**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.

Class aviones extends Thread

Void run

Try

Solicitar aterrizaje

Pista.adquire()

Aviones esperando --

Voy a aterrizar, Aterrizando

Sleep(n) según tiempo

Aterrizado

Avionesaterrizados ++

Pista.release()

Catch

Class Tiempo extends )

Aleatorio

Try

While(true

Tiempo= aleatorio.nextint(5)

Sleep(1000)

Fin while

Catch

Void main

Int Tiempoactual, aviones\_aterrizados; aviones\_esperando;

Semaphore Pista =new Semaphore(1)

Tiempo t= new tiempo;

For ( 1 hasta 1000)

Aviones avion=new aviones

Avion.start()

Avionesesperando++

Sleep(1000)

Fin for

Tiempo.parar

Fin main

**SOLUCION**

package Aeropuerto;

import java.io.\*;

import java.lang.\*;

import java.util.concurrent.Semaphore;

import java.util.\*;

**class contadores {**

public static int T\_actual=1;

public static int aviones\_aterrizados=0;

public static int aviones\_esperando=0;

public static int T\_aterrizaje=0;

}

**class aviones extends Thread {**

private int num\_avion;

private Semaphore Sem\_Pista;

**aviones(int id\_numero, Semaphore Semaforo){**

this.num\_avion=id\_numero;

this.Sem\_Pista=Semaforo;

}

**public void run() {**

System.out.println("Solicitando Aterrizaje... Avion " + num\_avion);

try{

**Sem\_Pista.acquire();**

System.out.println("Hay "+ contadores.aviones\_esperando + " aviones esperando.." + Sem\_Pista.getQueueLength() + " - " + contadores.aviones\_aterrizados + " aterrizados");

contadores.aviones\_esperando--;

System.out.println ("Voy a aterrizar el avión...con tiempo.." +contadores.T\_actual+" Aterrizando..." + num\_avion);

this.sleep(contadores.T\_aterrizaje);

contadores.aviones\_aterrizados ++;

System.out.println(num\_avion +" ...Aterrizado y ya van" + contadores.aviones\_aterrizados + " aterrizados");

**Sem\_Pista.release();**

} catch (InterruptedException e) {e.printStackTrace();}

}

}

**class Tiempo extends Thread {**

public Random aleatorio= new Random();

boolean continua = true;

**public void para(){**

continua=false;

}

**public void run(){**

while(continua) {

contadores.T\_actual= aleatorio.nextInt(5)+1;

switch(contadores.T\_actual) {

case 1:

contadores.T\_aterrizaje=100;

break;

case 2:

contadores.T\_aterrizaje=200;

break;

case 3:

contadores.T\_aterrizaje=300;

break;

case 4:

contadores.T\_aterrizaje=400;

break;

case 5:

case 6:

contadores.T\_aterrizaje=4000;

break;

default:

contadores.T\_aterrizaje=10;

break;

}

}

System.out.println("Parando el tiempo....");

}

}

**public class Aeropuerto {**

private static Semaphore S\_Pista =new Semaphore(1);

private static aviones a\_aviones[];

**public static void main (String[] args) {**

int n\_aviones=50;

Tiempo tiempo= new Tiempo();

tiempo.start();

System.out.println("Arrancando aterrizajes....");

a\_aviones=new aviones[n\_aviones];

for ( int i=0; i<n\_aviones; i++) {

a\_aviones[i]=new aviones(i,S\_Pista);

a\_aviones[i].start();

contadores.aviones\_esperando++;

try{ Thread.sleep(900);} catch(InterruptedException e) {e.printStackTrace();} //espera hasta el siguiente avion

}

for(int i=0;i<n\_aviones;i++){

try{a\_aviones[i].join();} catch (InterruptedException e){e.printStackTrace();}

}

tiempo.para();

System.out.println (" Todos los aviones aterrizados....");

}

}