

## Simularea opririi la boxe a unei masini de Formula 1

### 1 Introducere - prezentarea problemei

In campionatul de Formula 1, mai mult decat in oricare alta competitie de curse auto, fiecare secunda conteaza. De aceea, performantele nu depind doar de pilot si de caracteristicile masinii, ci si de strategia cursei si de coordonarea mecanicilor pentru realizarea unei opriri la boxe cat mai scurte. Acest lucru presupune o sincronizare extraordinara si un efort de echipa care de multe ori este trecut cu vederea de catre fani, insa in aceasta lucrare imi propun sa simulez ce se intampla in garajul unei echipe in timpul unui pit stop, moment esential din timpul unei curse, ce poate determina o diferența importanta de puncte atat in campionatul pilotilor, cat si in campionatul constructorilor.



#### 1.1 Pas 1. Definire problemă

Aplicatia permite utilizatorului sa comande urmatoarele actiuni:

- Chemarea pilotului la boxe
- Oprirea rularii programului

Realizarea unei opriri la boxe eficiente depinde de comunicarea si sincronizarea mai multor entitati:

- Inginer cursa
- Pilot
- Mecanici pozitie: 2 mecanici (unul in fata si unul in spatele masinii) responsabili de ridicarea, respectiv coborarea acesteia + 2 mecanici (in lateralul masinii) responsabili de pastrarea acesteia intr-o stare de echilibru pentru a usura schimbarea pneurilor
- Mecanici pneuri: 12 mecanici, distribuiti astfel
  - Roata stanga fata: 3 mecanici
  - Roata stanga spate: 3 mecanici
  - Roata dreapta fata: 3 mecanici
  - Roata dreapta spate: 3 mecanici
- Persoana responsabila de stingerea focului/racirea franelor

Importanta coordonarii si necesitatea unei reactii rapide devin evidente in mod dramatic cand ne aducem aminte de momente critice din trecut, precum oprirea la boxe a pilotului Jos Verstappen, cand benzina din furtunul de alimentare s-a aprins pe o masina inca fierbinte pe vremea cand realimentarea in timpul cursei era permisa.



Figure 1: Jos Verstappen, Germania GP 1994

## 2 Analiza problemei

Aceasta sectiune va cuprinde analiza cerintelor.

### 2.1 Pas 2. Analiza cerintelor

Desfasurarea evenimentelor:

- Inginerul de cursa stabileste, conform strategiei urmarite, cand cheama pilotul la boxe. In aceasta simulare, tot el transmite echipei de la boxe sa isi pregateasca pozitiile cu 1 lap inainte de oprire.
- La semnalul inginerului de cursa, pilotul confirma oprirea si se indreapta catre linia boxelor, pe care mentine o viteza maxima de 80 kmh (in caz contrar fiind penalizat), actionand butonul PLS (pit lane speed) de pe volan. Este foarte important ca pilotul sa opreasca in locul potrivit, in caz contrar echipa pierzand timp pretios pentru a se repositiona, riscand chiar si accidente.
- Cand pilotul ajunge in pozitia marcata, mecanicii de pozitie ridica masina. De asemenea, cu o foarte mica intarziere, isi incep activitatea si mecanicii responsabili de pneuri.
- In acest timp, masina este mentinuta stabila de 2 mecanici aflati in stanga si dreapta masinii.
- Cand mecanicii responsabili de pneuri isi termina treaba, acestia le transmit un semnal mecanicilor de pozitie, pentru ca acestia sa coboare masina.
- Masina este apoi libera sa plece, accelerand pana la o viteza maxima de 80 kmh pana la iesirea de pe linia boxelor.
- Observatie: Temperatura franelor este generata de fiecare data cu ajutorul functiei rand() astfel incat sa aiba o valoare cuprinsa intre 500 si 1200 de grade Celsius. O valoare mai mare de 1000 de grade Celsius semnaleaza un pericol si determina activitatatile de stingere a focului/racire a franelor pana la o valoare de aproximativ 500 de grade Celsius, odata cu ajungerea la boxe a pilotului. Abia apoi sunt efectuate celelalte activitati: ridicarea masinii, schimbarea pneumelor etc.

Secvențele posibile sunt cele care respecta ordinea impusa de desfasurarea evenimentelor conform observatiilor de mai sus. Deoarece pneumile celor 4 roti sunt schimbat simultan, timpii fiind generati de functia rand(), nu se poate prezice ordinea de efectuare a activitatilor specifice schimbarii pneumelor: demontarea cu ajutorul unui pistol pneumatic, scoaterea pneului vechi, punerea pneului nou si asigurarea acestuia. Astfel, la rularea programului, vor fi afisati timpii corespunzatori fiecarii roti pentru a observa clar ordinea in care au terminat mecanicii de efectuat operatiile necesare.

Secvențe imposibile ar fi, de exemplu, schimbarea pneumelor inainte de a sosi pilotul, plecarea pilotului de la boxe inainte de a fi coborata masina etc.

### 3 Definirea structurii aplicației

În această secțiune se vor identifica taskurile care compun aplicația, în aşa fel încât să se folosească toate concepțele prezentare la curs:

- cerința principală este **planificarea taskurilor pe condiție de timp**;
- sincronizare taskurilor;
- excludere mutuală.

#### 3.1 Pas 3. Definirea taskurilor care compun aplicația

Aplicația implementată este formată din mai multe clase ce extind clasa Thread, pentru a oferi o simulare cat mai aproape de realitate. Pentru o organizare mai eficientă a codului, acestea sunt împărțite în pachete astfel:

- Pachetul pitStopStrategy
  - Coordonator principal (clasa responsabilă pentru declararea și initializarea semafoarelor și a unor variabile de tip static care vor putea fi modificate de către celelalte thread-uri, reprezentând totodată intrarea principală în program): **MainClass.java**
  - Inginer cursă (clasa responsabilă pentru chemarea pilotului la boxa și anunțarea mecanicilor): **InginerCursa.java**
- Pachetul driver
  - Pilot (clasa responsabilă pentru accelerarea/decelerarea mașinii și aducerea acesteia în zona marcată): **Pilot.java**
- Pachetul pitCrew
  - Mecanici poziție (clasa responsabilă atât pentru ridicarea și coborarea mașinii, cât și pentru stabilizarea acesteia în timpul schimbării pneurilor pentru a compensa dezechilibrul de forte la care este supusă mașina din partea mecanicilor de la fiecare roată): **MecaniciPozitie.java**
  - Mecanici pneuri (clasa ce coordonează schimbarea pneurilor): **MecaniciPneuri.java**
  - Roata stanga fata (clasa responsabilă pentru schimbarea pneului stanga fata): **PneuStangaFata.java**
  - Roata stanga spate (clasa responsabilă pentru schimbarea pneului stanga spate): **PneuStangaSpate.java**
  - Roata dreapta fata (clasa responsabilă pentru schimbarea pneului dreapta fata): **PneuDreaptaFata.java**
  - Roata dreapta spate (clasa responsabilă pentru schimbarea pneului dreapta spate): **PneuDreaptaSpate.java**
  - Stingere foc (clasa responsabilă pentru stingerea focului/racirea franelor în cazul în care temperatura acestora depășeste 1000 de grade Celsius): **StingatorFoc.java**

- semfaoare
  - **Semafor.java**: Clasa ce implementeaza un semafor generalizat, utilizand concepte de sincronizare si functiile monitor din clasa Object: wait(), notify()
  - **SemaforBinar.java**: Clasa ce extinde Semafor pentru a implementa un semafor binar, ce poate lua doar valorile 0 si 1

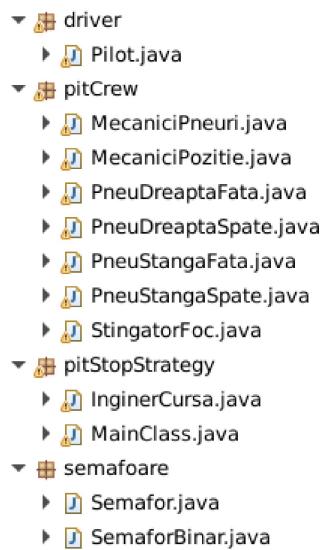


Figure 2: Vizualizarea pachetelor si a claselor componente

## 4 Definirea soluției în vederea implementării

Aplicatia se va implementa in Java, utilizand mecanisme de sincronizare, comunicare intre taskuri si de planificare a taskurilor pe conditie de timp.

### 4.1 Pas 4. Soluție de implementare

Aleg mecanismele de sincronizare și comunicare între taskuri, prezentând organigramele taskurilor (e.g. în Fig. 3 si în Fig. 4).

- semafoare binare:
  - sem\_pilot: inginerul de cursa anunta pilotul sa vina la boxe
  - sem\_mecanici: inginerul de cursa anunta mechanicii sa isi pregateasca pozitiile
  - sem\_pozitie, sem\_pneuri: odata cu sosirea pilotului la boxe, mechanicii de pozitie si cei responsabili de anvelope sunt semnalizati sa isi inceapa activitatea
  - sem\_go: pilotului i se semnalizeaza faptul ca poate pleca de la boxe
  - sem\_foc, foc\_stins: semnalizeaza o problema legata de temperatura franelor, respectiv rezolvarea acesteia
  - sem\_sf, gata\_sf: semnalizeaza inceperea schimbarii envelopelor la roata din stanga fata, respectiv terminarea acestei activitatii
  - sem\_ss, gata\_ss: semnalizeaza inceperea schimbarii envelopelor la roata din stanga spate, respectiv terminarea acestei activitatii
  - sem\_df, gata\_df: semnalizeaza inceperea schimbarii envelopelor la roata din dreapta fata, respectiv terminarea acestei activitatii
  - sem\_ds, gata\_ds: semnalizeaza inceperea schimbarii envelopelor la roata din dreapta spate, respectiv terminarea acestei activitatii
- blocuri synchronized:
  - In task-ul pilot, pentru a asigura ca viteza nu e modificata din alta parte cand pilotul accelereaza
  - In task-ul pilot, pentru a asigura ca viteza nu e modificata din alta parte cand pilotul decelereaza
  - In task-ul stingator\_foc, pentru a asigura ca temperatura nu e modificata din alta parte cand sunt racite franele

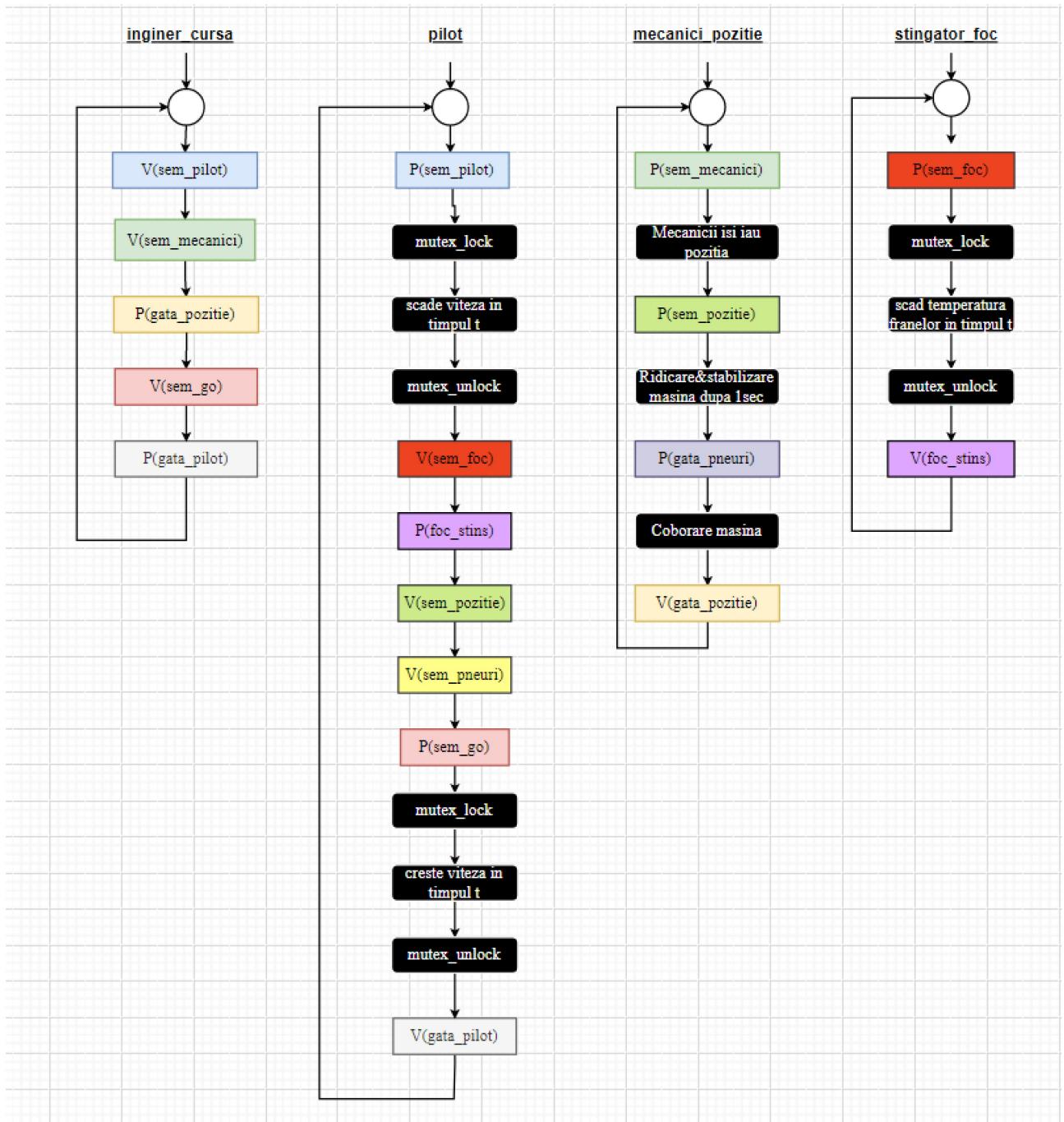


Figure 3: Solutie implementare - organigrame taskuri

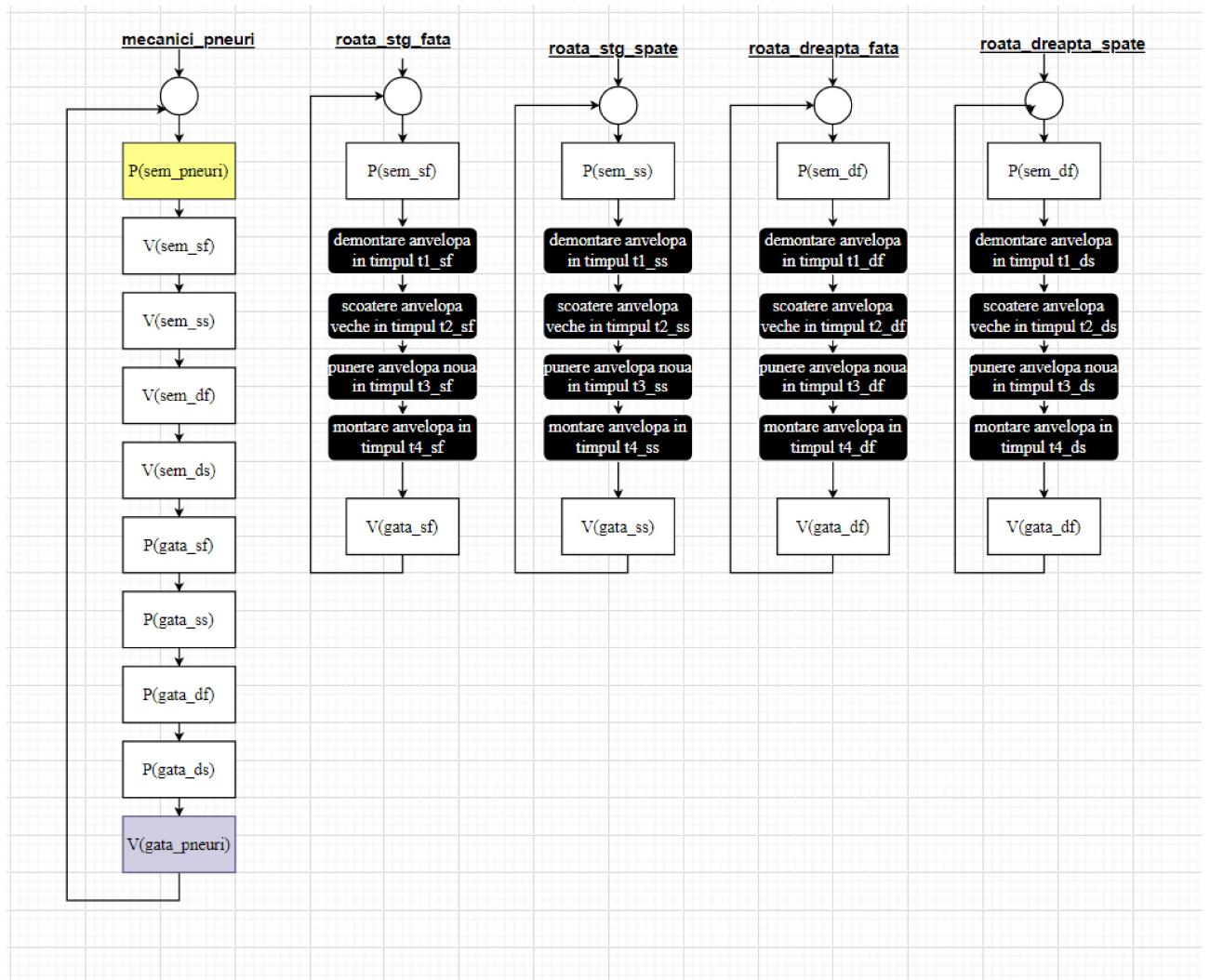


Figure 4: Soluție implementare - organigrame taskuri

## 5 Implementarea soluției

### 5.1 Cod program

---

```
//MainClass.java

package pitStopStrategy;
import pitStopStrategy.InginerCursa;
import pitCrew.*;
import driver.*;
import semafoare.*;

public class MainClass {
    public static int viteza, temperatura;
    public static int threshold_viteza = 80;

    public static SemaforBinar sem_pilot = new SemaforBinar();
    public static SemaforBinar sem_mecanici = new SemaforBinar();
    public static SemaforBinar sem_positie = new SemaforBinar();
    public static SemaforBinar sem_pneuri = new SemaforBinar();
    public static SemaforBinar sem_go = new SemaforBinar();
    public static SemaforBinar sem_foc = new SemaforBinar();
    public static SemaforBinar sem_sf = new SemaforBinar();
    public static SemaforBinar sem_ss = new SemaforBinar();
    public static SemaforBinar sem_df = new SemaforBinar();
    public static SemaforBinar sem_ds = new SemaforBinar();
    public static SemaforBinar gata_sf = new SemaforBinar();
    public static SemaforBinar gata_ss = new SemaforBinar();
    public static SemaforBinar gata_df = new SemaforBinar();
    public static SemaforBinar gata_ds = new SemaforBinar();
    public static SemaforBinar gata_pneuri = new SemaforBinar();
    public static SemaforBinar gata_positie = new SemaforBinar();
    public static SemaforBinar gata_pilot = new SemaforBinar();
    public static SemaforBinar foc_stins = new SemaforBinar();

    public static void main(String[] args) {
        InginerCursa inginerCursa = new InginerCursa();
        inginerCursa.start();
        Pilot pilot = new Pilot();
        pilot.start();
        MecaniciPneuri mechaniciPneuri = new MecaniciPneuri();
        mechaniciPneuri.start();
        MecaniciPositie mechaniciPositie = new MecaniciPositie();
        mechaniciPositie.start();
        StingatorFoc stingator = new StingatorFoc();
        stingator.start();
        PneuStangaFata psf = new PneuStangaFata();
        psf.start();
        PneuDreaptaFata pdf = new PneuDreaptaFata();
        pdf.start();
        PneuStangaSpate pss = new PneuStangaSpate();
        pss.start();
        PneuDreaptaSpate pds = new PneuDreaptaSpate();
```

```

        pds.start();
    }

}

//InginerCursa.java
package pitStopStrategy;
import pitStopStrategy.MainClass;
import java.util.Random;
import java.util.Scanner;

public class InginerCursa extends Thread {

    MainClass main = new MainClass();
    Random rand = new Random();
    Scanner scanner = new Scanner(System.in);

    @Override
    public void run() {

        int operatie = 0;

        main.viteza = 200 + rand.nextInt(120);
        main.temperatura = 500 + rand.nextInt(700);

        System.out.println("Conditii masina:\nViteza = " + main.viteza + " kmh\nTemperatura discurilor de frana = " + main.temperatura + " grade C");
        System.out.println("Selectati o comanda:");
        System.out.println("1. Chemare la boxe.");
        operatie = scanner.nextInt();
        System.out.println("Ati ales: " + operatie);
        while(1==1) {
            switch(operatie) {
                case 1:
                    main.sem_pilot.sem_post();
                    main.sem_mecanici.sem_post();
                    main.gata_pozitie.sem_wait();
                    main.sem_go.sem_post();
                    main.gata_pilot.sem_wait();
                    break;
                default:
                    System.out.println("Alegeti altceva");
                    operatie = scanner.nextInt();
            }
        }
    }

}

//Pilot.java
package driver;
import java.util.Calendar;

import pitStopStrategy.MainClass;

public class Pilot extends Thread {

```

```

MainClass main = new MainClass();
Calendar startPit, endPit;

@Override
public void run() {
    main.sem_pilot.sem_wait();

    System.out.println("[pilot]: Pilotul e chemat la boxe");
    try {
        sleep(1000);
    } catch (InterruptedException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
    System.out.println("[pilot]: Pilotul confirma oprirea");

    synchronized(this) {
        System.out.println("[pilot]: Viteza masinii este = " + main.viteza + " kmh");
        System.out.println("[pilot]: Viteza masinii scade...\n");
        while(main.viteza > main.threshold_viteza){
            main.viteza -= 10;
            try {
                sleep(250);
            } catch (InterruptedException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
        }
    }

    System.out.println("[pilot:] Intrare pit lane");
    System.out.println("[pilot]: Pilotul apasa butonul de reducere a vitezei in
pitlane");
    startPit = Calendar.getInstance();
    System.out.println("[pilot]: Viteza masinii este = " + main.viteza + " kmh\n");
    try {
        sleep(2000);
    } catch (InterruptedException e1) {
        // TODO Auto-generated catch block
        e1.printStackTrace();
    }

    System.out.println("[pilot]: Pilotul aduce masina in zona marcata din dreptul
garajului");
    main.viteza = 0;
    System.out.println("[pilot]: Viteza masinii este = " + main.viteza + " kmh\n");

    if(main.temperatura > 1000) {
        main.sem_foc.sem_post();
        main.foc_stins.sem_wait();
    }

    main.sem_positie.sem_post();
    try {

```

```

        sleep(1000);
    } catch (InterruptedException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
    main.sem_pneuri.sem_post();

    main.sem_go.sem_wait();

    synchronized(this) {
        System.out.println("\n[pilot]: Viteza masinii este = " + main.viteza + " kmh");
        System.out.println("[pilot]: Pilotul accelereaza");
        main.viteza = 80;
        System.out.println("[pilot]: Viteza masinii este = " + main.viteza + " kmh\n");
        try {
            sleep(5000);
        } catch (InterruptedException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        System.out.println("[pilot]: Exit pitlane");

        endPit = Calendar.getInstance();
        long durataPit = endPit.getTimeInMillis() - startPit.getTimeInMillis();
        System.out.println("\n[pilot]: Pitstop-ul a durat: " + durataPit/1000.0 + " secunde\n");

        System.out.println("[pilot]: Pilotul accelereaza");
        while(main.viteza < 200){
            main.viteza += 50;
            System.out.println("[pilot]: Viteza masinii este = " + main.viteza + " kmh");
            try {
                sleep(500);
            } catch (InterruptedException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
        }
        main.gata_pilot.sem_post();
    }
}

//MecaniciPozitie.java
package pitCrew;
import pitStopStrategy.MainClass;

public class MecaniciPozitie extends Thread{

```

```

@Override
public void run() {
    main.sem_mecanici.sem_wait();

    System.out.println("[mecanici_pozitie]: Mecanicii isi iau pozitia");

    main.sem_pozitie.sem_wait();

    try {
        sleep(1000);
    } catch (InterruptedException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }

    System.out.println("[mecanici_pozitie]: 2 mecanici ridica masina");
    System.out.println("[mecanici_pozitie]: 2 mecanici mentin masina stabila din
        lateral...");

    main.gata_pneuri.sem_wait();

    System.out.println("[mecanici_pozitie]: 2 mecanici coboara masina");

    main.gata_pozitie.sem_post();
}
}

//MecaniciPneuri.java
package pitCrew;
import java.util.Calendar;

import pitStopStrategy.MainClass;

public class MecaniciPneuri extends Thread{

    MainClass main = new MainClass();
    Calendar startPneuri, endPneuri;

    @Override
    public void run() {
        main.sem_pneuri.sem_wait();

        startPneuri = Calendar.getInstance();

        main.sem_sf.sem_post();
        main.sem_ss.sem_post();
        main.sem_df.sem_post();
        main.sem_ds.sem_post();
        main.gata_sf.sem_wait();
        main.gata_ss.sem_wait();
        main.gata_df.sem_wait();
        main.gata_ds.sem_wait();

        endPneuri = Calendar.getInstance();
    }
}

```

```

        long timpPneuri = endPneuri.getTimeInMillis() - startPneuri.getTimeInMillis();

        System.out.println("[mecanici_pneuri]: Pneurile au fost schimbatе in " +
                           timpPneuri/1000.0 + " secunde");

        main.gata_pneuri.sem_post();
    }
}

//PneuStangaFata.java
package pitCrew;
import pitStopStrategy.MainClass;

import java.util.Calendar;
import java.util.Random;

public class PneuStangaFata extends Thread{

    MainClass main = new MainClass();
    Random rand = new Random();
    Calendar start_sf, end_sf;

    @Override
    public void run() {
        main.sem_sf.sem_wait();

        start_sf = Calendar.getInstance();

        int timp = 400 + rand.nextInt(600);
        try {
            sleep(timp);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        System.out.println("[mecanici_stanga_fata ]: demonteaza pneurile");

        timp = 400 + rand.nextInt(600);
        try {
            sleep(timp);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        System.out.println("[mecanici_stanga_fata ]: scot pneurile vechi");

        timp = 400 + rand.nextInt(600);
        try {
            sleep(timp);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        System.out.println("[mecanici_stanga_fata ]: pun pneurile noi");

        timp = 400 + rand.nextInt(600);
        try {
            sleep(timp);
        }

```

```

} catch (InterruptedException e) {
    e.printStackTrace();
}
System.out.println("[mecanici_stanga_fata ]: monteaza pneurile");

end_sf = Calendar.getInstance();
long timp_sf = end_sf.getTimeInMillis() - start_sf.getTimeInMillis();
System.out.println("[mecanici_stanga_fata ]: Pneul stanga fata a fost schimbat
    in " + timp_sf/1000.0 + " secunde");

main.gata_sf.sem_post();
}

//PneuStangaSpate.java
package pitCrew;
import pitStopStrategy.MainClass;

import java.util.Calendar;
import java.util.Random;

public class PneuStangaSpate extends Thread{

MainClass main = new MainClass();
Random rand = new Random();
Calendar start_ss, end_ss;

@Override
public void run() {
    main.sem_ss.sem_wait();

    start_ss = Calendar.getInstance();

    int timp = 400 + rand.nextInt(600);
    try {
        sleep(timp);
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
    System.out.println("[mecanici_stanga_spate ]: demonteaza pneurile");

    timp = 400 + rand.nextInt(600);
    try {
        sleep(timp);
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
    System.out.println("[mecanici_stanga_spate ]: scot pneurile vechi");

    timp = 400 + rand.nextInt(600);
    try {
        sleep(timp);
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
}

```

```

        }
        System.out.println("[mecanici_stanga_spate ]: pun pneurile noi");

        timp = 400 + rand.nextInt(600);
        try {
            sleep(timp);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        System.out.println("[mecanici_stanga_spate ]: monteaza pneurile");

        end_ss = Calendar.getInstance();
        long timp_ss = end_ss.getTimeInMillis() - start_ss.getTimeInMillis();
        System.out.println("[mecanici_stanga_spate ]: Pneul stanga spate a fost
            schimbat in " + timp_ss/1000.0 + " secunde");

        main.gata_ss.sem_post();
    }
}

//PneuDreaptaFata.java
package pitCrew;
import pitStopStrategy.MainClass;

import java.util.Calendar;
import java.util.Random;

public class PneuDreaptaFata extends Thread{

    MainClass main = new MainClass();
    Random rand = new Random();
    Calendar start_df, end_df;

    @Override
    public void run() {
        main.sem_df.sem_wait();

        start_df = Calendar.getInstance();

        int timp = 400 + rand.nextInt(600);
        try {
            sleep(timp);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        System.out.println("[mecanici_dreapta_fata ]: demonteaza pneurile");

        timp = 400 + rand.nextInt(600);
        try {
            sleep(timp);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        System.out.println("[mecanici_dreapta_fata ]: scot pneurile vechi");
    }
}

```

```

        timp = 400 + rand.nextInt(600);
        try {
            sleep(timp);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        System.out.println("[mecanici_dreapta_fata ]: pun pneurile noi");

        timp = 400 + rand.nextInt(600);
        try {
            sleep(timp);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        System.out.println("[mecanici_dreapta_fata ]: monteaza pneurile");

        end_df = Calendar.getInstance();
        long timp_df = end_df.getTimeInMillis() - start_df.getTimeInMillis();
        System.out.println("[mecanici_dreapta_fata ]: Pneul dreapta fata a fost
            schimbat in " + timp_df/1000.0 + " secunde");

        main.gata_df.sem_post();
    }

}

//PneuDreaptaSpate.java
package pitCrew;
import pitStopStrategy.MainClass;

import java.util.Calendar;
import java.util.Random;

public class PneuDreaptaSpate extends Thread{

    MainClass main = new MainClass();
    Random rand = new Random();
    Calendar start_ds, end_ds;

    @Override
    public void run() {
        main.sem_ds.sem_wait();

        start_ds = Calendar.getInstance();

        int timp = 400 + rand.nextInt(600);
        try {
            sleep(timp);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        System.out.println("[mecanici_dreapta_spate ]: demonteaza pneurile");

        timp = 400 + rand.nextInt(600);
        try {

```

```

        sleep(timp);
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
    System.out.println("[mecanici_dreapta_spate ]: scot pneurile vechi");

    timp = 400 + rand.nextInt(600);
    try {
        sleep(timp);
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
    System.out.println("[mecanici_dreapta_spate ]: pun pneurile noi");

    timp = 400 + rand.nextInt(600);
    try {
        sleep(timp);
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
    System.out.println("[mecanici_dreapta_spate ]: monteaza pneurile");

    end_ds = Calendar.getInstance();
    long timp_ds = end_ds.getTimeInMillis() - start_ds.getTimeInMillis();
    System.out.println("[mecanici_dreapta_spate ]: Pneul dreapta spate a fost
        schimbat in " + timp_ds/1000.0 + " secunde");

    main.gata_ds.sem_post();
}

}

//StingatorFoc.java
package pitCrew;
import pitStopStrategy.MainClass;

public class StingatorFoc extends Thread {

    MainClass main = new MainClass();

    @Override
    public void run() {
        main.sem_foc.sem_wait();

        synchronized(this) {
            System.out.println("[mecanic_stingator_foc]: Stinge focul si raceste
                franele");
            while(main.temperatura > 500) {
                main.temperatura -= 90;
                System.out.println("Temperatura = " + main.temperatura + " grade C");
                try {
                    sleep(1000);
                } catch (InterruptedException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                }
            }
        }
    }
}

```

```

        }
    }
}

main.foc_stins.sem_post();
}
}

//Semafor.java
package semafoare;

public class Semafor {
    protected int value = 0;

    public Semafor() {
        value = 0;
    }

    public Semafor(int initial) {
        if (initial < 0)
            throw new IllegalArgumentException("initial<0");
        value = initial;
    }

    //equivalent of sem_wait() from POSIX (Linux, QNX)
    public synchronized void sem_wait() {
        if (value > 0)
            value--;
        else {
            while (true) { // we must be notified not interrupted
                try {
                    wait();
                    break; // notify(), so sem_wait() succeeds
                } catch (InterruptedException e) {
                    System.err.println("Semafor.wait():InterruptedException, wait again");
                    if (value > 0)
                        break; // race condition fix
                    else
                        continue; // no sem_post() yet
                }
            }
            value--;
        }
    }

    //equivalent of sem_post() from POSIX (Linux, QNX)
    public synchronized void sem_post() {
        value++;
        if (value == 1)
            notify();
    }

    public synchronized int getValue() {
        return value;
    }
}

```

```
public synchronized String toString() {
    return String.valueOf(value);
}
}

//SemaforBinar.java
package semafoare;

public final class SemaforBinar extends Semafor {

    public SemaforBinar() {
        super();
    }

    public SemaforBinar(int initial) {
        super(initial);
        if(initial > 1)
            throw new IllegalArgumentException("initial>1");
    }

    public SemaforBinar(boolean initial) {
        super(initial ? 1 : 0);
    }

    public final synchronized void sem_post( ) {
        super.sem_post();
        if(value > 1)
            value = 1;
    }
}
```

---

## 6 Testarea aplicației si validarea soluției propuse

```
Conditii masina:  
Viteza = 312 kmh  
Temperatura discurilor de frana = 898 grade C  
Selectati o comanda:  
1. Chemare la boxe.  
  
Ati ales: 1  
[pilot]: Pilotul e chemat la boxe  
[mecanici_pozitie]: Mecanicii isi iau pozitia  
[pilot]: Pilotul confirma oprirea  
[pilot]: Viteza masinii este = 312 kmh  
[pilot]: Viteza masinii scade...  
  
[pilot]: Intrare pit lane  
[pilot]: Pilotul apasa butonul de reducere a vitezei in pitlane  
[pilot]: Viteza masinii este = 72 kmh  
  
[pilot]: Pilotul aduce masina in zona marcata din dreptul garajului  
[pilot]: Viteza masinii este = 0 kmh  
  
[mecanici_pozitie]: 2 mecanici ridica masina  
[mecanici_pozitie]: 2 mecanici mentin masina stabila din lateral...  
[mecanici_stanga_fata ]: demonteaza pneumurile  
[mecanici_dreapta_fata ]: demonteaza pneumurile  
[mecanici_dreapta_spate ]: demonteaza pneumurile  
[mecanici_stanga_spate ]: demonteaza pneumurile  
[mecanici_stanga_fata ]: scot pneumurile vechi  
[mecanici_dreapta_fata ]: scot pneumurile vechi  
[mecanici_stanga_spate ]: scot pneumurile vechi  
[mecanici_dreapta_spate ]: scot pneumurile vechi  
[mecanici_dreapta_fata ]: pun pneumile noi  
[mecanici_stanga_fata ]: pun pneumile noi  
[mecanici_stanga_spate ]: pun pneumile noi  
[mecanici_stanga_fata ]: monteaza pneumurile  
[mecanici_stanga_fata ]: Pneul stanga fata a fost schimbat in 2.548 secunde  
[mecanici_dreapta_spate ]: pun pneumile noi  
[mecanici_dreapta_fata ]: monteaza pneumurile  
[mecanici_dreapta_fata ]: Pneul dreapta fata a fost schimbat in 2.789 secunde  
[mecanici_stanga_spate ]: monteaza pneumurile  
[mecanici_stanga_spate ]: Pneul stanga spate a fost schimbat in 3.114 secunde  
[mecanici_dreapta_spate ]: monteaza pneumurile  
[mecanici_dreapta_spate ]: Pneul dreapta spate a fost schimbat in 3.539 secunde  
[mecanici_pneuri]: Pneurile au fost schimbate in 3.543 secunde  
[mecanici_pozitie]: 2 mecanici coboara masina  
  
[pilot]: Viteza masinii este = 0 kmh  
[pilot]: Pilotul accelereaza  
[pilot]: Viteza masinii este = 80 kmh
```

```
[pilot]: Exit pitlane  
[pilot]: Pitstop-ul a durat: 11.654 secunde  
[pilot]: Pilotul accelereaza  
[pilot]: Viteza masinii este = 130 kmh  
[pilot]: Viteza masinii este = 180 kmh  
[pilot]: Viteza masinii este = 230 kmh
```

Figure 5: Rularea programului in cazul in care nu este necesara racirea franelor

```

Conditii masina:
Viteza = 270 kmh
Temperatura discurilor de frana = 1126 grade C
Selectati o comanda:
1. Chemare la boxe.
1
Ati ales: 1
[pilot]: Pilotul e chemat la boxe
[mecanici_pozitie]: Mecanicii isi iau pozitia
[pilot]: Pilotul confirma oprirea
[pilot]: Viteza masinii este = 270 kmh
[pilot]: Viteza masinii scade...

[pilot]: Intrare pit lane
[pilot]: Pilotul apasa butonul de reducere a vitezei in pitlane
[pilot]: Viteza masinii este = 80 kmh

[pilot]: Pilotul aduce masina in zona marcata din dreptul garajului
[pilot]: Viteza masinii este = 0 kmh

[mecanic_stingator_foc]: Stinge focul si raceste franele
Temperatura = 1036 grade C
Temperatura = 946 grade C
Temperatura = 856 grade C
Temperatura = 766 grade C
Temperatura = 676 grade C
Temperatura = 586 grade C
Temperatura = 496 grade C
[mecanici_pozitie]: 2 mecanici ridica masina
[mecanici_pozitie]: 2 mecanici mentin masina stabila din lateral...
[mecanici_dreapta_fata]: demonteaza pneurile
[mecanici_stanga_fata]: demonteaza pneurile
[mecanici_dreapta_spate]: demonteaza pneurile
[mecanici_stanga_spate]: demonteaza pneurile
[mecanici_dreapta_fata]: scot pneurile vechi
[mecanici_dreapta_spate]: scot pneurile vechi
[mecanici_stanga_fata]: scot pneurile vechi
[mecanici_stanga_spate]: scot pneurile vechi
[mecanici_dreapta_fata]: pun pneurile noi
[mecanici_dreapta_spate]: pun pneurile noi
[mecanici_stanga_spate]: pun pneurile noi
[mecanici_dreapta_fata]: monteaza pneurile
[mecanici_dreapta_fata]: Pneul dreapta fata a fost schimbat in 2.447 secunde
[mecanici_stanga_fata]: pun pneurile noi
[mecanici_dreapta_spate]: monteaza pneurile
[mecanici_dreapta_spate]: Pneul dreapta spate a fost schimbat in 2.499 secunde
[mecanici_stanga_spate]: monteaza pneurile
[mecanici_stanga_spate]: Pneul stanga spate a fost schimbat in 3.089 secunde

[mecanici_stanga_fata]: monteaza pneurile
[mecanici_stanga_fata]: Pneul stanga fata a fost schimbat in 3.398 secunde
[mecanici_pneuri]: Pneurile au fost schimbate in 3.401 secunde
[mecanici_pozitie]: 2 mecanici coboara masina

[pilot]: Viteza masinii este = 0 kmh
[pilot]: Pilotul accelereaza
[pilot]: Viteza masinii este = 80 kmh

[pilot]: Exit pitlane

[pilot]: Pitstop-ul a durat: 18.612 secunde

[pilot]: Pilotul accelereaza
[pilot]: Viteza masinii este = 130 kmh
[pilot]: Viteza masinii este = 180 kmh
[pilot]: Viteza masinii este = 230 kmh

```

Figure 6: Rularea programului in cazul in care este necesara racirea franelor

```
Conditii masina:  
Viteza = 286 kmh  
Temperatura discurilor de frana = 696 grade C  
Selectati o comanda:  
1. Chemare la boxe.  
3  
Ati ales: 3  
Alegeti altceva  
5  
Alegeti altceva  
1  
[pilot]: Pilotul e chemat la boxe  
[mecanici_pozitie]: Mecanicii isi iau pozitia  
[pilot]: Pilotul confirma oprirea  
[pilot]: Viteza masinii este = 286 kmh  
[pilot]: Viteza masinii scade...
```

Figure 7: Rularea programului in cazul in care se introduce o comanda gresita

## References

- [1] <https://statathlon.com/analysis-of-the-pit-stop-strategy-in-f1/>
- [2] <https://www.dummies.com/sports/auto-racing/formula-one-racing/what-happens-during-an-f1-pit-stop/>
- [3] <https://medium.com/@jake.bengco/f1-pit-stops-deconstructed-2c114b756531>
- [4] <https://www.sportskeeda.com/f1/what-happens-during-f1-pit-stop>