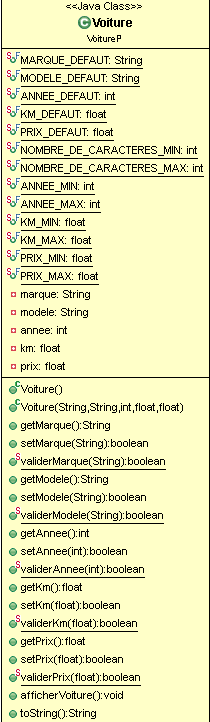
**Exercice- Classe et Objet**

**1- Diagramme de la classe Voiture**



**2- Code à compléter de la classe Voiture**

**package** VoitureP;

**import** javax.swing.JOptionPane;

/\*\*

\* Classe représentant une voiture avec marque, modèle, année, km et prix.

\*

\*/

**public** **class** Voiture

{

// Constantes de classe

/\*\*

\* marque par défaut

\*/

**public** **static** **final** String ***MARQUE\_DEFAUT*** = "marque";

/\*\*

\* modèle par défaut: modele

\*/

..

/\*\*

\* Année par défaut: 1900

\*/

....

/\*\*

\* Kilométrage par défaut: 0

\*/

...

/\*\*

\* Prix par défaut: 0

\*/

**....**

/\*\*

\* Nombre de caractères minimum pour la chaîne marque et modèle: 2

\*/

**...**

/\*\*

\* Nombre de caractères maximum pour la chaîne marque et modèle: 25

\*/

**...**

/\*\*

\* Année minimum: 1900

\*/

**....**

/\*\*

\* Année maximum: 2015

\*/

**...**

/\*\*

\* Kilométrage minimum: 0

\*/

**.....**

/\*\*

\* Kilométrage maximum: 500000

\*/

......

/\*\*

\* Prix minimum: 0

\*/

**...**

/\*\*

\* Prix maximum: 300000

\*/

**.....**

// Attributs de l'objet

/\*\*

\* Le nom de la marque de la voiture

\*/

**....**

/\*\*

\* Le nom du modèle de la voiture

\*/

**.....**

/\*\*

\* L'année de la voiture

\*/

**.....**

/\*\*

\* Le kilométrage de la voiture

\*/

**...**

/\*\*

\* Le prix de la voiture

\*/

**....**

/\*\*

\* Constructeur voiture par défaut

\*/

**public** Voiture()

{

**........**

}

/\*\*

\* Constructeur voiture avec les attributs reçus en entrée.

\*

\* **@param** paramMarque une string, la marque de la voiture

\* **@param** paramModele une string, le modèle de la voiture

\* **@param** paramAnnee un entier, l'année de la voiture

\* **@param** paramKm un réel, le kilométrage de la voiture

\* **@param** paramPrix un réel, le prix de la voiture

\*/

**public** Voiture(String paramMarque, String paramModele, **int** paramAnnee,

**float** paramKm, **float** paramPrix)

{

// Valider les paramètres reçus et les assigner aux attributs de

// l'objet, s'ils sont tous valides.

**.......**

// Sinon, attribuer les valeurs par défaut aux attributs de l'objet

**..........**

}

/\*\*

\* Retourner la marque de la voiture.

\*

\* **@return** string, la marque de la voiture

\*/

**...........**

/\*\*

\* Modifier la marque de la voiture à partir de celle reçue en entrée.

\*

\* **@param** paramMarque une string, la marque de la voiture

\*

\* **@return** boolean, indiquant vrai si la marque a été modifiée

\*/

**public** **boolean** setMarque(String paramMarque)

{

// Valider la marque reçue en entrée

**...**

// Si elle est valide, attribuer sa valeur à l'attribut marque

**....**

**return** ok;

}

/\*\*

\* Vérifier si la marque reçue en entrée est valide.

\*

\* **@param** paramMarque une string, la marque de la voiture

\*

\* **@return** boolean, indiquant vrai si c’est une marque valide.

\*/

**public** **static** **boolean** validerMarque(String paramMarque)

{

**return** ((paramMarque.length() >= ***NOMBRE\_DE\_CARACTERES\_MIN***) && (paramMarque.length() <= ***NOMBRE\_DE\_CARACTERES\_MAX***));

}

/\*\*

\* Retourner le modèle de la voiture.

\*

\* **@return** string, le modèle de la voiture

\*/

**.......**

/\*\*

\* Modifier le modèle de la voiture à partir de celui reçu en entrée.

\*

\* **@param** paramModele une string, le modèle de la voiture

\*

\* **@return** boolean, indiquant vrai si le modèle a été modifié.

\*/

**public** **boolean** setModele(String paramModele)

{

// Valider le modèle reçu en entrée

**....**

// S'il est valide, attribuer sa valeur à l'attribut modele

**.........**

}

/\*\*

\* Vérifier si le modèle reçu en entrée est valide.

\*

\* **@param** paramModele une string, le modèle de la voiture

\*

\* **@return** boolean, indiquant vrai si c’est un modèle valide.

\*/

**public** **static** **boolean** validerModele(String paramModele)

{

**.................**

}

/\*\*

\* Retourner l’année de la voiture.

\*

\* **@return** int, l'année de la voiture

\*/

**..............**

/\*\*

\* Modifier l’année de la voiture à partir de celle reçue en entrée.

\*

\* **@param** paramAnnee un entier, l'année de la voiture

\*

\* **@return** boolean, indiquant vrai si l’année a été modifiée.

\*/

**public** **boolean** setAnnee(**int** paramAnnee)

{

// Valider l’année reçue en entrée

**.....**

// Si elle est valide, attribuer sa valeur à l'attribut annee

**...................**

}

/\*\*

\* Vérifier si l’année reçue en entrée est valide.

\*

\* **@param** paramAnnee un entier, l'année de la voiture

\*

\* **@return** boolean, indiquant vrai si c’est une année valide

\*/

**public** **static** **boolean** validerAnnee(**int** paramAnnee)

{

**.........................**

}

/\*\*

\* Retourner le kilométrage de la voiture.

\*

\* **@return** float, le kilométrage actuel de la voiture.

\*/

**.........................**

/\*\*

\* Modifier le kilométrage de la voiture à partir de celui reçu en entrée.

\*

\* **@param** paramKm un réel, le kilométrage de la voiture

\*

\* **@return** boolean, indiquant vrai si le kilométrage a été modifié.

\*/

**public** **boolean** setKm(**float** paramKm)

{

// Valider le kilométrage reçu en entrée

**.**

// S'il est valide, attribuer sa valeur à l'attribut km

**.............**

}

/\*\*

\* Vérifier si le kilométrage reçu en entrée est valide.

\*

\* **@param** paramKm un réel, le kilométrage de la voiture

\*

\* **@return** boolean, indiquant vrai si c’est un kilométrage valide.

\*/

**public** **static** **boolean** validerKm(**float** paramKm)

{

**.....................**

}

/\*\*

\* Retourner le prix de la voiture.

\*

\* **@return** float, le prix de la voiture

\*/

**...........**

/\*\*

\* Modifier le prix de la voiture à partir de celui reçu en entrée.

\*

\* **@param** paramPrix un réel, le prix de la voiture

\*

\* **@return** boolean, indiquant vrai si le prix a été modifié.

\*/

**public** **boolean** setPrix(**float** paramPrix)

{

// Valider le prix reçu en entrée

**..........**

// S'il est valide, attribuer sa valeur à l'attribut prix

**.....................**

}

/\*\*

\* Vérifier si le prix reçu en entrée est valide.

\*

\* **@param** paramPrix un réel, le prix de la voiture.

\*

\* **@return** boolean, indiquant vrai si c’est un prix valide.

\*/

**public** **static** **boolean** validerPrix(**float** paramPrix)

{

**..................**

}

/\*\*

\* Afficher les attributs d’une voiture

\*/

**public** **void** afficherVoiture()

{

System.***out***

.println(**this**.getMarque() + "\t" + **this**.getModele() + "\t"

+ **this**.getAnnee() + "\t" + **this**.getKm() + "\t"

+ **this**.getPrix());

}

/\*\*

\* Permet d'afficher le contenue de l'objet Voiture

\*

\* **@return** string, une chaîne présentant les valeurs des attributs d'un

\* objet Voiture

\*

\* **@see** java.lang.Object#toString()

\*/

**public** String toString()

{

**return** **this**.getMarque() + "\t" + **this**.getModele() + "\t"

+ **this**.getAnnee() + "\t" + **this**.getKm() + "\t" + **this**.getPrix();

}

}

3- Code de la classe TestVoiture

**package** VoitureP;

**public** **class** TestVoiture

{

/\*\*

\* Faire des essais unitaires sur la classe

\*

\* **@param** args un tableau de string, les arguments reçus en entrée du

\* programme

\*/

**public** **static** **void** main(String[] args)

{

// Appel de la méthode statique avec le nom de la classe

// même à l’intérieur de la classe

System.***out***.println(Voiture.*validerKm*(3500));

// instancier une voiture par défaut

Voiture maVoiture = **new** Voiture();

maVoiture.afficherVoiture();

// Tester tous les cas possibles de toutes les méthodes :

// Tester les 2 cas valide :

Voiture voitA = **new** Voiture("Honda", "Civic", 2008, 10000, 30200);

System.***out***.println(voitA.getMarque() + " " + voitA.getModele() + " "

+ voitA.getAnnee() + " " + voitA.getKm() + " "

+ voitA.getPrix());

Voiture voitB = **new** Voiture("Nisan", "Sentra", 2000, 1000, 30000);

voitB.afficherVoiture();

// Méthode toString appelée par défaut par la JVM

System.***out***.println(voitA);

// Méthode toString appelée par défaut par la JVM

System.***out***.println(voitB);

// Tester les 2 cas de marque invalide :

Voiture voit1 = **new** Voiture("M", "Civic", 2000, 1000, 30000);

System.***out***.println(voit1.getMarque() + " " + voit1.getModele() + " "

+ voit1.getAnnee() + " " + voit1.getKm() + " "

+ voit1.getPrix());

Voiture voit2 = **new** Voiture("Marqueeeeeeeeeeeeeeeeeeeee", "Civic",

2000, 1000, 30000);

voit2.afficherVoiture();

// Tester les 2 cas de modèle invalide :

Voiture voit3 = **new** Voiture("Honda", "C", 2000, 1000, 30000);

voit3.afficherVoiture();

Voiture voit4 = **new** Voiture("Honda", "Civicccccccccccccccccccccc",

2000, 1000, 30000);

voit4.afficherVoiture();

// Tester les 2 cas d’année invalide :

Voiture voit5 = **new** Voiture("Honda", "Civic", 1899, 1000, 30000);

voit5.afficherVoiture();

Voiture voit6 = **new** Voiture("Honda", "Civic", 2010, 1000, 30000);

voit6.afficherVoiture();

}

}

**4- Résultat**

true

marque modele 1900 0.0 0.0

Honda Civic 2008 10000.0 30200.0

Nisan Sentra 2000 1000.0 30000.0

Honda Civic 2008 10000.0 30200.0

Nisan Sentra 2000 1000.0 30000.0

marque modele 1900 0.0 0.0

marque modele 1900 0.0 0.0

marque modele 1900 0.0 0.0

marque modele 1900 0.0 0.0

marque modele 1900 0.0 0.0

Honda Civic 2010 1000.0 30000.0