Solution TD/TP 4

```
enum Carburant {
    diesel, essence, gaz
}
```

```
public interface Decapotable {
    public void replieLeToit();
}
```

```
public interface Motorisation {

    public Carburant typeCarburant();
    public void periodiciteVidange();

}
```

```
import java.io.Serializable;

public abstract class Vehicule implements Motorisation, Serializable {

    //attributs
    private String matricule;
    private int modele;
    private double carburant;
    private double prix;

    // constructeur
    public Vehicule(int _modele){ modele = _modele; }

    public void setCarburant(double _carburant){carburant=_carburant;}
    public double getCarburant(){return carburant;  }
    public void setMatricule(String _matricule){matricule=_matricule;}
    public String getMatricule(){return matricule;}
    public void setModele(int _modele){modele=_modele;}
    public int getModele(){return modele;}
    public void setPrix(double _prix) {prix=_prix;}
    public double getPrix() {return prix;}

    public String toString() {return "matricule: "+matricule+", modele:
"+modele;}

}
```

```
import java.io.Serializable;

class Voiture extends Vehicule implements Decapotable {

    private static int nbMaxVoitures = 10;
```

```
    private static int capacite = 60;
    private static int nbVoitures = 0;
    private Carburant carburant;
    private int periodiciteVidange;
    private boolean toitReplie;

    // le constructeur est private pour empecher la création d'objet avant de
s'assurer qu'il y'a de la place dans le parc
    private Voiture (int modele, Carburant carburant) {
        super(modele);
        this.setMatricule(this.getClass().getName()+":"+ ++nbVoitures);
        this.carburant = carburant;
    }

    public static Voiture creerVoiture(int modele, Carburant carburant) {
        if (nbVoitures < nbMaxVoitures)  return new Voiture(modele,carburant);
        else System.out.println("Nombre max de voiture est atteint et le parc est
fermé");
        return null;
    }

    public void addCarburant(double carburant) {
        if ((getCarburant()+carburant)<= capacite)
setCarburant(getCarburant()+carburant);
        else {
            double max_carburant_ajout = capacite - getCarburant();
            this.setCarburant(capacite);
            System.out.println("la quantite :"+(carburant-max_carburant_ajout)+"
a deborder");
        }
    }

    public Carburant typeCarburant() {
        return carburant;
    }

    public void periodiciteVidange(){
        switch (carburant) {
            case diesel : periodiciteVidange = 10; break;
            case essence: periodiciteVidange = 12; break;
            case gaz:     periodiciteVidange = 18; break;
            default:      periodiciteVidange = 12; break;
        }
    }

    public void replieLeToit() {toitReplie = true;}

    public int getPeriodiciteeVidange() {return periodiciteVidange;}

    public String toString() {
        return "Voiture " + super.toString() +
```

```
        ", type du carburant: "+carburant+", periodicite de vidange:
"+periodiciteVidange +" mois";
    }

}
```

```
import java.io.*;

public class Parc {
    public static void main(String[] args) throws IOException /*,
ClassNotFoundException */{

        // modifieur static caractérise la classe càd qu'on peut appliquer la
méthodes Créer sur la classe sans créer l'objet
        Voiture v1 = Voiture.creerVoiture(2013,Carburant.diesel);
        Voiture v2 = Voiture.creerVoiture(2014,Carburant.essence);

        v1.addCarburant(30); v1.periodiciteVidange();
        System.out.println(v1.toString());
        v2.addCarburant(50); v2.periodiciteVidange();
        System.out.println(v2.toString());

        //**************** Serialisation
*********************************************
        FileOutputStream fo = new FileOutputStream("garage");
        ObjectOutputStream oo = new ObjectOutputStream(fo);
        oo.writeObject(v1);
        oo.writeObject(v2);
        oo.close(); fo.close();

        //**************** Dé-Serialisation
*********************************************
       /* FileInputStream fi = new FileInputStream ("garage");
        ObjectInputStream oi = new ObjectInputStream(fi);
        Voiture voiture1 = (Voiture)oi.readObject();
        //Voiture voiture2 = (Voiture)oi.readObject();
        oi.close(); fi.close();
        System.out.println(voiture1.toString());*/

    }
}
```