

Projecte Síntesi



Gasolinera

Curs: DAM2 (Desenvolupament d'Aplicacions Multiplataforma)

Alumne: Aitor Marín González

Plantejament del Projecte.....	4
implementacions amb .net:.....	4
Implementacions amb java:.....	5
Implementacions amb android:.....	6
Model Conceptual.....	7
GetData Java:.....	7
ProcessData Java:.....	8
SQL DataBase:.....	8
Codi Generacio BD:.....	9
Joc de Proves d'Aplicacions.....	11
GetData:.....	11
ProcessData:.....	11
Manual d'Usuari.....	12
Model Casos d'Us.....	15
Diagrames de Seqüència.....	16
GetData:.....	16
ProcessData:.....	17
Flux de Treball.....	18
Gestio de Riscos.....	19
Estimacio Economica.....	20

Control del Projecte.....	21
Codi Java.....	22
GetData:.....	22
GetData.java.....	22
Data.java.....	24
ProcessData:.....	26
Gasolinera.java.....	26
Combustible.java.....	29
Tanc.java.....	32
ProcessData.java.....	33
Proveiidor.java.....	41
Sortidor.java.....	43
Codi Android + APK's.....	46

Plantejament del Projecte

Realitzarem el diseny i la implementacio d'una base de dades i una serie d'aplicacions per a l'us i automatitzacio d'una gasolinera.

implementacions amb .net:

Una benzinera vol automatitzar la venda i distribucio del carburant, aixi con la reposicio del mateix, la estacio de servei en qüesto disposa de:

- **4 sortidors, el quals disposen de 2 tipus de combustibles.**
- **4 tipus de combustibles, distribuïts entre 4 sortidors.**
- **4 depòsits de combustible, on s'emmagatzemen el diferents combustibles**
- **una petita botiga on disposen de 4 articles relacionats amb el manteniment de l'automobil.**

Crearem una base de dades per emmagatzemar i poder consultar l'informacio necessaria, la base de dades contindra les següents taules:

[Link diagrama sqlServer](#)

Realitzarem 2 aplicacions de formulari, una client i una admin.

Client: Introduccio de les vendes d'articles i serveis de repostatge, generacio de tickets.

Admin: Mostrar de forma automatica l'estat dels sortidors, la capacitat actual dels depòsits i el total de vendes diari i mensual per articles, combustibles i sortidors.

Implementacions amb java:

Crearem dos apps java, una que recollira dades i les pasara a un archiu "SERVEIS.txt" i l'altra agafara les dades del "SERVEIS.txt" generat i les processara i mostrara de determinades formes.

GetData

- Javadoc
- Diagrama Classes
- Diagrama Seqüència
- Diagrama Casos d'Us

El format del fitxer "SERVEIS.txt" sera:

	nº de sortidor;	T	tipus de carburant;	HH:mm;	litres	
1	;	T2	;	10:50	;	50.20

ProcessData

- Javadoc
- Diagrama Classes
- Diagrama Seqüència
- Diagrama Casos d'Us

Mostrarem, per tipus de carburant, quin server a gastat mes litres i quin menys.

Recorrerem els sortidors i mostrarem quin dels deposits esta mes buit i realitzarem una comanda per reomplir-lo.

Llistarem els carburants per ordre de mes a menys contaminant.

Demanarem un nom de carburant, o un nom de proveïdor (els quals mostrarem alhora de demanarlos), i cercarem els combustibles que ens ven cada proveïdor o quins proveïdors en venen cada carburant.

Demanarem un tipus de carburant i mostrarem les seves caracteristiques.

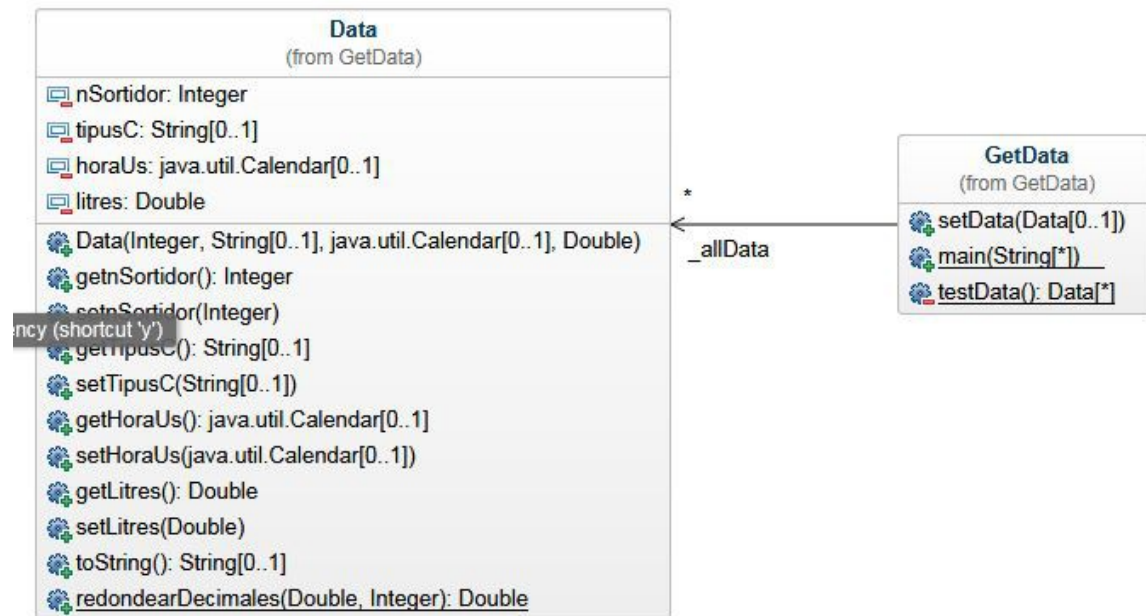
Implementacions amb android:

1^a: realitzar una app que llegeixi un fitxer (amb objectes tipus servei) on guardarem els serveis realitzats, l'app permetra llegir les dades existents e inserir noves dades.

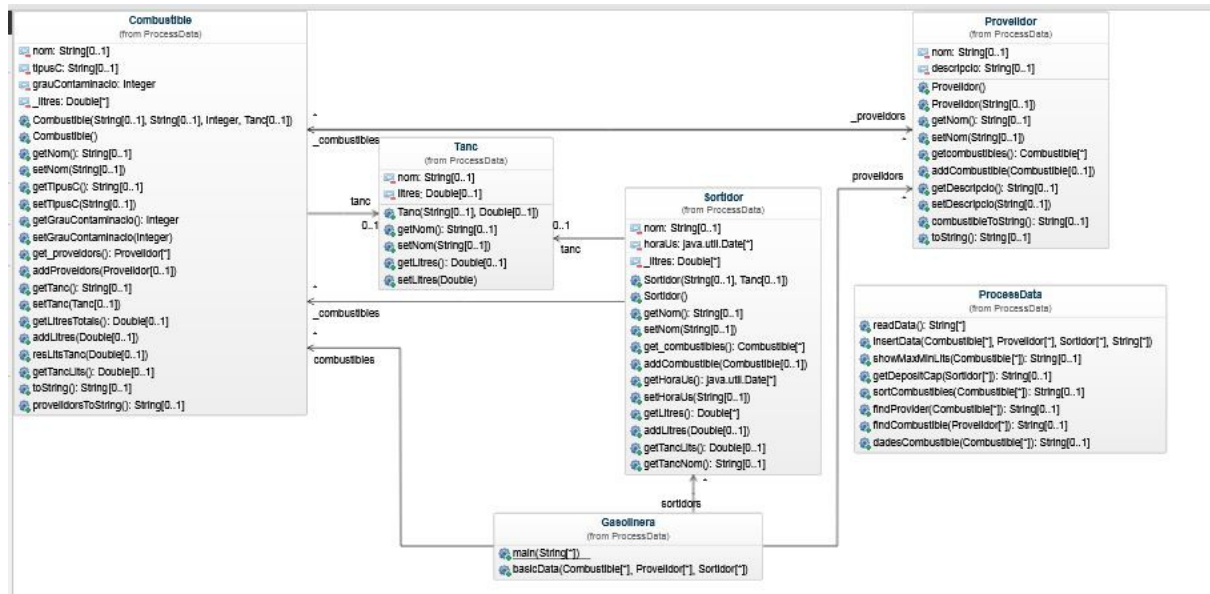
2^a: realitzar una app que mostri les gasolineres properes en relacio a la posicio actual del GPS. (Es tindra en compte com es mostren les dades)(esta tambien pa recu M8 GPS)

Model Conceptual

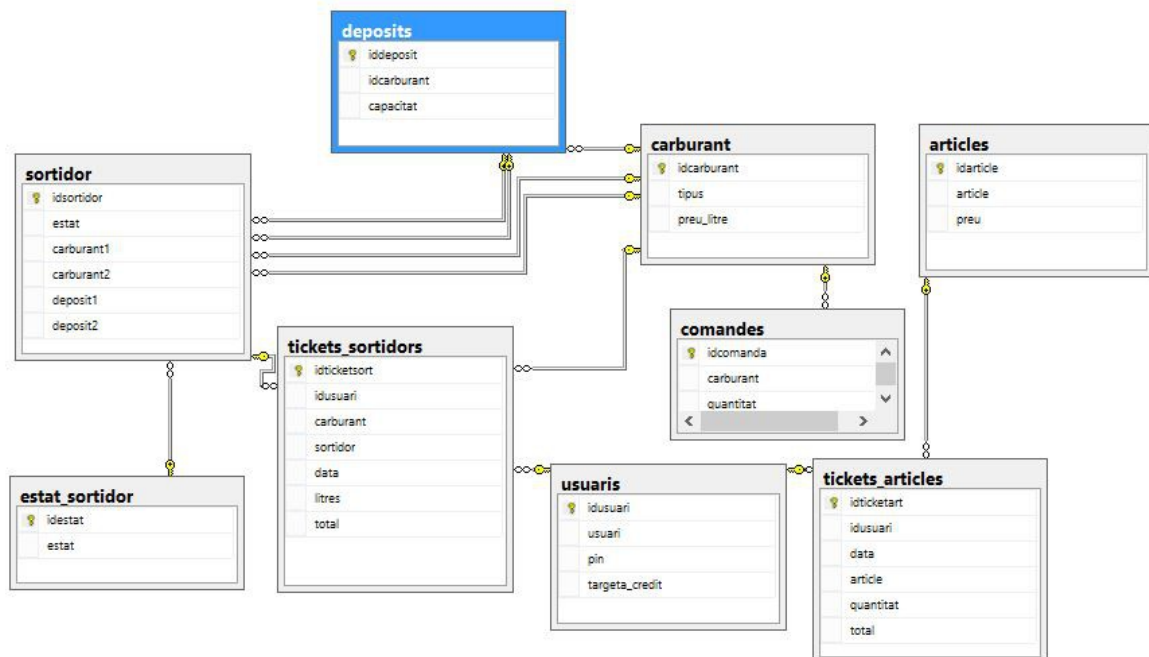
GetData Java:



ProcessData Java:



SQL DataBase:



Codi Generacio BD:

```
CREATE DATABASE Gasolinera;
GO
USE Gasolinera;
GO
CREATE TABLE deposits
(
    iddeposit INT IDENTITY(1,1) PRIMARY KEY,
    idcarburant INT FOREIGN KEY REFERENCES carburant(idcarburant),
    capacitat DECIMAL(8,2) NOT NULL CHECK(Capacitat <= 10000),
)
CREATE TABLE carburant
(
    idcarburant INT IDENTITY(1,1) PRIMARY KEY,
    tipus VARCHAR(100) NOT NULL,
    preu_litre DECIMAL(4,2),
)
CREATE TABLE estat_sortidor
(
    idestat INT IDENTITY(1,1) PRIMARY KEY,
    estat VARCHAR(100) NOT NULL,
)
CREATE TABLE sortidor
(
    idsortidor INT IDENTITY(1,1) PRIMARY KEY,
    estat INT FOREIGN KEY REFERENCES estat_sortidor(idestat),
    carburant1 INT FOREIGN KEY REFERENCES carburant(idcarburant),
    carburant2 INT FOREIGN KEY REFERENCES carburant(idcarburant),
    deposit1 INT FOREIGN KEY REFERENCES deposits(iddeposit),
    deposit2 INT FOREIGN KEY REFERENCES deposits(iddeposit),
)
CREATE TABLE comandes
(
    idcomanda INT IDENTITY(1,1) PRIMARY KEY,
    carburant INT FOREIGN KEY REFERENCES carburant(idcarburant),
    quantitat INT NOT NULL,
)
CREATE TABLE articles
(
    idarticle INT IDENTITY(1,1) PRIMARY KEY,
    article VARCHAR(100) NOT NULL,
    preu DECIMAL(4,2) NOT NULL,
)
CREATE TABLE usuaris
(
```

```

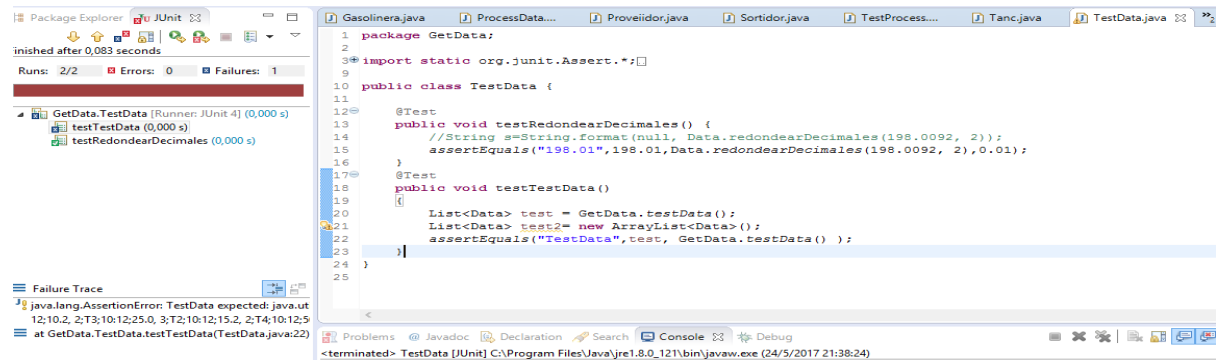
        idusuari INT IDENTITY(1,1) PRIMARY KEY,
        usuari VARCHAR(100) NOT NULL,
        pin VARCHAR(100) NOT NULL,
        targeta_credit VARCHAR(100),
    )
    CREATE TABLE tickets_sortidors
    (
        idticketsort INT IDENTITY(1,1) PRIMARY KEY,
        idusuari INT FOREIGN KEY REFERENCES usuaris(idusuari),
        carburant INT FOREIGN KEY REFERENCES carburant(idcarburant),
        sortidor INT FOREIGN KEY REFERENCES sortidor(idsortidor),
        data DATE NOT NULL,
        litres DECIMAL(8,2) NOT NULL,
        total DECIMAL(5,2) ,
    )
    CREATE TABLE tickets_articles
    (
        idticketart INT IDENTITY(1,1) PRIMARY KEY,
        idusuari INT FOREIGN KEY REFERENCES usuaris(idusuari),
        data DATE NOT NULL,
        article INT FOREIGN KEY REFERENCES articles(idarticle),
        quantitat SMALLINT NOT NULL,
        total DECIMAL(5,2) ,
    )
    --Triggers para pedido y para los totales de tiquets
    CREATE TRIGGER tr_comanda
    ON deposits
    AFTER UPDATE
    AS
    DECLARE @carburant INT = (SELECT idcarburant FROM deposits WHERE capacitat <=
    1000)
    DECLARE @capacitat DECIMAL(8,2) = (SELECT capacitat FROM deposits WHERE
    idcarburant=@carburant)
    DECLARE @total DECIMAL(8,2) = (@capacitat+9000)
    IF (@carburant IS NOT NULL)
    BEGIN
        INSERT INTO comandes (carburant, quantitat) VALUES (@carburant,9000)
        UPDATE deposits SET capacitat = @total WHERE idcarburant = @carburant
    END
    GO

```

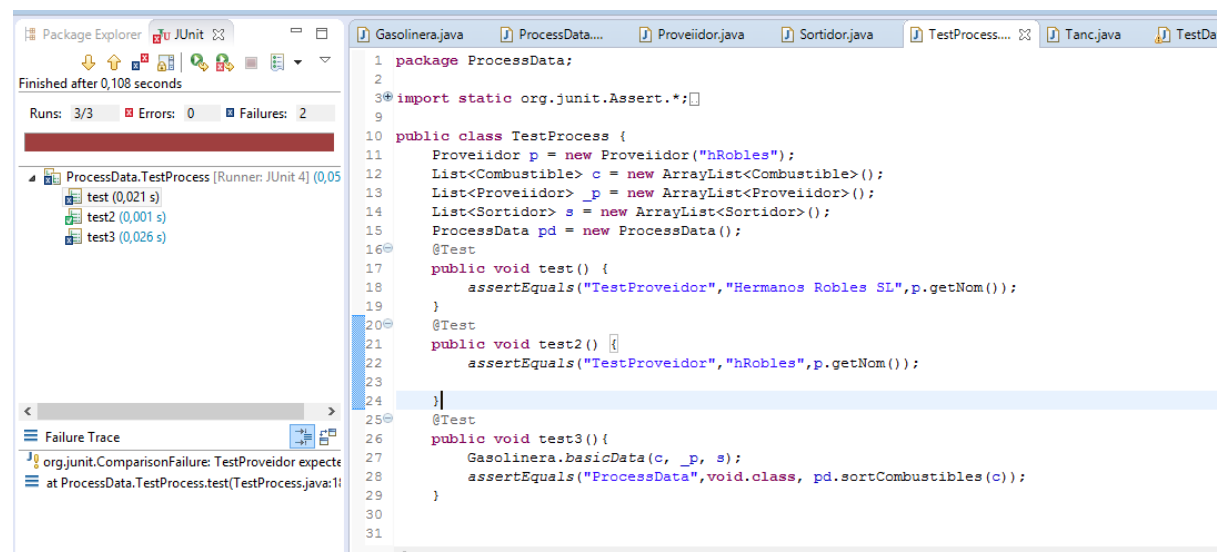
Joc de Proves d'Aplicacions

3 o 4 jocs de proves en JUnit per al codi java.

GetData:

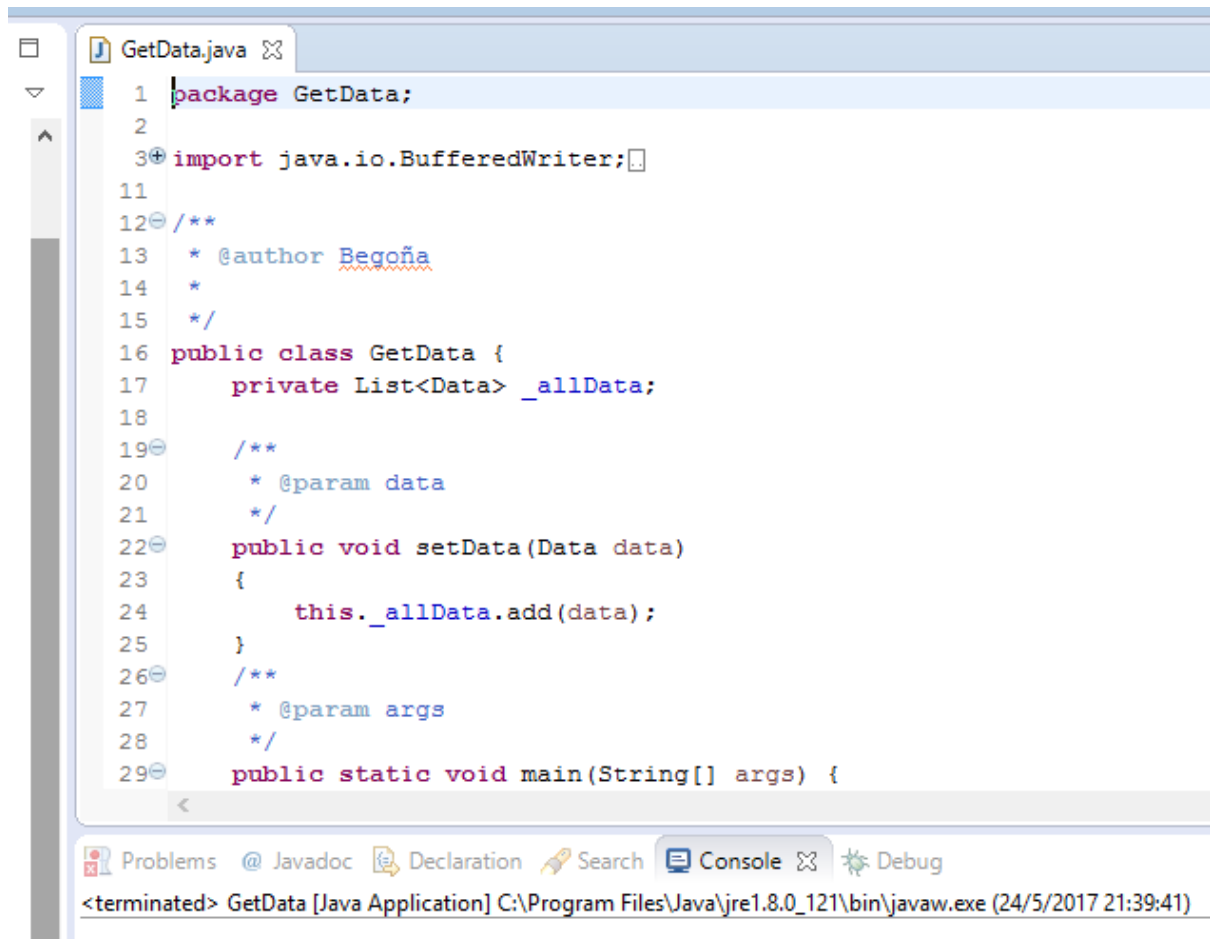


ProcessData:



Manual d'Usuari

GetData: Nomes necessitem executar aquesta app i automaticament ens generara el fitxer "SERVEIS.txt".

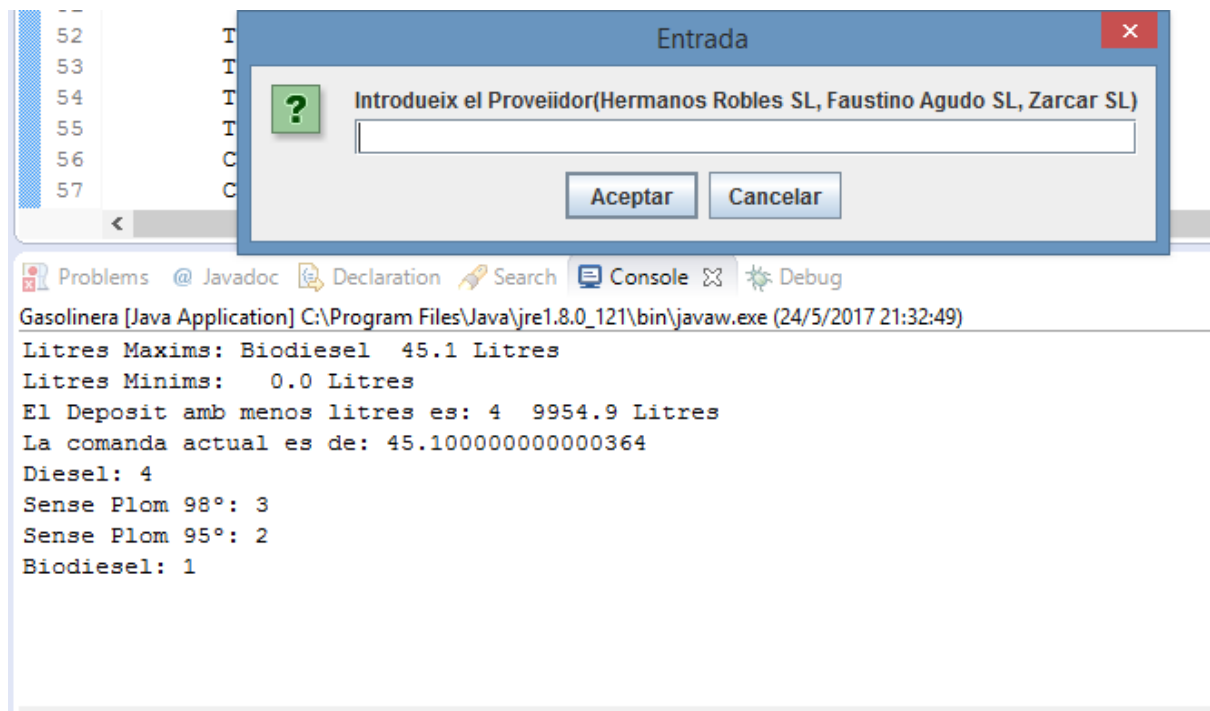


```
1 package GetData;
2
3 import java.io.BufferedWriter;
4
5 /**
6  * @author Begoña
7  */
8 public class GetData {
9     private List<Data> _allData;
10
11     /**
12      * @param data
13      */
14     public void setData(Data data)
15     {
16         this._allData.add(data);
17     }
18
19     /**
20      * @param args
21      */
22     public static void main(String[] args) {
23
24     }
```

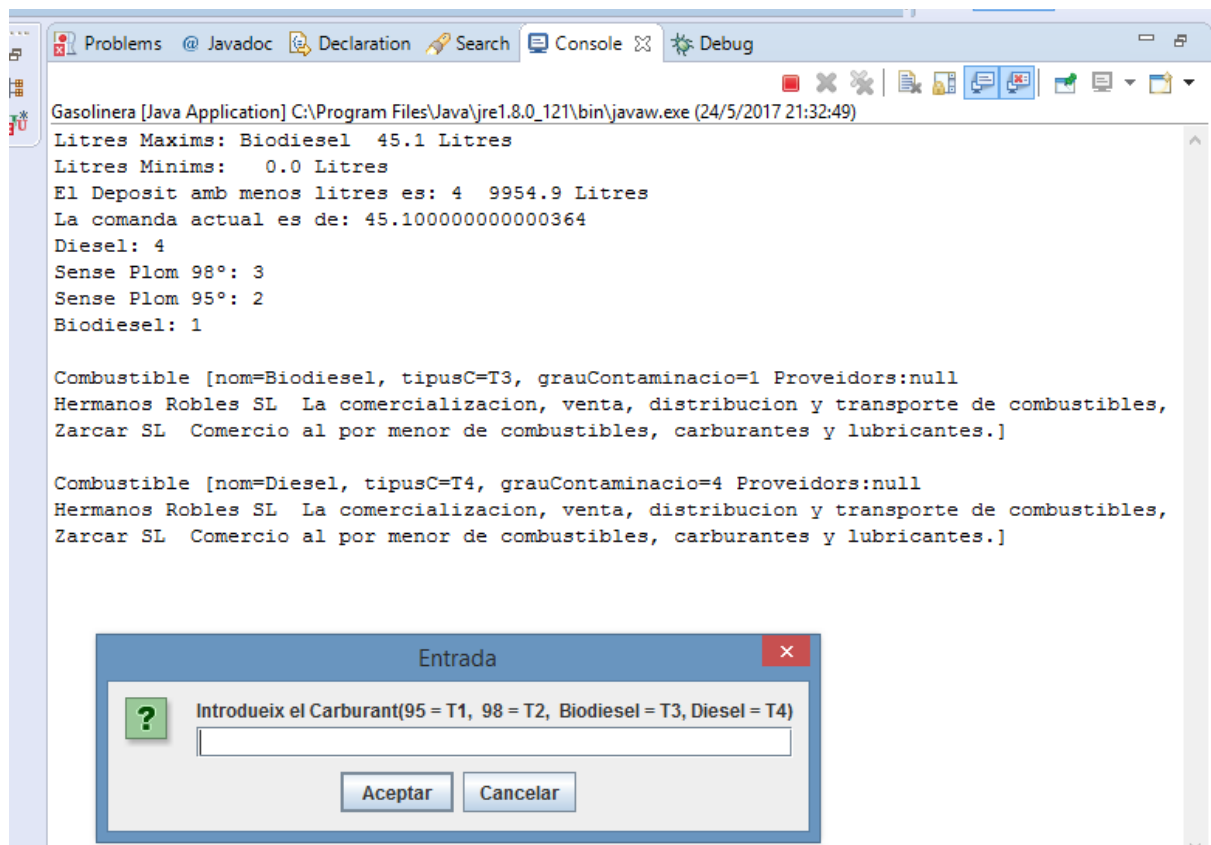
Problems @ Javadoc Declaration Search Console Debug

<terminated> GetData [Java Application] C:\Program Files\Java\jre1.8.0_121\bin\javaw.exe (24/5/2017 21:39:41)

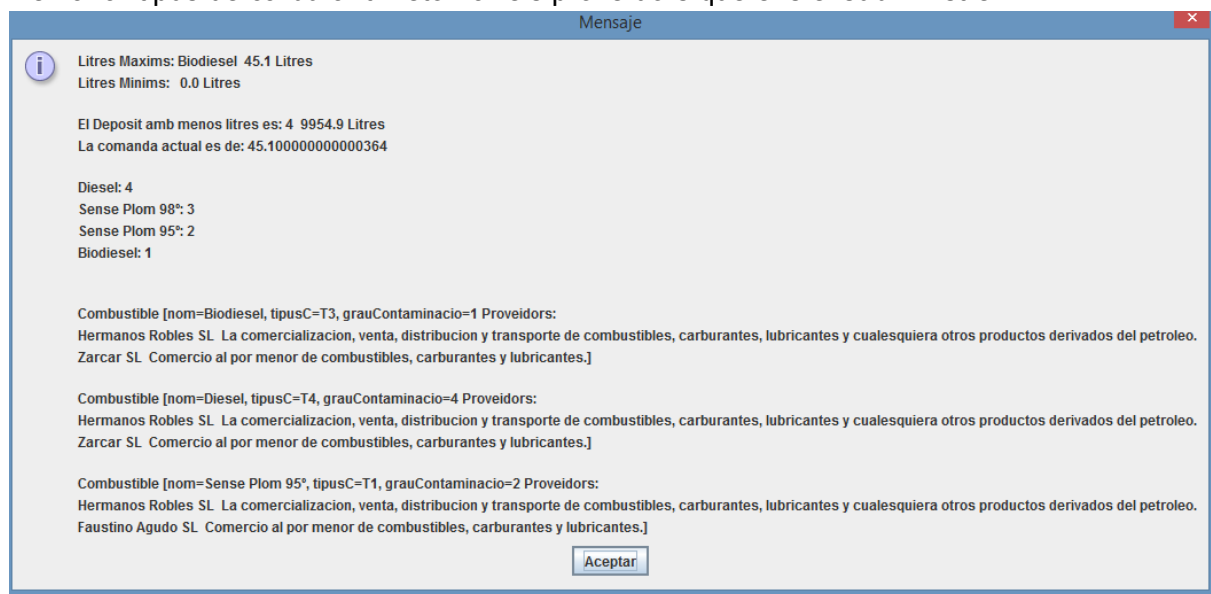
ProcessData: Al iniciar la app s'ens mostraran les dades dels serveis amb litres minim i maxim i la llista de carburants ordenats de mes a menys contaminant.



Com a opcions adicionales tindrem: Demanar un tipus de carburant a l'usuari, retornant les seves caracteristiques.

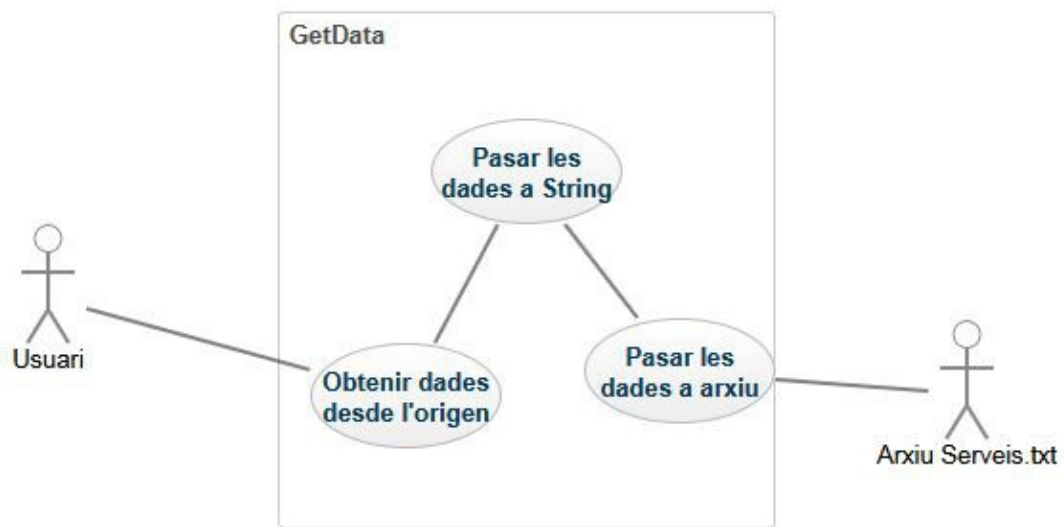


Demandar tipus de carburant i retornar els proveïdors que ens el subministren.

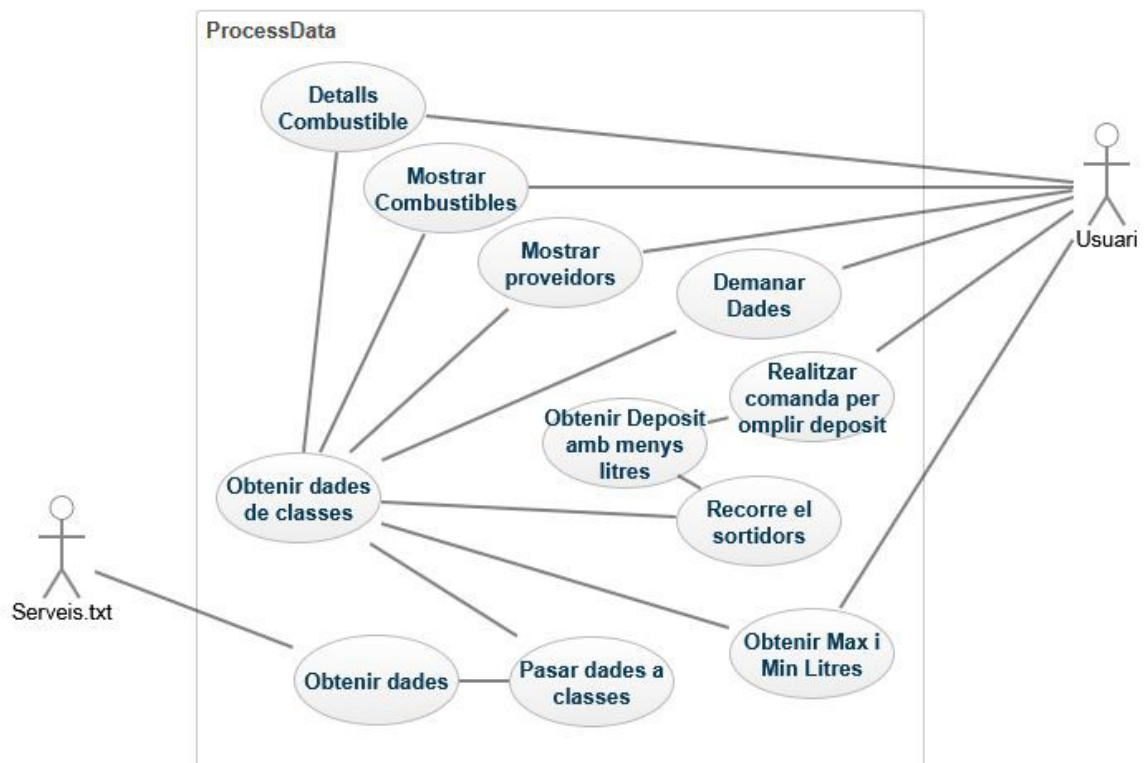


Model Casos d'Us

GetData:

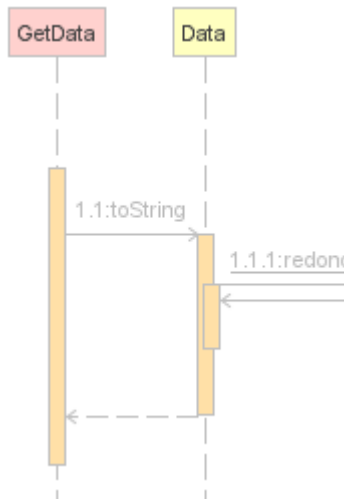


ProcessData:

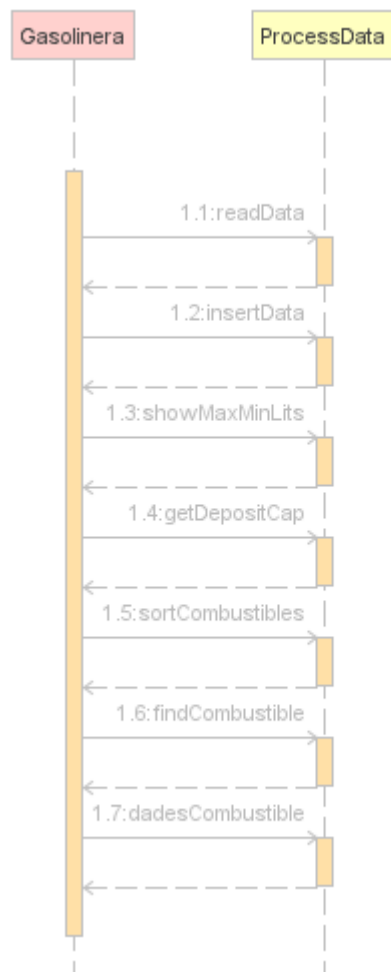


Diagramas de Seqüencia

GetData:



ProcessData:



Flux de Treball

Dilluns	Dimarts	Dimecres	Dijous	Divendres
Plantejament projecte - 1:30 h Implementacio amb java GetData - 2:30 h	Javadoc + Uml classes + seqüència + cas d'us GetData - 5 h	Implementacio amb java ProcessData + Javadoc - 5 h	Uml classes + seqüència + cas d'us ProcessData - 3:30 h	Manual d'usuari 0:30 h Disseny BBDD - 2 h Implementacio amb .net Client - 3 h
Dilluns	Dimarts	Dimecres	Dijous	Divendres
Implementacio amb .net Client - 2 h Implementacio amb .net Admin - 4 h	Implementacio amb android GPS	Implementacio amb android SERVEIS	<u>Uml classes + seqüència + cas d'us android</u>	Junit tests – 2 h

Gestio de Riscos

Estimacio Economica

Control del Projecte

Codi Java

GetData:

GetData.java

```
package GetData;

import java.io.BufferedWriter;
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.GregorianCalendar;
import java.util.List;

/**
 * @author Begoña
 */
public class GetData {
    private List<Data> _allData;

    /**
     * @param data
     */
    public void setData(Data data)
    {
        this._allData.add(data);
    }

    /**
     * @param args
     */
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        GetData gd=new GetData();
        gd._allData=new ArrayList<Data>();
        gd._allData=testData();
        File serveis = new File("SERVEIS.txt");
        try {
            FileWriter escriure = new FileWriter(serveis);
            BufferedWriter bw = new BufferedWriter(escriure);
```

```

        for(int i = 0; i<gd._allData.size(); i++)
        {
            bw.write(gd._allData.get(i).toString());
            bw.newLine();
        }
        bw.close();
        escriure.close();
    } catch (IOException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
}

```

```

/* Data Generated for Test Purposes */
/**
 * @return
 */
private static List<Data> testData()
{
    List<Data> _test = new ArrayList<Data>();
    double a= 9.5;
    Calendar c1 = new GregorianCalendar();
    c1.set(Calendar.HOUR_OF_DAY, 12);
    c1.set(Calendar.MINUTE, 40);
    Data d1 = new Data(1,"T4",c1,a);
    _test.add(d1);
    a= 10.2;
    c1.set(Calendar.HOUR_OF_DAY, 13);
    c1.set(Calendar.MINUTE, 04);
    Data d2 = new Data(1,"T3",c1,a);
    _test.add(d2);
    a= 25.0;
    c1.set(Calendar.HOUR_OF_DAY, 10);
    c1.set(Calendar.MINUTE, 11);
    Data d3 = new Data(2,"T3",c1,a);
    _test.add(d3);
    a= 15.18;
    c1.set(Calendar.HOUR_OF_DAY, 9);
    c1.set(Calendar.MINUTE, 45);
    Data d4 = new Data(3,"T2",c1,a);
    _test.add(d4);
    a= 50.9;
    c1.set(Calendar.HOUR_OF_DAY, 17);
    c1.set(Calendar.MINUTE, 30);
    Data d5 = new Data(2,"T4",c1,a);
}

```



```

        _test.add(d5);
        a= 25.0;
        c1.set(Calendar.HOUR_OF_DAY, 11);
        c1.set(Calendar.MINUTE, 46);
        Data d6 = new Data(3,"T3",c1,a);
        _test.add(d6);
        a= 11.5;
        c1.set(Calendar.HOUR_OF_DAY, 10);
        c1.set(Calendar.MINUTE, 15);
        Data d7 = new Data(2,"T2",c1,a);
        _test.add(d7);
        a= 20.11;
        c1.set(Calendar.HOUR_OF_DAY, 20);
        c1.set(Calendar.MINUTE, 26);
        Data d8 = new Data(3,"T3",c1,a);
        _test.add(d8);

        return _test;
    }
}

```

Data.java

```

package GetData;

import java.util.Calendar;

/**
 * @author Begoña
 *
 */
public class Data {
    /**
     * @param nSortidor
     * @param tipusC
     * @param horaUs
     * @param litres
     */
    public Data(int nSortidor, String tipusC, Calendar horaUs, double
    litres) {
        this.nSortidor = nSortidor;
        this.tipusC = tipusC;
        this.horaUs = horaUs;
        this.litres = litres;
    }
    private int nSortidor;
    private String tipusC;
    private Calendar horaUs;
    private double litres;
}

```

```

/**
 * @return
 */
public int getnSortidor() {
    return nSortidor;
}
/**
 * @param nSortidor
 */
public void setnSortidor(int nSortidor) {
    this.nSortidor = nSortidor;
}
/**
 * @return
 */
public String getTipusC() {
    return tipusC;
}
/**
 * @param tipusC
 */
public void setTipusC(String tipusC) {
    this.tipusC = tipusC;
}
/**
 * @return
 */
public Calendar getHoraUs() {

    return horaUs;
}
/**
 * @param horaUs
 */
public void setHoraUs(Calendar horaUs) {
    this.horaUs = horaUs;
}
/**
 * @return
 */
public double getLitres() {
    return litres;
}
/**
 * @param litres
 */
public void setLitres(double litres) {
    this.litres = litres;
}
/* (non-Javadoc)

```

```

        * @see java.lang.Object#toString()
        */
        @SuppressWarnings("static-access")
        public String toString()
        {
            return (Integer.toString(nSortidor)
+";"+tipusC+";"+horaUs.HOUR+": "+horaUs.MINUTE+";"+redondearDecimales(litres
,1));
        }
    /**
     * @param valorInicial
     * @param numeroDecimales
     * @return
     */
    public static double redondearDecimales(double valorInicial, int
numeroDecimales) {
        double parteEntera, resultado;
        resultado = valorInicial;
        parteEntera = Math.floor(resultado);
        resultado=(resultado-parteEntera)*Math.pow(10, numeroDecimales);
        resultado=Math.round(resultado);
        resultado=(resultado/Math.pow(10, numeroDecimales))+parteEntera;
        return resultado;
    }
}

```

ProcessData:

Gasolinera.java

```
package ProcessData;
```

```
import java.util.ArrayList;
```

```
import java.util.List;
```

```
import javax.swing.JOptionPane;
```

```
/**
```

```
 * @author Begoña
```

```
 *
```

```
 */
```

```
public class Gasolinera {
```

```
    private List<Combustible> combustibles = new ArrayList<Combustible>();
```

```
    private List<Proveiidor> proveiidores = new ArrayList<Proveiidor>();
```

```
    private List<Sortidor> sortidores = new ArrayList<Sortidor>();
```

```

/**
 * @param args
 */
public static void main(String[] args) {
    String[] data;
    Gasolinera g1 = new Gasolinera();
    basicData(g1.combustibles, g1.proveiidors, g1.sortidors);
    ProcessData process = new ProcessData();
    List<String> readedData = process.readData();
    StringBuilder sb = new StringBuilder();
    for(int i=0;i<readedData.size();i++)
    {
        sb.append(readedData.get(i)+"\n");
        data = readedData.get(i).split(";");
        process.insertData(g1.combustibles, g1.proveiidors, g1.sortidors,
data);
    }
    //JOptionPane.showMessageDialog(null, sb.toString());
    String litsMN = process.showMaxMinLits(g1.combustibles); //devuelve litros
max y min por carburante (es return)
    String depCap = process.getDepositCap(g1.sortidors);
    String Contaminants =
process.sortCombustibles(g1.combustibles).replace("null", " ");
    String CombFromProv =
process.findCombustible(g1.proveiidors).replace("null", " ");
    //String ProvFromComb = process.findProvider(g1.combustibles);
    String dadcomb =
process.dadesCombustible(g1.combustibles).replace("null", " ");

    JOptionPane.showMessageDialog(null,
litsMN+"\n\n"+depCap+"\n\n"+Contaminants+"\n\n"+CombFromProv+"\n\n"+dadcomb);
}

/**
 * @param c
 * @param p
 * @param s
 */
public static void basicData(List<Combustible> c, List<Proveiidor> p, List<Sortidor>
s) {

    Tanc t95 = new Tanc("Deposit 95", 10000.00);
    Tanc t98 = new Tanc("Deposit 98", 10000.00);
    Tanc tbiodiesel = new Tanc("Deposit Biodiesel", 10000.00);
    Tanc tdiesel = new Tanc("Deposit Diesel", 10000.00);

```

```

Combustible sp95 = new Combustible("Sense Plom 95º","T1",2,t95);
Combustible sp98 = new Combustible("Sense Plom 98º","T2",3,t98);
Combustible biodiesel = new Combustible("Biodiesel","T3",1,tbiodiesel);
Combustible diesel = new Combustible("Diesel","T4",4,tdiesel);
Sortidor s1 = new Sortidor("1",t95);
Sortidor s2 = new Sortidor("2",t98);
Sortidor s3 = new Sortidor("3",tbiodiesel);
Sortidor s4 = new Sortidor("4",tdiesel);
Proveiidor hRobles = new Proveiidor("Hermanos Robles SL");
Proveiidor fAgudo = new Proveiidor("Faustino Agudo SL");
Proveiidor zarcar = new Proveiidor("Zarcar SL");

```

```

s1.addCombustible(sp95);
s1.addCombustible(sp98);
s2.addCombustible(sp95);
s2.addCombustible(sp98);
s3.addCombustible(biodiesel);
s3.addCombustible(diesel);
s4.addCombustible(biodiesel);
s4.addCombustible(diesel);
hRobles.addCombustible(sp95);
hRobles.addCombustible(sp98);
hRobles.addCombustible(biodiesel);
hRobles.addCombustible(diesel);
fAgudo.addCombustible(sp95);
fAgudo.addCombustible(sp98);
zarcar.addCombustible(biodiesel);
zarcar.addCombustible(diesel);
sp95.addProveidores(hRobles);
sp95.addProveidores(fAgudo);
sp98.addProveidores(hRobles);
sp98.addProveidores(fAgudo);
biodiesel.addProveidores(hRobles);
biodiesel.addProveidores(zarcar);
diesel.addProveidores(hRobles);
diesel.addProveidores(zarcar);
c.add(sp95);
c.add(sp98);
c.add(biodiesel);
c.add(diesel);
p.add(hRobles);
p.add(fAgudo);
p.add(zarcar);
s.add(s1);
s.add(s2);
s.add(s3);

```

```

        s.add(s4);
    }
}

```

Combustible.java

```

package ProcessData;
import java.util.ArrayList;
import java.util.List;

```

```

/**
 * @author Begoña
 *
 */
public class Combustible {

    /**
     * @param nom
     * @param tipusC
     * @param grauContaminacio
     * @param tanc
     */
    public Combustible(String nom, String tipusC, int grauContaminacio, Tanc tanc) {
        this.nom = nom;
        this.tipusC = tipusC;
        this.grauContaminacio = grauContaminacio;
        this.setTanc(tanc);
    }

    public Combustible() {
    }

    private String nom;
    private String tipusC;
    private int grauContaminacio;
    private Tanc tanc;
    private List<Proveidor> _proveidors = new ArrayList<Proveidor>();
    private List<Double> _litres = new ArrayList<Double>();

    /**
     * @return
     */
    public String getNom() {
        return nom;
    }
}

```

```

}

/**
 * @param nom
 */
public void setNom(String nom) {
    this.nom = nom;
}

/**
 * @return
 */
public String getTipusC() {
    return tipusC;
}

/**
 * @param tipusC
 */
public void setTipusC(String tipusC) {
    this.tipusC = tipusC;
}

/**
 * @return
 */
public int getGrauContaminacio() {
    return grauContaminacio;
}

/**
 * @param grauContaminacio
 */
public void setGrauContaminacio(int grauContaminacio) {
    this.grauContaminacio = grauContaminacio;
}

/**
 * @return
 */
public List<Proveiidor> get_proveidors() {
    return _proveidors;
}

/**
 * @param proveidor

```

```

    */
    public void addProveidors(Proveiidor proveidor) {
        this._proveidors.add(proveidor);
    }

    /**
     * @return
     */
    public String getTanc() {
        return tanc.getNom();
    }

    /**
     * @param tanc
     */
    public void setTanc(Tanc tanc) {
        this.tanc = tanc;
    }

    /**
     * @return
     */
    public Double getLitresTotals() {
        Double sum = 0.0;
        for(Double d : this._litres)
        {
            sum+=d;
        }
        return sum;
    }

    /**
     * @param _litres
     */
    public void addLitres(Double _litres) {
        this._litres.add(_litres);
    }

    /**
     * @param lits
     */
    public void resLitsTanc(Double lits)
    {
        this.tanc.setLitres(lits);
    }

    /**
     * @return

```



```

        */
        public Double getTancLits()
        {
            return this.tanc.getLitres();
        }

        /* (non-Javadoc)
         * @see java.lang.Object#toString()
         */
        @Override
        public String toString() {
            return "Combustible [nom=" + nom + ", tipusC=" + tipusC
                + ", grauContaminacio=" + grauContaminacio + "
                Proveidors:" + proveidorsToString() + "]";
        }
        /**
         * @return
         */
        public String proveidorsToString()
        {
            String s=null;
            for(int i=0;i<this._proveidors.size();i++)
            {
                s+= "\n"+this._proveidors.get(i).getNom()+"
                "+this._proveidors.get(i).getDescripcio();
            }
            return s;
        }

    }

```

Tanc.java

```

package ProcessData;

/**
 * @author Begoña
 *
 */
public class Tanc {
    /**
     * @param nom
     * @param litres
     */

```

```

    public Tanc(String nom, Double litres) {
        this.nom = nom;
        this.litres = litres;
    }
    private String nom;
    private Double litres;
    /**
     * @return
     */
    public String getNom() {
        return nom;
    }
    /**
     * @param nom
     */
    public void setNom(String nom) {
        this.nom = nom;
    }
    /**
     * @return
     */
    public Double getLitres() {
        return litres;
    }
    /**
     * @param litres
     */
    public void setLitres(double litres) {
        this.litres += litres;
    }
}

```

ProcessData.java

```
package ProcessData;
```

```

import java.io.BufferedReader;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;

```

```
import javax.swing.JOptionPane;
```

```

/**
 * @author Begoña
 */

```

```

*/
public class ProcessData {

    /**
     * @return
     */
    public List<String> readData()
    {
        try{
            FileReader lector;
            try {
                lector = new FileReader("SERVEIS.txt");

                BufferedReader bf = new BufferedReader(lector);
                String linea;
                List<String> arlist = new ArrayList<String>();
                while((linea=bf.readLine())!=null){
                    arlist.add(linea);
                }

                bf.close();

                lector.close();

                return arlist;
            } catch (FileNotFoundException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
        } catch (IOException e){
            e.printStackTrace();
        }
        return null;
    }

    /**
     * @param combustibles
     * @param proveiidors
     * @param sortidors
     * @param data
     */
    public void insertData(List<Combustible> combustibles, List<Proveiidor> proveiidors,
        List<Sortidor> sortidors, String[] data)
    {
        if (data[0].equals("1")){
            sortidors.get(0).setHoraUs(data[2]);

```

```

        sortidors.get(0).addLitres(Double.parseDouble(data[3]));
        sortidors.get(0).addLitres(Double.parseDouble(data[3]));
        if(data[1].equals("T1")){
            combustibles.get(0).addLitres(Double.parseDouble(data[3]));
        } else if(data[1].equals("T2")){
            combustibles.get(1).addLitres(Double.parseDouble(data[3]));
        }
    }
} else if (data[0].equals("2")){
    sortidors.get(1).setHoraUs(data[2]);
    sortidors.get(1).addLitres(Double.parseDouble(data[3]));
    sortidors.get(1).addLitres(Double.parseDouble(data[3]));
    if(data[1].equals("T1")){
        combustibles.get(0).addLitres(Double.parseDouble(data[3]));
        combustibles.get(0).resLitsTanc(Double.parseDouble(data[3]));
    } else if(data[1].equals("T2")){
        combustibles.get(1).addLitres(Double.parseDouble(data[3]));
        combustibles.get(1).resLitsTanc(Double.parseDouble(data[3]));
    }
}
} else if (data[0].equals("3")){
    sortidors.get(2).setHoraUs(data[2]);

    sortidors.get(2).addLitres(Double.parseDouble(data[3]));

    sortidors.get(2).addLitres(Double.parseDouble(data[3]));
        if(data[1].equals("T3")){

        combustibles.get(2).addLitres(Double.parseDouble(data[3]));

        combustibles.get(2).resLitsTanc(Double.parseDouble(data[3]));
            } else if(data[1].equals("T4")){

        combustibles.get(3).addLitres(Double.parseDouble(data[3]));

        combustibles.get(3).resLitsTanc(Double.parseDouble(data[3]));
            }
        } else if (data[0].equals("4")){
            sortidors.get(3).setHoraUs(data[2]);

            sortidors.get(3).addLitres(Double.parseDouble(data[3]));

            sortidors.get(3).addLitres(Double.parseDouble(data[3]));
                if(data[1].equals("T3")){

                combustibles.get(2).addLitres(Double.parseDouble(data[3]));
                    } else if(data[1].equals("T4")){

```

```

        combustibles.get(3).addLitres(Double.parseDouble(data[3]));
    }
}

/**
 * @param combustibles
 * @return
 */
public String showMaxMinLits(List<Combustible> combustibles)
{
    Double d1, d2, d3, d4;
    String n1, n2, n3, n4, nmax = "", nmin = "";

    d1=combustibles.get(0).getLitresTotals();
    n1=combustibles.get(0).getNom();
    d2=combustibles.get(1).getLitresTotals();
    n2=combustibles.get(1).getNom();
    d3=combustibles.get(2).getLitresTotals();
    n3=combustibles.get(2).getNom();
    d4=combustibles.get(3).getLitresTotals();
    n4=combustibles.get(3).getNom();

    Double max = 0.0;
    if(d1>d2 && d1>d3 && d1>d4){
        max=d1;
        nmax=n1;
    } else if(d2>d1 && d2>d3 && d2>d4){
        max=d2;
        nmax=n2;
    } else if(d3>d1 && d3>d2 && d3>d4){
        max=d3;
        nmax=n3;
    } else if(d4>d1 && d4>d2 && d4>d3){
        max=d4;
        nmax=n4;
    }

    Double min = 0.0;
    if(d1<d2 && d1<d3 && d1<d4){
        min=d1;
        nmin=n1;
    } else if(d2<d1 && d2<d3 && d2<d4){
        min=d2;
        nmin=n2;
    } else if(d3<d1 && d3<d2 && d3<d4){

```

```

        min=d3;
        nmin=n3;
    } else if(d4<d1 && d4<d2 && d4<d3){
        min=d4;
        nmin=n4;
    }
    String s = ("Litres Maxims: "+nmax.toString()+" "+max.toString()+" Litres\n"+
        "Litres Minims: "+nmin.toString()+" "+min.toString()+" Litres");
    System.out.println(s);
    //JOptionPane.showMessageDialog(null,s);
    return s;
}
/**
 * @param sortidors
 * @return
 */
public String getDepositCap(List<Sortidor> sortidors)
{
    Double t1, t2, t3, t4, tt = null;
    String n1, n2, n3, n4 = null, nn=null;
    t1=sortidors.get(0).getTancLits();
    t2=sortidors.get(1).getTancLits();
    t3=sortidors.get(2).getTancLits();
    t4=sortidors.get(3).getTancLits();
    n1=sortidors.get(0).getNom();
    n2=sortidors.get(1).getNom();
    n3=sortidors.get(2).getNom();
    n3=sortidors.get(3).getNom();
    if(t1<t2 && t1<t3 && t1<t4){
        tt= t1;
        nn=n1;
    }else if(t2<t1 && t2<t3 && t2<t4){
        tt=t2;
        nn=n2;
    }else if(t3<t1 && t3<t2 && t3<t4){
        tt=t3;
        nn=n3;
    }else if(t4<t1 && t4<t2 && t4<t3){
        tt=t4;
        nn=n4;
    }
    double totcom = 10000.00-tt;
    System.out.println("El Deposit amb menos litres es: "+nn+" "+tt.toString()+"
    Litres"+"\\n"
    +"La comanda actual es de: "+totcom);
    return "El Deposit amb menos litres es: "+nn+" "+tt.toString()+" Litres"+"\\n"

```

```

        +"La comanda actual es de: "+totcom;
    }

    /**
     * @param combustibles
     * @return
     */
    public String sortCombustibles(List<Combustible> combustibles)
    {
        int c1 = 0, c2 = 0, c3 = 0, c4 = 0;
        String n1 = null, n2 = null, n3 = null, n4 = null;
        for(Combustible c : combustibles)
        {
            if(c.getGrauContaminacio()==4){
                c1=c.getGrauContaminacio();
                n1=c.getNom();
            }
            if(c.getGrauContaminacio()==3){
                c2=c.getGrauContaminacio();
                n2=c.getNom();
            }
            else if(c.getGrauContaminacio()==2 ){
                c3=c.getGrauContaminacio();
                n3=c.getNom();
            }
            else if(c.getGrauContaminacio()==1 ){
                c4=c.getGrauContaminacio();
                n4=c.getNom();
            }
        }

        System.out.println(n1+": "+c1+"\n"
            +n2+": "+c2+"\n"
            +n3+": "+c3+"\n"
            +n4+": "+c4+"\n");
        return n1+": "+c1+"\n"
            +n2+": "+c2+"\n"
            +n3+": "+c3+"\n"
            +n4+": "+c4+"\n";
    }

    /**
     * @param combustibles
     * @return
     */
    public String findProvider(List<Combustible> combustibles)
    {
        List<Proveiidor> lc = new ArrayList<Proveiidor>();
    }

```

```

        String s = (JOptionPane.showInputDialog("Introdueix el Carburant(95 = T1,
98 = T2, Biodiesel = T3, Diesel = T4)"));
        if(s==combustibles.get(0).getTipusC()){
            lc=combustibles.get(0).get_proveidors();
        }
        if(s==combustibles.get(1).getTipusC()){
            lc=combustibles.get(1).get_proveidors();
        }
        if(s==combustibles.get(2).getTipusC()){
            lc=combustibles.get(2).get_proveidors();
        }
        if(s==combustibles.get(3).getTipusC()){
            lc=combustibles.get(3).get_proveidors();
        }
        if (lc.size()==4)
        {
            System.out.println(lc.get(0).toString()+"\n\n"+lc.get(1).toString()
+"\\n\\n"+lc.get(2).toString()+"\\n\\n"+lc.get(3).toString());
            return "\\n"+lc.get(0).toString()+"\\n\\n"+lc.get(1).toString()
+"\\n\\n"+lc.get(2).toString()+"\\n\\n"+lc.get(3).toString();
        }else{
            //System.out.println("\\n"+lc.get(0).toString()+"\\n\\n"+lc.get(1).toString());
            return "\\n"+lc.get(0).toString()+"\\n\\n"+lc.get(1).toString();
        }
    }

/**
 * @param proveiidors
 * @return
 */
public String findCombustible(List<Proveiidor> proveiidors)
{
    List<Combustible> lc = new ArrayList<Combustible>();
    String s = (JOptionPane.showInputDialog("Introdueix el Proveiidor(Hermanos
Robles SL, Faustino Agudo SL, Zarcas SL)"));
    String busc = proveiidors.get(0).getNom().trim().toLowerCase();
    String ss = s.trim().toLowerCase();
    if (busc.equals(ss)){
        lc=proveiidors.get(0).getcombustibles();
    }
    busc = proveiidors.get(1).getNom().trim().toLowerCase();
    if (busc.equals(ss)){
        lc=proveiidors.get(1).getcombustibles();
    }
    busc = proveiidors.get(2).getNom().trim().toLowerCase();
    if (busc.equals(ss)){

```



```

        lc=proveiidors.get(2).getcombustibles();
    }
    if (lc.size()==4)
    {
        System.out.println(lc.get(0).toString()+"\n\n"+lc.get(1).toString()
+" \n\n"+lc.get(2).toString()+"\n\n"+lc.get(3).toString());
        return lc.get(0).toString()+"\n\n"+lc.get(1).toString()
+" \n\n"+lc.get(2).toString()+"\n\n"+lc.get(3).toString();
    }else{
        System.out.println(lc.get(0).toString()+"\n\n"+lc.get(1).toString());
        return lc.get(0).toString()+"\n\n"+lc.get(1).toString();
    }
}
/**
 * @param combustibles
 * @return
 */
public String dadesCombustible(List<Combustible> combustibles)
{
    String s = (JOptionPane.showInputDialog("Introdueix el Carburant(95 = T1,
98 = T2, Biodiesel = T3, Diesel = T4)"));

    if(combustibles.get(0).getTipusC().toLowerCase().trim().equals(s.toLowerCase().trim()))
    {
        System.out.println(combustibles.get(0).toString());
        return combustibles.get(0).toString();
    }else
    if(combustibles.get(1).getTipusC().toLowerCase().trim().equals(s.toLowerCase().trim()))
    {
        System.out.println(combustibles.get(1).toString());
        return combustibles.get(1).toString();
    }else
    if(combustibles.get(2).getTipusC().toLowerCase().trim().equals(s.toLowerCase().trim()))
    {
        System.out.println(combustibles.get(2).toString());
        return combustibles.get(2).toString();
    }else
    if(combustibles.get(3).getTipusC().toLowerCase().trim().equals(s.toLowerCase().trim()))
    {
        System.out.println(combustibles.get(3).toString());
        return combustibles.get(3).toString();
    }
    return "No s'ha trobat la busqueda: "+s;
}
}

```

Proveiidor.java

```
package ProcessData;
import java.util.ArrayList;
import java.util.List;

/**
 * @author Begoña
 */
public class Proveiidor {
    public Proveiidor() {
    }

    /**
     * @param nom
     */
    public Proveiidor(String nom) {
        this.nom = nom;
        this.setDescripcio(nom);
    }

    private String nom;
    private String descripcio;
    private List<Combustible> _combustibles = new ArrayList<Combustible>();

    /**
     * @return
     */
    public String getNom() {
        return nom;
    }

    /**
     * @param nom
     */
    public void setNom(String nom) {
        this.nom = nom;
    }

    /**
     * @return
     */
    public List<Combustible> getcombustibles() {
```

```

        return _combustibles;
    }

    /**
     * @param combustible
     */
    public void addCombustible(Combustible combustible) {
        this._combustibles.add(combustible);
    }

    /**
     * @return
     */
    public String getDescripcio() {
        return descripcio;
    }

    /**
     * @param nom
     */
    public void setDescripcio(String nom) {
        if (nom.equals("Hermanos Robles SL")){
            this.descripcio = "La comercializacion, venta, distribucion y transporte
de combustibles, carburantes, lubricantes y cualesquiera otros productos derivados del
petroleo.";
        }
        else if (nom.equals("Faustino Agudo SL")){
            this.descripcio = "Comercio al por menor de combustibles,
carburantes y lubricantes.";
        }
        else if (nom.equals("Zarcas SL")){
            this.descripcio = "Comercio al por menor de combustibles,
carburantes y lubricantes.";
        }
    }

    /**
     * @return
     */
    public String combustibleToString()
    {
        String s=null;
        for(int i=0;i<this._combustibles.size();i++)
        {
            s+= "\n"+this._combustibles.get(i).getNom()+"
"+this._combustibles.get(i).getTipusC()+"
"+this._combustibles.get(i).getGrauContaminacio()+" "+this._combustibles.get(i).getTanc();
        }
    }

```

```

        return s;
    }

    /* (non-Javadoc)
     * @see java.lang.Object#toString()
     */
    @Override
    public String toString() {
        return "Proveiidor [nom=" + nom + ", descripcio=" + descripcio
+combustibleToString()+ "]";
    }
}

```

Sortidor.java

```

package ProcessData;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.Date;
import java.util.List;

//import javax.swing.JOptionPane;

/**
 * @author Begoña
 *
 */
public class Sortidor {
    /**
     * @param nom
     * @param tanc
     */
    public Sortidor(String nom, Tanc tanc) {
        this.nom = nom;
        this.tanc=tanc;
    }

    public Sortidor() {
    }

    private String nom;
    private List<Date> horaUs = new ArrayList<Date>();
    private List<Double> _litres = new ArrayList<Double>();
}

```

```

private List<Combustible> _combustibles = new ArrayList<Combustible>();
private Tanc tanc;

/**
 * @return
 */
public String getNom() {
    return nom;
}

/**
 * @param nom
 */
public void setNom(String nom) {
    this.nom = nom;
}

/**
 * @return
 */
public List<Combustible> get_combustibles() {
    return _combustibles;
}

/**
 * @param combustible
 */
public void addCombustible(Combustible combustible) {
    this._combustibles.add(combustible);
}

/**
 * @return
 */
public List<Date> getHoraUs() {
    return horaUs;
}

/**
 * @param horaUs
 */
public void setHoraUs(String horaUs)
{
    try {
        SimpleDateFormat sdf = new SimpleDateFormat("HH:mm");
    }
}

```

```

        Calendar cal = Calendar.getInstance();

        cal.setTime(sdf.parse(horaUs));
        this.horaUs.add(cal.getTime());
    } catch (ParseException e) {
        e.printStackTrace();
    }
}

/**
 * @return
 */
public List<Double> getLitres() {
    return _litres;
}

/**
 * @param _litres
 */
public void addLitres(Double _litres) {
    this._litres.add(_litres);
}

/**
 * @return
 */
public Double getTancLits()
{
    return this.tanc.getLitres();
}

/**
 * @return
 */
public String getTancNom()
{
    return this.tanc.getNom();
}
}

```

Codi Android + APK's