# Trabajo Práctico Final

Programación Concurrente, Departamento de Programación

**Nombre: Guillermo Nicolás**

**Apellido: Diaz**

**Legajo: FAI-3197**

**Año Cursado: 2022**

# Índice

**Enunciado** 3

**Recursos Compartidos** 3

- **Carrera** 3
- **Faro** 3
- **Restaurantes** 3
- **Snorkell** 4
- **Tienda** 4
- **Entrada Parque** 4

**Hilos** 4

- **AdminFaro** 4
- **Asistente** 4
- **Reloj** 5
- **Visitante** 5
- **Tren** 5
- **Gomón** 5

**Implementación** 6

**Consideraciones** 6

- **Configuración** 6
- **Test** 7
- **Correcciones** 8

# Enunciado

ECO-PARK

# Recursos Compartidos

- Carrera

| Recurso | Mecanismo de Sincronización |
|---|---|
| Bicicletas | Lock |
| Tren | Lock |
| Gomones Disponibles | BlockingQueue |
| Inicio Carrera | Cyclicbarrier |
| Gomon (lugares y comunicacion) | Semáforos |

- Faro

| Recurso | Mecanismo de Sincronización |
|---|---|
| Escalera | Semáforos |
| Tobogan | Lock |

- Restaurantes

| Recurso | Mecanismo de Sincronización |
|---|---|
| Cola de Espera | BlockingQueue |
| Lugares | Monitor |

- Snorkell

| Recurso | Mecanismo de Sincronización |
|---|---|
| Equipos | Lock |

- Tienda

| Recurso | Mecanismo de Sincronización |
|---|---|
| Cajas | Semáforos |

- Entrada Parque

| Recurso | Mecanismo de Sincronización |
|---|---|
| Colectivos | Cyclicbarrier |
| Molinetes | Monitor |

- LugarGomon

| Recurso | Mecanismo de Sincronización |
|---|---|
| Lugar | Semáforos |
| Comunicación (Entre gomon y visitante) | Semáforos |

# Hilos

*(Todos los hilos tienen ejecución infinita)*

- AdminFaro:
  - Administra los toboganes de la actividad del Faro
  - Espera que el visitante baje del tobogán
  - Una vez el visitante bajó del tobogán, avisa a los demás que hay un tobogán disponible

- Asistente:
  - Es un asistente en la actividad de Snorkell, administra los equipos
  - Entrega a los visitantes los equipos
  - Les avisa cuando hay un equipo disponible en caso de que estén todos en uso

- Reloj:
  - Utiliza el recurso Horario y aumenta el tiempo (hora y minuto)

- Visitante:
  - Accede al sitio donde está el parque
  - Intenta entrar por el molinete de acceso del parque
  - Mientras esté abierto el parque, realiza una serie de actividades, las cuales pueden ser:
    1. Ir a un Restaurante
    2. Ir al Faro Mirador
    3. Ir a Snorkell
    4. Ir a la Carrera de Gomones
    5. Ir al Shop del parque

- Tren:
    - Espera que la gente suba al tren
    - Una vez lleno, arranca el viaje hacia el sitio de la carrera de gomones
    - Deja a los pasajeros en el destino
    - Regresa de vuelta y le avisa a los proximos pasajeros que ya está disponible nuevamente

- Gomón:

  Si bien el gomón puede ser considerado un recurso, decidi hacerlo un hilo debido a que el enunciado decia que la carrera empezaba cuando habian n gomones listos, sin importar el tipo, y me pareció adecuado hacerlo como un hilo para poder trabajar con CyclicBarrier, de todas formas el gomón tiene como recurso compartido los lugares.
    - Espera que la gente se suba
    - Una vez lleno, espera la largada de la carrera
    - Hace la carrera (sleep)
    - Baja a los pasajeros y vuelve a esperar a los siguientes

# Implementación

Enlace al repositorio de github: TP-Final-Concurrente

# Consideraciones

- Configuración

  Puede modificar la configuración en el archivo config del proyecto
  Dentro se encuentran las variables que utiliza el parque, y puede realizar las siguientes modificaciones:

- Cambiar el horario de apertura/cierre del parque, controlar el delay del avance del tiempo.

- Cambiar el tamaño de variables que utilizan ciertas actividades, tales como la cantidad de molinetes, capacidad de los restaurantes, etc

- Modificar la cantidad de hilos (visitantes) que entran al parque

- Probar una actividad aislada (el visitante sólo podrá realizar dicha actividad)

## Test

Ejecutar el programa desde el main, se abrirá una GUI donde se mostrarán todas los visitantes que están en el parque, donde cada uno estará realizando alguna actividad del parque.
Se recomienda mirar el funcionamiento desde la interfaz gráfica, aunque si quiere puede verlo desde la consola, aunque habrán muchisimos mensajes y será dificil de interpretar.

Para probar alguna actividad en particular, entrar al archivo config y realizar lo siguiente:

```
public static boolean PRUEBA_INDIVIDUAL = false;
public static String ACTIVIDAD_AISLADA = "Ninguna";
```

Cambiar el valor de PRUEBA_INDIVIDUAL a true y luego escribir en ACTIVIDAD_AISLADA, la actividad que quiera probar, ejemplo:

```
public static boolean PRUEBA_INDIVIDUAL = true;
public static String ACTIVIDAD_AISLADA = "faro";
```

Luego puede elegir mirarlo desde la GUI o por consola, solo se imprimirá quien entra al parque y la actividad que eligió.

En la interfaz puede detener los hilos para ver la consola tranquilamente, pero tenga en cuenta que tendra que ejecutar de nuevo el código y no podrá reanudar la ejecución.  En la consola posiblemente cuando apriete el botón de detener vea algunas excepciones pero puede ignorarlas, son provenientes de usar el stop() en los hilos.

# Correcciones

- Faro

  Ya se modificó el faro y ahora tiene un límite en la plataforma, y evita el problema de que se llene infinitamente.
  Ahora cuando alguien quiera dejar las escaleras, esperará agarrar un permiso de un semaforo el cual representa la cantidad de gente que puede entrar a la plataforma

```
public void dejar_escaleras(Visitante v) throws InterruptedException{
      sem_plataforma.acquire(); //intenta entrar a la plataforma
      escalera_actual--;
      escribir(C.PURPLE, Color.magenta, v.getID()+" termino de subir las
escaleras "+escalera_actual+"/"+CAPACIDAD_ESCALERA);
      sem_escalera.release();
   }
```

Luego cuando el visitante sube al tobogán, libera un permiso de la plataforma para que otro pueda entrar:

```
public void subir_tobogan(Visitante v) throws InterruptedException{
      lock_tobogan.lock();

      while (tobogan_usado >= CANT_TOBOGANES){
         escribir(C.AMARILLO, Color.YELLOW, v.getID()+" espera
tobogan");
         esperar_tobogan.await();
      }
      tobogan_usado++;

      escribir(C.VERDE, Color.green,v.getID()+" entro a un tobogan
"+tobogan_usado+"/"+CANT_TOBOGANES);
      lock_tobogan.unlock();
      sem_plataforma.release();
   }
}
```

- Carrera de Gomones

  El hilo gomón ya no es más un objeto activo y pasivo, ahora tiene otra clase como referencia a el lugar de dicho gomon, alli es donde los visitantes deciden sentarse y ahi es donde el gomon les avisa cuando bajar (terminó la carrera)

  Metodo run de gomon, ya no tiene ni semaforos, ni ningun mecanismo, solo utiliza los recursos que tiene, primero espera que el gomón esté listo (los asientos estén ocupados), luego espera la largada de la carrera, una vez iniciada, hace un sleep y luego les avisa a los visitantes que desocupen el asiento, por último vuelve a colocarse el gomón dentro de la lista de gomones disponibles segun su tipo

```
 public void run() {
        try {
            while (true) {
                lugarGomon.esperar_gente(); //espera que se suba gente
                c.esperar_largada(this);
                dormir(10000); //lo que tarda en hacer la carrera
                lugarGomon.finCarrera(); //avisa a los visitantes que se
bajen
```

```
                dejarGomon();
            }
        } catch (InterruptedException  | BrokenBarrierException e) {

        }
  }
```

Metodos del recurso LugarGomon:

- esperar_gente(): El gomón espera el semaforo de n permisos (n es el límite del asiento, si es simple es 1 y si es doble es 2), es decir espera que se llene

```
 public void esperar_gente() throws InterruptedException{
        avisos.acquire(limite);

        if (doble){
            escribir(C.VERDE, Color.GREEN,"Visitantes " + getNombres() +
" agarraron el gomon doble N°" +id);
        } else{
            escribir(C.VERDE, Color.GREEN,"Visitante " +
lugares.getLast().getID() + " agarro el gomon simple N°" +id);
        }
    }
```

- finCarrera(): El gomón le avisa a los visitantes que terminó la carrera y reinicia los lugares

```
 public void finCarrera(){
        finRace.release(limite);
        lugares.clear();
        lugar.release(limite);
  }
```

- subir(): Visitante agarra un lugar y luego le avisa al gomón que subió

```
public void subir(Visitante v) throws InterruptedException{
        lugar.acquire();
        lugares.add(v);
        avisos.release();
    }
```

- esperarFinCarrera(): Visitante espera que termine la carrera

```
public void esperarFinCarrera(Visitante v) throws InterruptedException{
        finRace.acquire(); //espera que termine la carrera y sale
        escribir(C.ROJO, Color.RED,"Visitante " + v.getID() + " dejó el
gomon N°" + id);
    }
```