Classe abstraite

Une **méthode** est **abstraite** si elle déclarée avec son prototype, mais n'est pas définie dans sa classe (fonction membre virtuelle pure en C++)

Une méthode abstraite ne peut pas être déclarée static ou private ou final.

Rappel : Une méthode finale ne peut pas être redéfinie dans une sous-classe.

Si une classe possède au moins une méthode abstraite alors elle est **abstraite**

Une classe est aussi abstraite si elle est qualifiée d'abstraite sans pour autant contenir de méthodes abstraites: public abstract Class A { … }.

Une classe abstraite ne peut pas être instanciée.

Son extension par héritage (classe concrète) doit définir toutes les méthodes abstraites pour pouvoir être instanciée.

Une classe abstraite a uniquement pour rôle de **généraliser** d'autres classes en devenant ainsi **classe parente** de ces autres classes

Elle permet ainsi de mettre en oeuvre le **polymorphisme** tout en n'étant pas nécessairement justifiée dans le cadre de la modélisation du monde réel.

Les méthoder abstraites *(abstract)* sont des méthodes qui ne comportent pas de définition et doivent par conséquent êrre définies dans les sous-classes

On rencontre des méthodes *abtract* 'à l'intérieur des classes *abstract* mais aussi des **interfaces.**

Toutes les méthodes abstraites doivent être redéfinies dans les sous-classes sous-peine d'obtenir des sous-classes abstraites.

CLASSE ABSTRAITE VEHICULE

```
package automobile;

abstract public class Vehicule
{
	private int roues;
	private float poids;

	protected String conducteur;

	public Vehicule(int roues, float poids, String conducteur)
	{
		this.roues = roues;
		this.poids = poids;
		this.conducteur = conducteur;
	}

	public int getRoues() { return roues; }
	public float getPoids() { return poids; }
	public String getConducteur() { return conducteur; }

	public void setRoues(int roues) { this.roues = roues; }
	public void setPoids(int poids) { this.poids = poids;}
	public void setConducteur(String conducteur) { this.conducteur =
conducteur; }


	abstract void message(String message);

	public String toString() { return ("roues =  " + roues + " poids= " +
poids + " conducteur: " + conducteur); }
}
```

CLASSE CONCRETE VOITURE

```
package automobile;

public class Voiture extends Vehicule
{
        private int nombrePassagers;

        public Voiture(int roues, float poids, String conducteur, int
nombrePassagers)
        {
              super(roues, poids, conducteur);
              this.nombrePassagers = nombrePassagers;
        }

        public int getNombrePassagers() { return nombrePassagers; }

        public void setNombrePassagers(int nombrePassagers)
{ this.nombrePassagers = nombrePassagers; }

        public void message(String message)
         {
              System.out.println("Message d'un voiture: " +
message);
         }

        public String toString() { return (super.toString() + "
nombrePassagers: " + nombrePassagers); }


        public static void main(String argv[])
        {
              Voiture renault = new Voiture(4, 3500.f, "Stroustrup", 5);
              System.out.println("la voiture Renault " +
renault.toString());
              renault.message("Renault la marque au losange");

              Voiture x = renault;
              x.message("La voiture x est une référence sur la Renault");
              System.out.println("la voiture x " + x.toString());

              Voiture peugeot = new Voiture(4, 4500.f, "Sun", 6);
              System.out.println("la voiture peugeot " +
peugeot.toString());

              renault.setNombrePassagers(8);

              Vehicule group [] = { renault, x, peugeot };

              System.out.println("polymorphisme pour les
éléments du tableau group");

              for (int i = 0; i < 3; i++)
              {
                    group[i].message(group[i].toString());
              }
        }
}
```