BOLETÍN 4 HILOS

1. Escribe una clase llamada Relevos que simule una carrera de relevos de la siguiente forma:

- Cree 4 threads, que se quedarán a la espera de recibir alguna señal para comenzar a correr. Una vez creados los threads, se indicará que comience la carrera, con lo que uno de los threads deberá empezar a correr.
- Cuando un thread termina de correr pone algún mensaje en pantalla y espera un par de segundos, pasando el testigo a otro de los hilos para que comience a correr, y terminando su ejecución (la suya propia).
- Cuando el último thread termine de correr, el programa principal mostrará un mensaje indicando que todos los hijos han terminado.

Ejemplo de ejecución:

```
Todos los hilos creados.
Doy la salida!
Soy el thread 1, corriendo . . .
Terminé. Paso el testigo al hijo 2
Soy el thread 2, corriendo . . .
Terminé. Paso el testigo al hijo 3
Soy el thread 3, corriendo . . .
Terminé. Paso el testigo al hijo 4
Soy el thread 4, corriendo . . .
Terminé!
Todos los hilos terminaron.
```

2. Escribe una clase llamada SuperMarket que implemente el funcionamiento de N cajas de un supermercado.
    - Los M clientes del supermercado estarán un tiempo aleatorio comprando y con posterioridad seleccionarán de forma aleatoria en qué caja posicionarse para situarse en su cola correspondiente.
    - Cuando les toque el turno serán atendidos, procediendo al pago correspondiente e ingresando en la variable Resultados del supermercado.

    Se deben crear tantos threads como clientes haya y los parámetros M y N se deben leer por entrada estándar. Para simplificar la implementación, el valor de pago de cada cliente puede ser aleatorio en el momento de su pago.

3. Escribe una clase llamada Parking que solicite el número de plazas del parking y el número de coches existentes en el sistema. Se deben crear tantos threads como coches haya.
    - El parking dispondrá de una única entrada y una única salida.
    - En la entrada de vehículos habrá un dispositivo de control que permita o impida el acceso de los mismos al parking, dependiendo del estado actual del mismo (plazas de aparcamiento disponibles).
    - Los tiempos de espera de los vehículos dentro del parking son aleatorios.
    - En el momento en el que un vehículo sale del parking, notifica al dispositivo de control el número de la plaza que tenía asignada y se libera la plaza que estuviera ocupando, quedando así estas nuevamente disponibles.
    - Un vehículo que ha salido del parking esperará un tiempo aleatorio para volver a entrar nuevamente en el mismo.
    - Por tanto, los vehículos estarán entrando y saliendo indefinidamente del parking.
    - Es importante que se diseñe el programa de tal forma que se asegure que, antes o después, un vehículo que permanece esperando a la entrada del parking entrará en el mismo (no se produzca inanición).

4. Escribe una clase llamada ParkingCamion que solicite el número de plazas del parking, el número de coches y el número de camiones existentes en el sistema. Dicha clase debe realizar lo mismo que la clase Parking pero debe permitir a su vez aparcar camiones.
    - Mientras un automóvil ocupa una plaza de aparcamiento dentro del parking, un camión ocupa dos plazas contiguas de aparcamiento.
    - Hay que tener especial cuidado con la inanición de camiones, que puede producirse si están saliendo coches indefinidamente y asignando la nueva plaza a los coches que esperan en vez de esperar a que haya un hueco para el camión (un camión solo podrá acceder al parking si hay, al menos, dos plazas contiguas de aparcamiento libre).