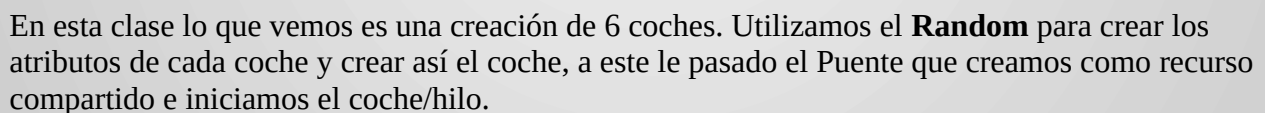
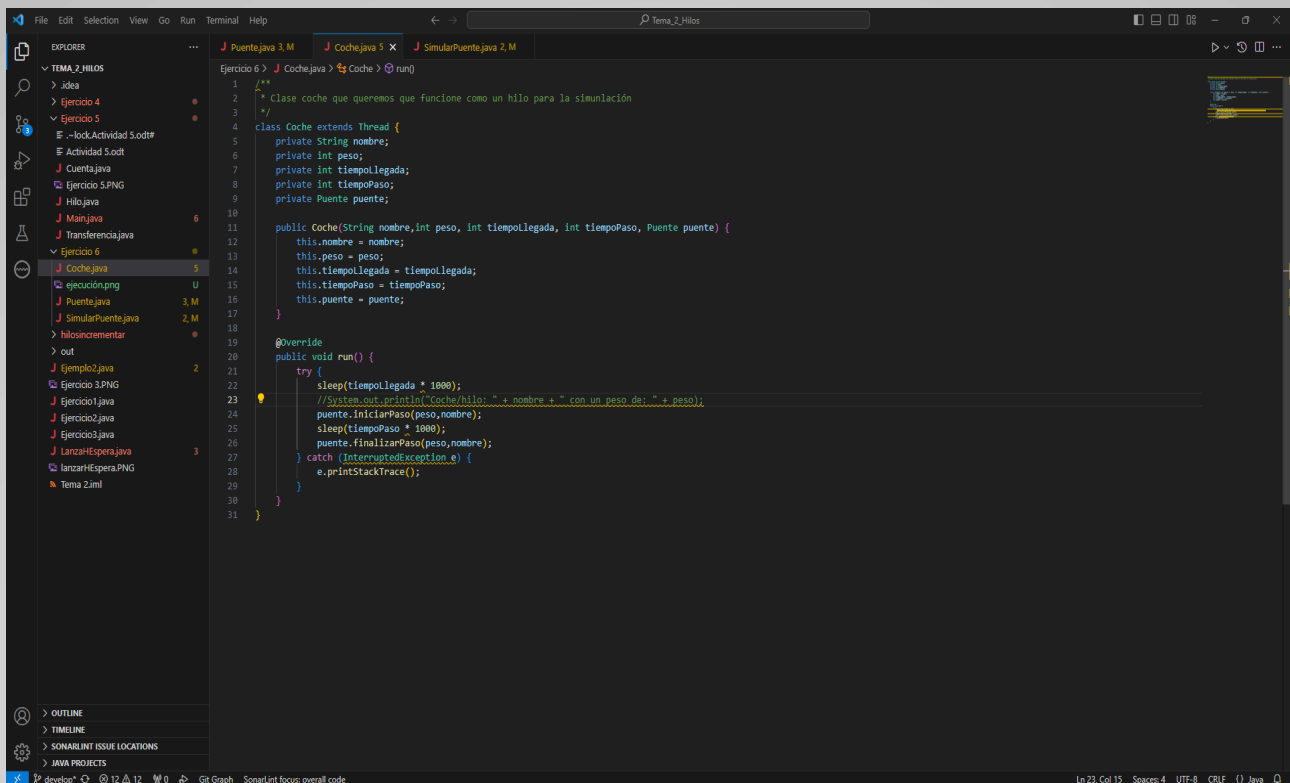


## Captura de la ejecución:



Clase Coche.java, que es modelada como hilos:



```
1  /**
2   * Clase coche que queremos que funcione como un hilo para la simulación
3   */
4  class Coche extends Thread {
5      private String nombre;
6      private int peso;
7      private int tiempoLlegada;
8      private int tiempoPaso;
9      private Puente puente;
10
11      public Coche(String nombre, int peso, int tiempoLlegada, int tiempoPaso, Puente puente) {
12          this.nombre = nombre;
13          this.peso = peso;
14          this.tiempoLlegada = tiempoLlegada;
15          this.tiempoPaso = tiempoPaso;
16          this.puente = puente;
17      }
18
19      @Override
20      public void run() {
21          try {
22              sleep(tiempoLlegada * 1000);
23              //System.out.println("Coche/hilo: " + nombre + " con un peso de: " + peso);
24              puente.iniciarPaso(peso, nombre);
25              sleep(tiempoPaso * 1000);
26              puente.finalizarPaso(peso, nombre);
27          } catch (InterruptedException e) {
28              e.printStackTrace();
29          }
30      }
31  }
```

En esta clase lo que hacemos llamar a los método del recurso compartido (**Puente**) y ejecutarlos.

Hacemos dos esperas para cada hilo según su tiempo de llegada para iniciar el paso y el tiempo que tarda en pasar para finalizar su paso.

## Clase Puente.java:

```

1  class Puente {
2      private static final int MAX_COCHES = 3;
3      private static final int PESO_MAXIMO = 5000;
4      private int cochesEnPonte = 0;
5      private int pesoEnPonte = 0;
6
7      /**
8       * Método para saber si puede pasar el coche
9       * @param peso El peso del coche para saber si puede pasar
10      * @return boolean
11      */
12      public synchronized boolean sePermitePaso(int peso) {
13          return cochesEnPonte < MAX_COCHES && (pesoEnPonte + peso) <= PESO_MAXIMO;
14      }
15
16      /**
17       * Método para modificar el estado del puente una vez haya salido un coche
18       * @param peso El peso del coche
19      */
20      public synchronized void finalizarPaso(int peso, String nombre) {
21          cochesEnPonte--;
22          pesoEnPonte -= peso;
23          System.out.println(nombre + " con un peso de: " + peso + " salió del puente.\n Coches en el puente: " + cochesEnPonte + ", Peso en el puente: " + pesoEnPonte);
24          notifyAll();
25      }
26
27      /**
28       * Método que empieza el paso del coche si este tienen el paso permitido
29       * @param peso Peso del coche
30       * @throws InterruptedException
31      */
32      public synchronized void iniciarPaso(int peso, String nombre) throws InterruptedException {
33          while (!sePermitePaso(peso)) {
34              wait();
35          }
36          cochesEnPonte++;
37          pesoEnPonte += peso;
38          System.out.println(nombre + " con un peso de: " + peso + " entró en el puente.\n Coches en el puente: " + cochesEnPonte + ", Peso en el puente: " + pesoEnPonte);
39      }
40  }

```

En esta clase vemos los métodos que utilizamos en la clase **Coche** y uno más, que se utiliza en el *iniciarPaso*, que es *sePermitePaso*: este método comprueba si el peso del coche que quiere entrar, más si hay algún otro dentro del puente, supera el peso máximo del puente, este método lo llamamos en el *iniciarPaso*, ya que si devuelve false no podría pasar al superar el peso máximo del puente y no iniciaría el paso del coche por el puente y hace una espera hasta que salga un coche y pueda este entrar.

Por último, método *finalizarPaso* lo que hace es finalizar el paso de un coche por el puente y lo notifica a todo (*notifyAll*) para así poder realizar el proceso con otro coche.