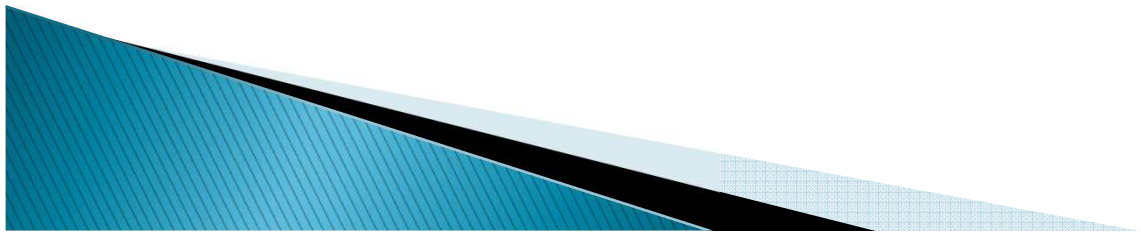
Prácticas de Semáforos. Aeropuerto

- Implementar aplicación de control de aterrizaje de aviones en un aeropuerto con una sola pista.
- El tiempo ambiente influye en el tiempo que el avión tarda en aterrizar. (1 soleado hasta 5 nieve)
- Visualizar los aviones aterrizados y los pendientes de aterrizar en cola.
- Cada X tiempo llega un avión nuevo al aeropuerto solicitando aterrizaje.
- Una vez implementado con una. Probar con varias pistas.



Propuesta solución Aeropuerto

```
Clase aviones extends Thread
Void run
    Solicitar aterrizaje
    Pista.acquire()
            Aviones esperando –
            Voy a aterrizar, Aterrizando
           Sleep(n) según el tiempo que haga
           Aterrizado
           Avionesaterrizados ++
    Pista.release()

Class Tiempo extends Thread
    Aleatorio calculo de tiempo
    Mientras cierto
            t= aleatorio.nextint(5)
            Sleep(1000)
    Fin mientras
```

```
main
        Int Tiempoactual,
        aviones_aterrizados;
        aviones_esperando;
        Semaphore Pista =new Semaphore(1)
        Tiempo t= new tiempo;
        tiempo.start()
        Desde ( 1 hasta 100)
            Aviones avion=new aviones
            Avion.start()
            Avionesesperando++
            Sleep(1000) hasta el próx avión
        Fin desde
        Esperar aviones.
        detener tiempo
Fin main
```

Prácticas Semáforos. Tarro Galletas

- https://encodingthecode.wordpress.com/2013/04/05/seguimos-con-concurrencia-ejemplo-de-uso-de-semaforos/
- Supongamos que disponemos de un tarro de galletas, y ‘k’ niños accediendo al bote, cuando este toca a su fin, los niños avisan a la madre para que reponga el tarro
- Primero, tendremos 3 clases en nuestro proyecto JAVA, una clase ‘niño’, encargada de actuar como el monstruo de las galletas xD, la clase ‘mama’, encargada de reponer el tarro cuando los niños lloren porque se acabó su “mirienda” (al estilo Shin Chan xD), y por último, una clase principal encargada de ‘crear’ a los niños y a la madre, para después lanzarlos

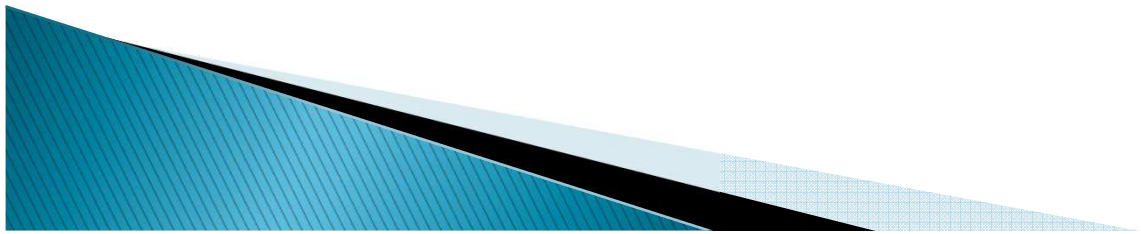
Análisis de la Solución. Galletas

- Main: lanza y crea niños y a la madre
- Hilo Hijo: Crear tantos como hijos vayan a comer
  - Cada hijo descuenta en uno el número de galletas y si hay =0?
    - Despierta a la madre y se bloquea
- Hilo Madre: Crear uno único hilo.
  - Inicialmente estará bloqueada hasta que un hijo diga que no hay galletas. En ese momento rellena el bote y despierta a los hijos
- Semáforos?
  - Hay galletas: los hijos comen mientras tanto, cuando sea 0, ya no pueden comer y despertaran a la madre
  - No hay galletas: la madre se despierta y rellena el bote
  - Mutex: cada hilo del hijo evalua el número de galletas y resta 1. Región crítica y por tanto es necesario asegurar que solo uno entra en el código

Propuesta Solución. Galletas

```
public  class  TARROGALLETAS {

    public  void  main () {
        galletas=n
        semáforo NoHayGalletas -> 0
        semáforo HayGalletas -> 0
        semáforo mutex -> 1
        CrearNiños()
        CrearMadre()
        LanzarNiños()
        LanzarMadre()
    }
}
```



Propuesta Solución. Galletas

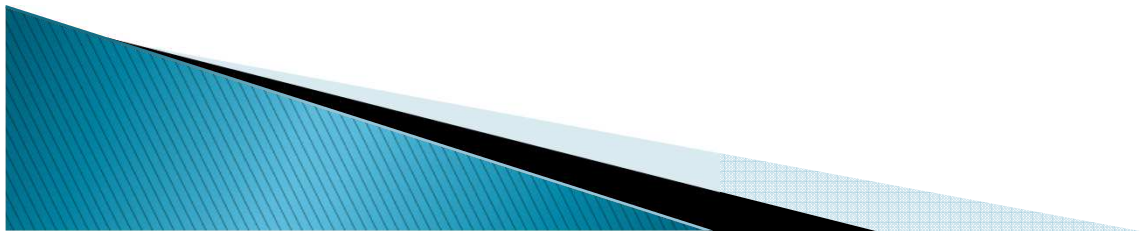
```
public  class  niño implements  Runnable {

    public  void  run () {
        while(true)                 //Los niños siempre estarán comiendo galletas
            Wait(mutex)            //Aquí evitamos que dos procesos comprueben el
número de galletas a la vez, ya que podría darse una falsa lectura
            if(galletas==0){    //Si no hay galletas...
                Signal(NohayGalletas)   //Desbloquea a la madre para que rellene
                Wait(HayGalletas)   //Espera a que la madre rellene para
continuar
                galletas--;      //coge galleta
            }else{
                galletas--;     //una galleta menos
            }
            signal(mutex)
    }
}
```

Propuesta Solución. Galletas

```
public  class  mama implements  Runnable {

    public  void  run () {
        while(true)  //La madre siempre esperando para reponer
            Wait(NoHayGalletas)  //Aquí se bloquea por que el semáforo
"NoHayGalletas" está a 0
            Wait(mutex)
            galletas=n              //Relleno el tarro de galletas
            Signal(mutex)           //El semáforo mutex impide que 2
procesos comprueben la variable "galletas" a la vez
            Signal(HayGalletas)  //Aviso a los niños de que hay
galletas poniendo a 1 el semáforo "HayGalletas"
    }
}
```



Solución. Galletas. Java 1

```
package TarroGalletas;
import java.util.concurrent.Semaphore; import java.util.*;
class missem {
        public static int galletas=0;
        public static Semaphore nohaygalletas= new Semaphore (0);
        public static Semaphore haygalletas= new Semaphore (0);
        public static Semaphore mutex= new Semaphore (1);
}
class nino extends Thread {

nino(int i){
        this.setName("nino"+i);     }
public void run () {
try{
while(true) { //Los ninos siempre estarán comiendo galletas
        sleep(1000);
        System.out.println("soy el " + this.getName() +" entrando");
        // evitamos que dos procesos comprueben el número de galletas a la vez, ya que podría darse una falsa lectura
        missem.mutex.acquire();
        System.out.println("soy el " + this.getName() + " paso mutex");
        if(missem.galletas == 0) { //Si no hay galletas...
                missem.nohaygalletas.release(); //Desbloquea a la madre para que rellene
                missem.haygalletas.acquire(); //Espera a que la madre rellene para continuar
                System.out.println("soy el " + this.getName() + " y Me como una galleta. Quedan " +
                        missem.galletas);
                missem.galletas--;  //coge galleta
        }else{
                System.out.println("soy el " + this.getName() + " y Me como una galleta. Quedan "+
                        missem.galletas);
                missem.galletas--;  //una galleta menos
        }
        missem.mutex.release();
}
}  catch (InterruptedExcep[illegible]e) {e.printStackTrace();}
} }
```

Solución. Galletas. Java 2

```
class mama extends Thread {

        mama(){
                this.setName("mama");
        }
        public void run () {
        try{
                while(true)  {//La madre siempre esperando para reponer
                        missem.nohaygalletas.acquire();
                        //Aquí se bloquea por que el semáforo "NoHayGalletas" está a 0
                        System.out.println("soy " + this.getName() + " paso nohaygalletas. mutex vale"
                                + missem.mutex.getQueueLength());
                        //missem.mutex.acquire();

                        System.out.println("soy " + this.getName() + " y relleno con 10 galletas.");
                        missem.galletas=10;
                          //Relleno el tarro de galletas

                        //missem.mutex.release();
                         //El semáforo mutex impide que 2 procesos comprueben la variable "galletas" a la vez
                        missem.haygalletas.release();
                          //Aviso a los niños de que hay galletas poniendo a 1 el semáforo "HayGalletas“
                }
                }  catch (InterruptedException e) {e.printStackTrace();}
        }
}
```

Solución. Galletas. Java 3

```
public class TarroGalletas {

        private static nino a_ninos[];

        public static void main(String[] args) {

                int num_ninos= 40;

                a_ninos=new nino[num_ninos];

                System.out.println( "voy a lanzar los niños...");
                //Creando y lanzando niños
                for (int i=0;i<num_ninos;i++){
                        a_ninos[i]= new nino(i+1);
                        a_ninos[i].start();
                }
                //Creando y Lanzando a la Madre

                System.out.println(" Voy a lanzar a la mama...");
                mama mimama=new mama();
                mimama.start();

        }
}
```