Reti e Laboratorio III
Modulo Laboratorio III
AA. 2023-2024

docente: Laura Ricci

laura.ricci@unipi.it

Esercizio 1

GreenHouse

28/9/2023

SIMULAZIONE DI UNA SERRA

- scrivere un programma che simuli la gestione di una serra,
- i processi che vengono attivati periodicamente nella serra sono i seguenti
  - periodicamente vengono aperti e poi chiusi dei distributori di acqua per annaffiare le piante
  - periodicamente si accendono e poi sis pengono dei led per fornire la luce che favorisce la fotosintesi
  - il termostato deve essere impostato ad un certo valore, durante le ore di luce e ad un altro valore durante la notte: si supponga che le ore di luce inizino alle `9:00` di mattina e terminino alle `19:00`, di sera
  - periodicamente una sorgente sonora scandisce il tempo
- il programma deve simulare i processi precedenti emettendo delle stampe che indico l'occorrenza del corrispondente evento

SIMULAZIONE DI UNA SERRA

```
import java.util.concurrent.*;
import java.time.LocalTime;
public class GreenhouseScheduler {
  private class LightOn implements Runnable {
    public void run() {
      System.out.println("Turning on lights");}}

  private class LightOff implements Runnable {
    public void run() {
      System.out.println("Turning off lights"); }}

 private class WaterOn implements Runnable {
    public void run() {
      System.out.println("Turning greenhouse water on");}}

  private class WaterOff implements Runnable {
    public void run() {
      System.out.println("Turning greenhouse water off");}}
```

SIMULAZIONE DI UNA SERRA

```
private class ThermostatNight implements Runnable {
    public void run() {
      System.out.println("Thermostat to night setting"); }}

 private class ThermostatDay implements Runnable {
    public void run() {
      System.out.println("Thermostat to day setting"); }}

private class Bell implements Runnable {
    public void run() { System.out.println("Bing!"); } }
```

SIMULAZIONE DI UNA SERRA

```
private class ThermostatNight implements Runnable {
    public void run() {
      System.out.println("Thermostat to night setting"); }}

 private class ThermostatDay implements Runnable {
    public void run() {
      System.out.println("Thermostat to day setting"); }}

private class Bell implements Runnable {
    public void run() { System.out.println("Bing!"); } }
```

SIMULAZIONE DI UNA SERRA

```
class Terminate implements Runnable {
      ScheduledThreadPoolExecutor scheduler;
      public Terminate (ScheduledThreadPoolExecutor scheduler)
             {this.scheduler=scheduler;}
      public void run() {
             System.out.println("Terminating");
             scheduler.shutdownNow();
   }
 }
```

SIMULAZIONE DI UNA SERRA

```
public static void schedule(ScheduledThreadPoolExecutor scheduler,
                            Runnable  event, long delay, TimeUnit tu) {
                            scheduler.schedule(event,delay,tu);
  }

public static void
  repeat(ScheduledThreadPoolExecutor scheduler,
              Runnable event, long initialDelay, long period, TimeUnit tu) {
              scheduler.scheduleAtFixedRate( event, initialDelay, period, tu);
  }
```

SIMULAZIONE DI UNA SERRA

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


  GreenhouseScheduler gh = new GreenhouseScheduler();
  ScheduledThreadPoolExecutor scheduler = new ScheduledThreadPoolExecutor(10);
  schedule(scheduler, gh.new Terminate(scheduler), 10000, TimeUnit.MILLISECONDS);
  repeat(scheduler,gh.new Bell(), 0, 1000, TimeUnit.MILLISECONDS);
  repeat(scheduler,gh.new LightOn(), 0, 200, TimeUnit.MILLISECONDS);
  repeat(scheduler,gh.new LightOff(), 100, 200, TimeUnit.MILLISECONDS);
  repeat(scheduler,gh.new WaterOn(), 0, 600, TimeUnit.MILLISECONDS);
  repeat(scheduler,gh.new WaterOff(), 300, 600, TimeUnit.MILLISECONDS);
```

SIMULAZIONE DI UNA SERRA

```
LocalTime now = LocalTime.now();
LocalTime morning = LocalTime.of(7, 0, 0);
LocalTime night= LocalTime.of(19, 0, 0);
Boolean light = now.isAfter(morning) && now.isBefore(night);
int interval = 86400;
if (light) {
     int secondi = night.toSecondOfDay() - now.toSecondOfDay();
     repeat(scheduler,gh.new ThermostatDay(), 0, interval, TimeUnit.SECONDS);
     repeat(scheduler,gh.new ThermostatNight(), secondi, interval, TimeUnit.SECONDS);}
 else {
      int secondi = morning.toSecondOfDay() - now.toSecondOfDay();
      repeat(scheduler,gh.new ThermostatNight(), 0, interval, TimeUnit.SECONDS);
     repeat(scheduler,gh.new ThermostatDay(),secondi, interval, TimeUnit.SECONDS);}}}
```