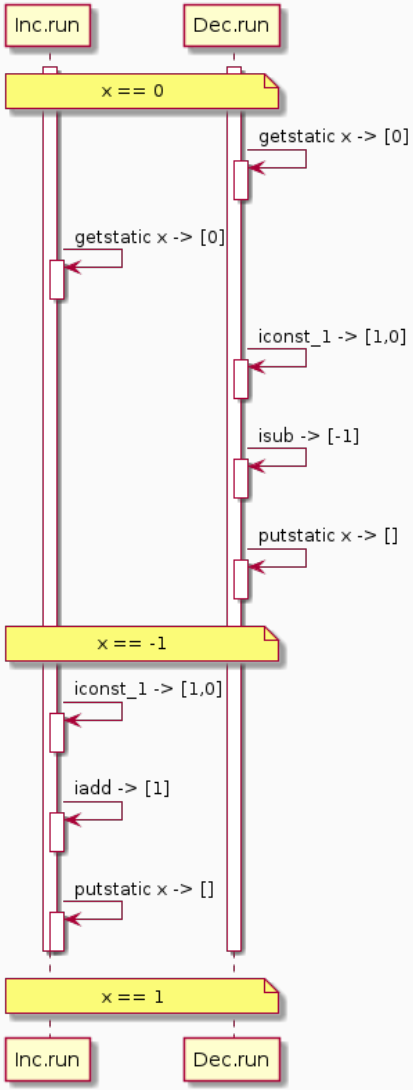
# Entrelazados de acciones atómicas

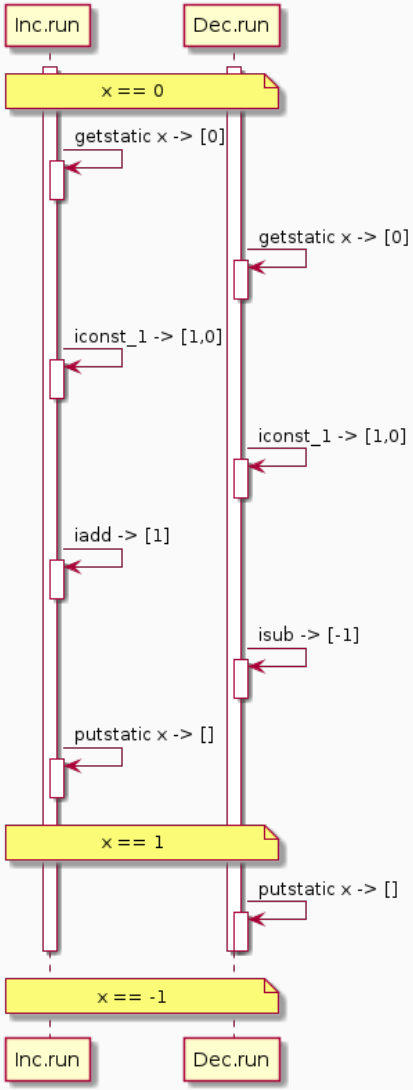


# Entrelazados de acciones atómicas



# Concepto: *condición de carrera*

Resultados indeseados por interacción de dos o más procesos que leen y modifican datos compartidos

=

condición de carrera

# Ejemplos de condiciones de carrera ii

*Singleton* accesible con `Database.getInstance()`

```
public class Database {
  private static Database instance;

  public static getInstance() {
    if (instance == null)
      instance = new Database();
    return instance;
  }
  ...
}
```

# Ejemplos de condiciones de carrera iii

Evitar *machacar* una entrada de una tabla

```
if (!map.contains(key)) {
  map.put(key, value)
}
```

# Ejemplos de condiciones de carrera iv

Código C para un microcontrolador de 8 bits

```
x = 271;   ||   x = 1;
```

¿Valores posibles para x?

# Ejemplos de condiciones de carrera iv

Código C para un microcontrolador de 8 bits

```
x = 271;   ||   x = 1;
```

¿Valores posibles para x?

271, 1

# Ejemplos de condiciones de carrera iv

Código C para un microcontrolador de 8 bits

```
x = 271;   ||   x = 1;
```

¿Valores posibles para x?

271, 1, 257, 15

¿Por qué?

# Ejemplos de condiciones de carrera iv

Código C para un microcontrolador de 8 bits

```
x = 271;   ||   x = 1;
```

Acciones atómicas:

```
LDI XH 0x01  ||  LDI XH 0x00
LDI XL 0x0f  ||  LDI XL 0x01
```

¿Valores posibles para x?

271, 1, 257, 15

¿Por qué?

# Ejemplos de condiciones de carrera v

Código ejecutando en CPU con caché

```
public static int x = 0;
public static int y = 0;
```

| | |
|---|---|
| `x = 1;` | `y = 1;` |
| `System.out.print(y);` | `System.out.printl(x);` |

¿Salidas posibles?

# Ejemplos de condiciones de carrera v

Código ejecutando en CPU con caché

```
public static int x = 0;
public static int y = 0;
```

| | |
|---|---|
| `x = 1;` | `y = 1;` |
| `System.out.print(y);` | `System.out.printl(x);` |

¿Salidas posibles?

`11`, `01`, `10`

# Ejemplos de condiciones de carrera v

Código ejecutando en CPU con caché

```
public static int x = 0;
public static int y = 0;
```

| | |
|---|---|
| `x = 1;` | `y = 1;` |
| `System.out.print(y);` | `System.out.printl(x);` |

¿Salidas posibles?

`11`, `01`, `10`, `00`

¿Por qué?

# Caché

Los compiladores y *RTEs* intentan optimizar los tiempos de acceso a los datos realizando una copia temporal de variables en la caché del procesador

# volatile

Asegurar que una variable es compartida de verdad

```
volatile public static int x = 0;
volatile public static int y = 0;
```

| | |
|---|---|
| `x = 1;` | `y = 1;` |
| `System.out.print(y);` | `System.out.printl(x);` |

¿Salidas posibles ahora?

# volatile

Asegurar que una variable es compartida de verdad

```
volatile public static int x = 0;
volatile public static int y = 0;
```

| | |
|---|---|
| `x = 1;` | `y = 1;` |
| `System.out.print(y);` | `System.out.printl(x);` |

¿Salidas posibles ahora?

11, 01, 10, ~~00~~

# Identificar a los *sospechosos habituales*

- Variables **`public static`** que varios procesos leen y modifican

# Identificar a los *sospechosos habituales*

- Variables **`public static`** que varios procesos leen y modifican
- Varios procesos reciben una referencia al mismo objeto que en ejecución leen y modifican

```
Dato d = new Dato();
Proceso p1 = new Proceso(d);
Proceso p2 = new Proceso(d);
p1.start();
p2.start();
```

# Concepto: *sección crítica*

Porción de código que puede dar lugar a una condición de carrera

=

sección crítica

# Secciones críticas en el primer ejemplo

```
public static int x = 0;
```

```
public class Inc
  extends Thread {
  public void run() {
    x = x + 1;
  }
}
```

```
public class Dec
  extends Thread {
  public void run() {
    x = x - 1;
  }
}
```

# Ejercicio obligatorio semanal

Hoja de ejerccios en:

`http://babel.ls.fi.upm.es/teaching/concurrencia`

Ejercicio 2:

**Provocar una condición de carrera**

Fichero a entregar:

`CC_02_Carrera.java`

Sistema de entrega:

`http://vps142.cesvima.upm.es`