

# Rapport

## Représenté par :

Mohamed Amine Kouki

Ayoub Zerdoum

## Groupe :

1ere info –group C

## Matière:

Programmation

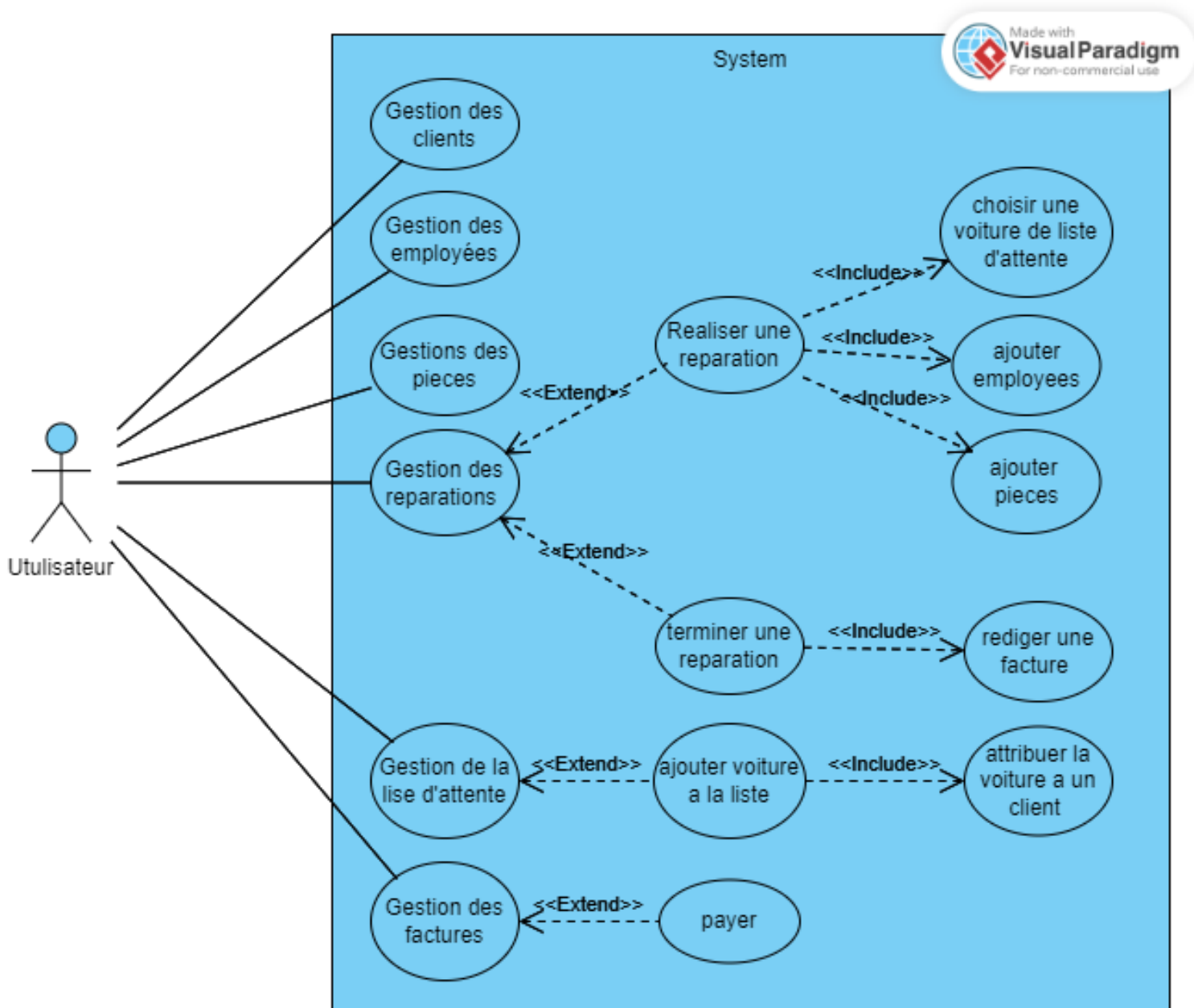
## Projet :

**Programme C++ de gestion  
d'un garage automobile**

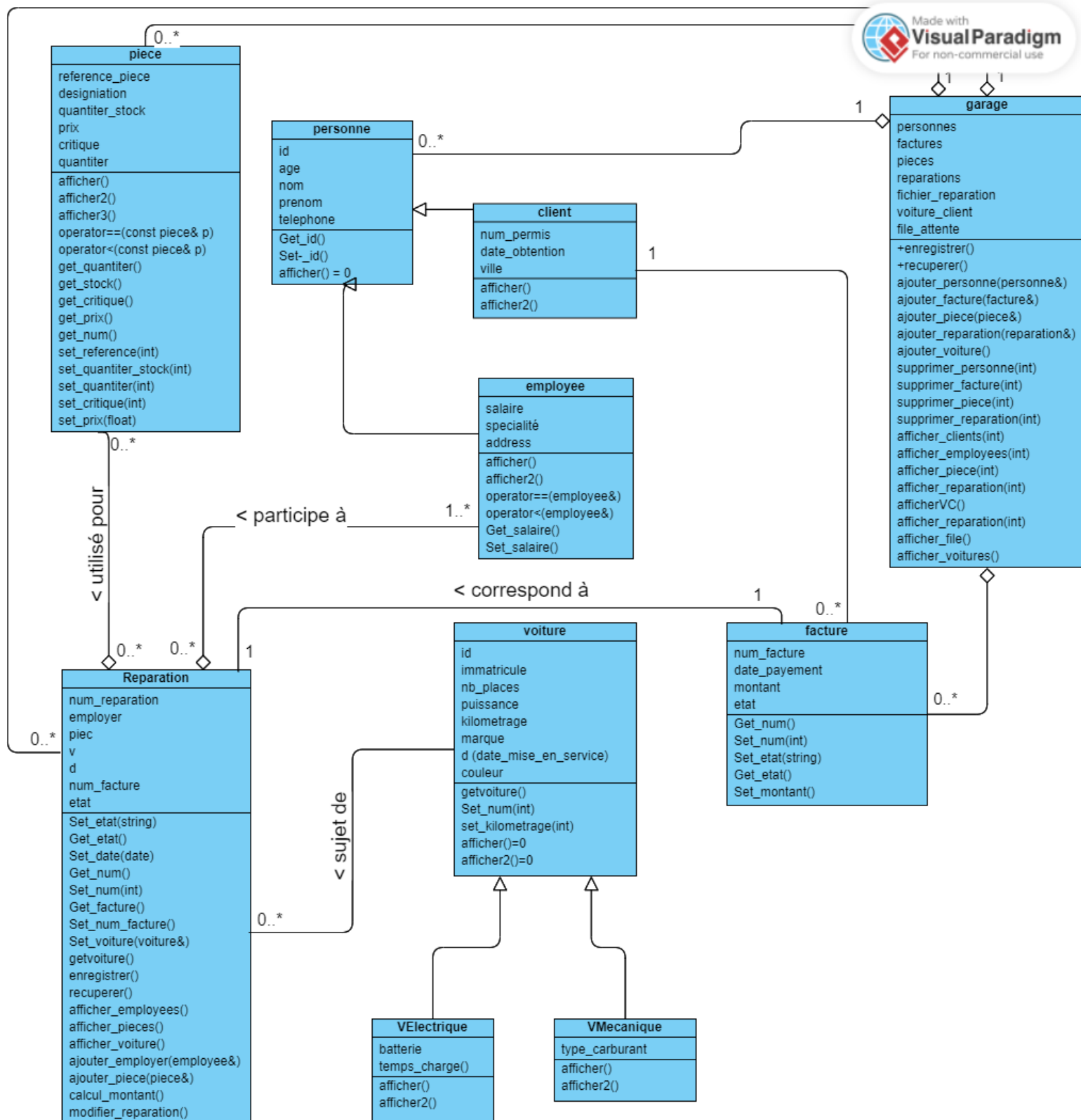
# Description du projet :

C'est un programme de **gestion d'un garage automobile** qui permet de gérer les employés, les clients, les pièces, les factures. Ainsi qu'effectuer des réparations sur les voitures électriques et mécaniques.

## I. Le diagramme de cas d'utilisation :



## II. Le diagramme de classe :



# III. Le fonctionnement du programme:

## 1. Gestion des clients:

### Deux type d'affichage des clients :

Affichage raccourci : offre une meilleure visibilité pour l'utilisateur avec les infos nécessaires.

```
----- Menu CLIENT-----  
taper 1 pour afficher la liste rapide des client  
taper 2 pour afficher la liste complete des client  
taper 3 pour ajouter un client  
taper 4 pour supprimer un client  
taper 5 pour modifier un client  
taper 6 pour rechercher un client  
taper 0 pour retourner au menu principale  
1  
1 hafeth gadacha 789  
2 mahmoud elfelhaoui 502  
3 mehdi bouzid 425
```

Affichage complet : donne tous les infos des clients.

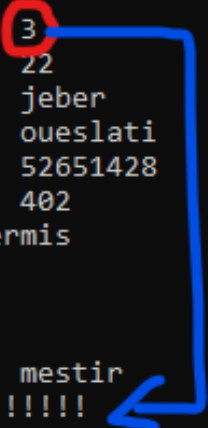
```
2  
----- Client -----  
l'id est : 1  
l'age est : 32  
le nom est : hafeth  
le prenom est : gadacha  
le numero du telephone est : 26748961  
le numero du permis est : 789  
la date est : 14/12/2015  
la ville est : nabeul  
  
----- Client -----  
l'id est : 2  
l'age est : 40  
le nom est : mahmoud  
le prenom est : elfelhaoui  
le numero du telephone est : 26785632  
le numero du permis est : 502  
la date est : 20/3/2012  
la ville est : beja  
  
----- Client -----  
l'id est : 3  
l'age est : 26  
le nom est : mehdi  
le prenom est : bouzid  
le numero du telephone est : 50756148  
le numero du permis est : 425  
la date est : 7/1/2017  
la ville est : mehdia
```

**Ajout d'un client:** avec contrôle de saisie, si le client existe déjà il ne sera pas ajouté

```
nouveau choix      3
----- saisir un client -----
saisir l'id          4
saisir l'age         22
saisir le nom        jeber
saisir le prenom     oueslati
saisir le numero du telephone 52651428
saisir le numero du permis 402
saisir la date d'obtention du permis
saisir le jour      15
saisir le mois      6
saisir l'annee2001
saisir la ville      mestir

taper 0 pour terminer
0
nouveau choix      1
1 hafeth gadacha 789
2 mahmoud elfelhaoui 502
3 mehdi bouzid 425
4 jeber oueslati 402
nouveau choix
```

```
nouveau choix      3
----- saisir un client -----
saisir l'id          3
saisir l'age         22
saisir le nom        jeber
saisir le prenom     oueslati
saisir le numero du telephone 52651428
saisir le numero du permis 402
saisir la date d'obtention du permis
saisir le jour      15
saisir le mois      6
saisir l'annee2001
saisir la ville      mestir
!!!!!!! le client existe deja !!!!!!!
#####
```



## Modification d'un client:

On affiche la liste raccourci des clients afin que l'utilisateur puisse choisir un id du client,

si l'id saisi correspond a un client le dernier va être afficher pour donner à l'utilisateur une idée sur l'état précédent du client, après il sera modifier.

**Alert !** L'id du client sera conservé pour des raisons d'intégrité.

```
nouveau choix      5
1 hafeth gadacha 789
2 mahmoud elfelhaoui 502
3 mehdi bouzid 425
4 jeber oueslati 402
-----+++-----
----- saisir l'id de client a modifier -----
4
saisir l'id          55555 ✗
saisir l'age         26
saisir le nom        ayoub
saisir le prenom     menzli
saisir le numero du telephone 72526378
saisir le numero du permis 403
saisir la date d'obtention du permis
saisir le jour      5
saisir le mois      3
saisir l'annee1997
saisir la ville      bizert
taper 0 pour terminer
0
nouveau choix      1
1 hafeth gadacha 789
2 mahmoud elfelhaoui 502
3 mehdi bouzid 425
4 ayoub menzli 403
nouveau choix
```

```
nouveau choix      5
1 hafeth gadacha 789
2 mahmoud elfelhaoui 502
3 mehdi bouzid 425
4 ayoub menzli 403
-----+++-----
----- saisir l'id de client a modifier -----
6
!!! le client n'existe pas !!!
taper 0 pour terminer
```

## Suppression d'un client:

On affiche la liste raccourci des clients afin que l'utilisateur puisse choisir un id du client, si l'id saisie correspond a un client le dernier va être supprimé.

```
nouveau choix      4
1 hafeth gadacha 789
2 mahmoud elfelhaoui 502
3 mehdi bouzid 425
4 ayoub menzli 403
-----+++-----
----- saisir l'id de client a supprimer -----
3
supprimé
taper 0 pour terminer
0
nouveau choix      1
1 hafeth gadacha 789
2 mahmoud elfelhaoui 502
4 ayoub menzli 403
nouveau choix
```

```
nouveau choix      4
1 hafeth gadacha 789
2 mahmoud elfelhaoui 502
4 ayoub menzli 403
-----+++-----
----- saisir l'id de client a supprimer -----
5
!!! le client n'existe pas !!!
taper 0 pour terminer
```

## La recherche d'un client:

On demande l'id du client, si le client existe il sera afficher

```
taper 0 pour terminer
1
----- saisir l'id de client a afficher -----
1
l'id est : 1
l'age est : 32
le nom est : hafeth
le prenom est : gadacha
le numero du telephone est : 26748961
le numero du permis est : 789
la date est : 14/12/2015
la ville est : nabeul
taper 0 pour terminer
```

```
nouveau choix      6
-----+++-----
----- saisir l'id de client a afficher -----
55
!!! le client n'existe pas !!!
taper 0 pour terminer
```

**Remarque :** tous les changements sont enregistrer dans un fichier

## 2. Gestion des employés:

### Deux types d'affichage des employés :

Affichage raccourci : offre une meilleure visibilité pour l'utilisation avec les infos nécessaires.

```
----- Menu EMPLOYEE-----  
taper 1 pour afficher la liste rapide des employees  
taper 2 pour afficher la liste complete des employees  
taper 3 pour ajouter un employee  
taper 4 pour supprimer un employee  
taper 5 pour modifier un employee  
taper 6 pour rechercher un employee  
taper 0 pour retourner au menu principale  
1  
1  ayoub  zerdoum  700  mec  
2  aziz   ben_massoud  600  elec  
nouveau choix
```

Affichage complet : donne tous les infos des clients.

```
nouveau choix 2  
----- Employee -----  
l'id est : 1  
l'age est : 25  
le nom est : ayoub  
le prenom est : zerdoum  
le numero du telephone est : 50718137  
la salaire est : 700  
la specialité est : mec  
l'address est : tunis  
  
----- Employee -----  
l'id est : 2  
l'age est : 23  
le nom est : aziz  
le prenom est : ben_massoud  
le numero du telephone est : 50265411  
la salaire est : 600  
la specialité est : elec  
l'address est : bizerte  
  
nouveau choix
```



**Ajout d'un employé:** avec contrôle de saisie, si l'employé existe déjà il ne sera pas ajouté

```
nouveau choix      3
----- saisir un employee -----
saisir l'id          3
saisir l'age         29
saisir le nom        mohamed
saisir le prenom     kouki
saisir le numero du telephone 50456187
saisir la salaire    850
saisir la specialité tol
saisir l'address     nabeul

taper 0 pour terminer
0
nouveau choix      1
1 ayoub  zerdoum    700  mec
2 aziz   ben_massoud 600  elec
3 mohamed kouki     850  tol
nouveau choix
```

```
nouveau choix      1
1 ayoub  zerdoum    700  mec
2 aziz   ben_massoud 600  elec
3 mohamed kouki     850  tol
nouveau choix      3
----- saisir un employee -----
saisir l'id          3
saisir l'age         23
saisir le nom        Zied
saisir le prenom     Salhi
saisir le numero du telephone 50475287
saisir la salaire    550
saisir la specialité mec
saisir l'address     jerba
!!!!!!! le employee existe deja !!!!!!!
taper 0 pour terminer
```

### Modification d'un employé:

On affiche la liste raccourci des employés afin que l'utilisateur puisse choisir un id de l'employé,

Si l'id saisie correspond a un employé le dernier va être affiché pour donner à l'utilisateur une idée sur l'état précédent de l'employé, après il sera modifier.

**Alert !** L'id sera conservé pour des raisons d'intégrité.

```
nouveau choix      5
1 ayoub  zerdoum    700  mec
2 aziz   ben_massoud 600  elec
3 mohamed kouki     850  tol
-----+++-----
----- saisir l'id de employee a modifier -----
3
saisir l'id          4 X
saisir l'age         23
saisir le nom        Zied
saisir le prenom     Salhi
saisir le numero du telephone 50475287
saisir la salaire    550
saisir la specialité mec
saisir l'address     jerba
taper 0 pour terminer
0
nouveau choix      1
1 ayoub  zerdoum    700  mec
2 aziz   ben_massoud 600  elec
3 Zied   Salhi     550  mec
nouveau choix
```

```
nouveau choix      5
1 ayoub  zerdoum    700  mec
2 aziz   ben_massoud 600  elec
3 Zied   Salhi     550  mec
-----+++-----
----- saisir l'id de employee a modifier -----
14
!!! l'employee n'existe pas !!!
taper 0 pour terminer
```



## Suppression d'un employé:

On affiche la liste raccourci des employés afin que l'utilisateur puisse choisir un id de l'employé, si l'id saisie correspond a un employé le dernier va être supprimé.

```
nouveau choix      4
1 ayoub  zerdoum    700  mec
2 aziz   ben_massoud 600  elec
3 Zied   Salhi      550  mec
-----+++-----
----- saisir l'id de employee a supprimer --
3
supprimé
taper 0 pour terminer
0
nouveau choix      1
1 ayoub  zerdoum    700  mec
2 aziz   ben_massoud 600  elec
nouveau choix
```

```
nouveau choix      4
1 ayoub  zerdoum    700  mec
2 aziz   ben_massoud 600  elec
-----+++-----
----- saisir l'id de employee a supprimer --
3
!!! le employee n'existe pas !!!
taper 0 pour terminer
```

## La recherche d'un employé:

On demande l'id de l'employé, si il existe il sera afficher.

```
nouveau choix      6
-----+++-----
----- saisir l'id de employee a afficher
1
l'id est : 1
l'age est : 25
le nom est : ayoub
le prenom est : zerdoum
le numero du telephone est : 50718137
la salaire est : 700
la specialité est : mec
l'address est : tunis
taper 0 pour terminer
```

### 3. Gestion des pièces:

#### Deux types d'affichage des pièces :

Affichage raccourci : offre une meilleure visibilité pour l'utilisation avec les infos nécessaires.

```
----- Menu PIECE-----
taper 1 pour afficher la liste rapide des pieces
taper 2 pour afficher la liste complete des pieces
taper 3 pour ajouter une piece
taper 4 pour supprimer une piece
taper 5 pour modifier une piece
taper 6 pour mise a jour la quantité en stock une piece
taper 7 pour rechercher une piece
taper 0 pour retourner au menu principale
1
1001  AZ300    200    15
1002  M40     500    5.2
1003  G120    20     120
nouveau choix
```

Affichage complet : donne tous les infos relatives à la pièce.

```
nouveau choix    2
----- Piece -----

reference de piece est :1001
designation :AZ300
quantiter dans les stock :200
prix de la piece 15
critique de stock50
quantiter qui sera utiliser 12

----- Piece -----

reference de piece est :1002
designation :M40
quantiter dans les stock :500
prix de la piece 5.2
critique de stock100
quantiter qui sera utiliser 25

----- Piece -----

reference de piece est :1003
designation :G120
quantiter dans les stock :20
prix de la piece 120
critique de stock10
quantiter qui sera utiliser 1

nouveau choix
```

**Ajout d'une pièce:** avec contrôle de saisie, si la pièces existe déjà il ne sera pas ajouté et la quantité est mise a zéro

```
nouveau choix      3
----- saisir une piece -----

reference de piece est :1004

designation :LH50

quantiter dans les stock :800

prix de la piece 2.3

critique de stock  200

taper 0 pour terminer
0
nouveau choix      1
1001  AZ300   200   15
1002  M40     500   5.2
1003  G120    20   120
1004  LH50    800   2.3
nouveau choix
```

```
nouveau choix      3
----- saisir une piece -----

reference de piece est :1004

designation :ad500

quantiter dans les stock :825

prix de la piece 5.6

critique de stock  120
!!!!!!! la piece existe deja !!!!!!!
taper 0 pour terminer
```

### Modification d'une pièce:

On affiche la liste raccourci des employés afin que l'utilisateur puisse choisir un id du client,

Si la référence saisie correspond a une pièce le dernier va être afficher pour donner à l'utilisateur une idée sue l'état précédent de la pièce, après il sera modifier.

**Alert !** La référence sera conservée pour des raisons d'intégrité.

```
nouveau choix      5
1001  AZ300   200   15
1002  M40     500   5.2
1003  G120    20   120
1004  LH50    800   2.3
-----+++-----
----- saisir la reference du piece a modifier
1004

reference de piece est :1005

designation :ML550

quantiter dans les stock :750

prix de la piece 7.8

critique de stock  70
taper 0 pour terminer
0
nouveau choix      1
1001  AZ300   200   15
1002  M40     500   5.2
1003  G120    20   120
1004  ML550   750   7.8
nouveau choix
```

```
nouveau choix      5
1001  AZ300   200   15
1002  M40     500   5.2
1003  G120    20   120
1004  ML550   750   7.8
-----+++-----
----- saisir la reference du piece a modifier
1000
!!! la piece n'existe pas !!!

taper 0 pour terminer
```

## Suppression d'une pièce:

On affiche la liste raccourci des pièces afin que l'utilisateur puisse choisir une référence de la pièce, si la référence saisie correspond a une pièce le dernier va être supprimé.

```
nouveau choix      4
1001  AZ300    200    15
1002  M40     500    5.2
1003  G120     20   120
1004  ML550    750    7.8
-----+++-----
----- saisir la reference du piece a supprimer
1004
supprimée
taper 0 pour terminer
0
nouveau choix      1
1001  AZ300    200    15
1002  M40     500    5.2
1003  G120     20   120
nouveau choix
```

```
nouveau choix      4
1001  AZ300    200    15
1002  M40     500    5.2
1003  G120     20   120
-----+++-----
----- saisir la reference du piece a supprimer
1000
!!! la piece n'existe pas !!!
taper 0 pour terminer
```

## Mise a jour du stock:

On affiche la liste raccourci des pièces afin que l'utilisateur puisse choisir une référence de la pièce, si la référence saisie correspond à une pièce, on demande le saisie de la quantité à ajouter.

```
nouveau choix      6
1001  AZ300    200    15
1002  M40     500    5.2
1003  G120     20   120
-----+++-----
----- saisir la reference du piece -----
1001
saisir la quantité a ajouter : 50
la nouvelle quantite en stock est: 250
taper 0 pour terminer
0
nouveau choix      1
1001  AZ300    250    15
1002  M40     500    5.2
1003  G120     20   120
nouveau choix
```

```
nouveau choix      6
1001  AZ300    250    15
1002  M40     500    5.2
1003  G120     20   120
-----+++-----
----- saisir la reference du piece -----
1000
!!! la piece n'existe pas !!!
taper 0 pour terminer
```

```
nouveau choix      6
1001  AZ300    250    15
1002  M40     500    5.2
1003  G120     20   120
-----+++-----
----- saisir la reference du piece -----
1001
saisir la quantité a ajouter : -1
ERREUR - la quantite en stock doit etre positive
saisir la quantité a ajouter :
a
ERREUR - quantite en stock non valide
saisir la quantité a ajouter :
```

## La recherche d'une pièce:

On demande la référence de la pièce, si elle existe il sera afficher.

```
nouveau choix      7
-----+++-----
----- saisir la reference du piece a afficher
1005

reference de piece : 1005
sa designation  ML550
quantiter en stock  750
sa prix  7.8
sa critique stock  70
quantiter 0taper 0 pour terminer
```

## 4. Gestion de la liste d'attentes des voitures:

### Deux types d'affichage des voitures :

Affichage raccourci : offre une meilleure visibilité pour l'utilisation avec les infos nécessaires.

```
5
----- Menu Voiture-----
taper 1 pour afficher la liste rapide des voitures en attentes
taper 2 pour afficher la liste complete des voitures en attentes
taper 3 pour ajouter une voiture a la file d'attentes
taper 4 pour rechercher toutes les voitures
taper 5 pour modifier une voiture
taper 6 pour rechercher proprietaire
taper 0 pour retourner au menu principale
1
8005  mat1  tesla  b1
8006  mat2  audi   gasoil
nouveau choix
```

Affichage complet : donne tous les infos relatives à la voiture.

On affiche les voitures présente a la liste d'attente // ou en affiche les voitures déjà traité

```
nouveau choix      2
----- affichage file -----

id de voiture est  8005
sa matricule :mat1
nombre de place :4
sa puissance 100
kilometre :200
marque:tesla
date_mise_en_service :10/10/2020
couleur est noire
le type de batterieb1
temp de charge10
----- affichage file -----

id de voiture est  8006
sa matricule :mat2
nombre de place :4
sa puissance 400
kilometre :200
marque:audi
date_mise_en_service :15/12/2020
couleur est rouge
type de carburantgasoil
nouveau choix
```

**Ajout d'une voiture a la liste** : on demande le type de voiture (électrique ou mécanique), après en demande l'id de client propriétaire a la voiture

```
5
----- Menu Voiture-----
taper 1 pour afficher la liste rapide des voitures en attentes
taper 2 pour afficher la liste complete des voitures en attentes
taper 3 pour ajouter une voiture a la file d'attentes
taper 4 pour rechercher toutes les voitures
taper 5 pour modifier une voiture
taper 6 pour rechercher proprietaire
taper 0 pour retourner au menu principale
3
donner le type de voiture E:elecrique / M:mecanique
E

id de Voiture Electrique est 8005

sa matricule :mat1

nombre de place :4

sa puissance 100

kilometre :200

marque:tesla

date_mise_en_service :saisir le jour 10
saisir le mois 10
saisir l'annee2020

couleur est noire

le type de batterieb1

temp de charge10
1 hafeth gadacha 789
2 mahmoud elfelhaoui 502
3 mehdi bouzid 425
donner l'id du client proprietaire de cette voiture
1
```



## Modification d'une voiture:

On affiche la liste raccourci des voitures en cours de réparation afin que l'utilisateur puisse choisir un id de la voiture,

Si l'id saisie correspond a une voiture déjà traité il sera modifier.

**Alert !** L'id sera conservée pour des raisons d'intégrité.

```
nouveau choix      5
2001  T200  tesla    batterie1 ←
3001  EGY56324  audi    essence
-----++-----
donner le numero du voiture a changer
2001

quele type de voiture qui sera choisit E electrique//

id de Voiture Mecanique est  8006

sa matricule :mat2

nombre de place :4

sa puissance 100

kilometre :200

marque:tesla

date_mise_en_service :saisir le jour  10
saisir le mois      10
saisir l'annee2021

couleur est vert
le type de carburantessence
nouveau choix      1
8005  mat1  tesla    b1
8006  mat2  audi     gasoil
nouveau choix      5
2001  mat2  tesla    essence ←
3001  EGY56324  audi    essence
```

## La recherche de tous les voiture (déjà traité):

```
4
2001  mat2  tesla    essence
3001  EGY56324  audi    essence
```

## La recherche du propriétaire d'une voiture:

On demande l'id de la voiture, si elle existe le client propriétaire sera affiché.

```
6
2001  T200  tesla  batterie1
3001  EGY56324  audi  essence
-----+++-----
donner le numero du voiture
2001
avant
apres
l'id est : 1
l'age est : 32
le nom est : hafeth
le prenom est : gadacha
le numero du telephone est : 26748961
le numero du permis est : 789
la date est : 14/12/2015
la ville est : nabeul
nouveau choix
```

## 5. Gestion des réparations:

### Deux types d'affichage des réparations :

Affichage raccourci : offre une meilleure visibilité pour l'utilisation avec les infos nécessaires.

```
nouveau choix      1
1      2001    employees:2    pieces:3    active
2      3001    employees:2    pieces:2    active
nouveau choix
```

Affichage complet : donne tous les infos relatives à la réparation.

```
*****affichage de reparation*****
le numero du reparation est :2
la date de reparation :      20/3/2012
aucune facture
la voiture est :

id de voiture mecniue est  3001
sa matricule :EGY56324
nombre de place :4
sa puissance 300
kilometre :156
marque:audi
date_mise_en_service :7/1/2017
couleur est gris
type de carburantessence
les employees participant a la reparation :
employe 1:

l'id est : 2
l'age est : 23
le nom est : aziz
le prenom est : ben_massoud
le numero du telephone est : 50265411
la salaire est : 600
la specialit  est : elec
l'address est : bizerte

employe 1:

l'id est : 3
l'age est : 29
le nom est : mohamed
le prenom est : kouki
le numero du telephone est : 50456187
la salaire est : 850
la specialit  est : tol
l'address est : nabeul

les pieces utlise dans la reparation est :Piece 1:

reference de piece est :1001
designation :AZ300
quatiter dans les stock :200
prix de la piece 15
critique de stock50
quantiter qui sera utiliser 12
Piece 2:

reference de piece est :1003
designation :G120
quatiter dans les stock :20
prix de la piece 120
critique de stock10
quantiter qui sera utiliser 1
la reparation est active

la montant de reparation 1750
```

```
*****affichage de reparation***
le numero du reparation est :1
la date de reparation :      14/12/2015
aucune facture
la voiture est :

id de voiture electrique est  2001
sa matricule :T200
nombre de place :4
sa puissance 100
kilometre :242
marque:tesla
date_mise_en_service :20/3/2012
couleur est bleu
la type de batteriebatterie1
temp de charge10
les employees participant a la reparation :
employe 1:

l'id est : 2
l'age est : 23
le nom est : aziz
le prenom est : ben_massoud
le numero du telephone est : 50265411
la salaire est : 600
la specialit  est : elec
l'address est : bizerte

employe 1:

l'id est : 1
l'age est : 25
le nom est : ayoub
le prenom est : zerdoum
le numero du telephone est : 50718137
la salaire est : 700
la specialit  est : mec
l'address est : tunis

les pieces utlise dans la reparation est :Piece 1:

reference de piece est :1002
designation :M40
quatiter dans les stock :500
prix de la piece 5.2
critique de stock100
quantiter qui sera utiliser 25
Piece 2:

reference de piece est :1001
designation :AZ300
quatiter dans les stock :200
prix de la piece 15
critique de stock50
quantiter qui sera utiliser 12
Piece 3:

reference de piece est :1003
designation :G120
quatiter dans les stock :20
prix de la piece 120
critique de stock10
quantiter qui sera utiliser 1
la reparation est active

la montant de reparation 1730
```

## Démarrer une réparation :

On demande les infos de la réparation,

**Alert !** Le numéro de facture est mise à zéro car la réparation est active.

**Alert !** L'état de réparation est mise à « active » car la réparation est active.

On affiche la voiture suivante dans la liste d'attente voiture, si l'utilisateur ne veut pas démarrer avec cette voiture, elle sera supprimée de la liste d'attente et la voiture suivante sera affichée

Si l'utilisateur valide la voiture on passe à l'ajout des employés,

On affiche la liste raccourcie des employés disponibles, l'utilisateur choisit l'employé,

Il y a possibilité de supprimer le dernier employé ajouté (en utilise une classe Template pile)

Si la saisie est terminée on ajoute les employés à la réparation

On passe à l'ajout des pièces,

On affiche la liste raccourcie des pièces disponibles, l'utilisateur choisit les pièces,

Il y a possibilité de supprimer la dernière pièce ajoutée (en utilise une classe Template pile)

Si la saisie est terminée on ajoute les pièces à la réparation avec la mise à jour des quantités en stock et déclenchement d'alarme si la quantité en stock est inférieure à la valeur critique.

```

3
8005  mat1  tesla    b1
----- +++ -----
demarrer la reparation avec cette voiture? Y/N    :N
8006  mat2  audi     gasoil
----- +++ -----
demarrer la reparation avec cette voiture? Y/N    :Y
donner le numero du reparation est : 7701
donner la date de reparation :      saisir le jour   2
saisir le mois      5
saisir l'annee2023
1  ayoub  zerdoum    700  mec
2  aziz   ben_massoud 600  elec
3  mohamed kouki     850  tol
----- +++ -----
choisir l'employee : 1
pour terminer le saisie des employee taper 0
pour annuler l'employee precedent taper 1
3
choisir l'employee : 3
pour terminer le saisie des employee taper 0
pour annuler l'employee precedent taper 1
1
l'employee suivant a ete annulé :
3  mohamed kouki     850  tol
choisir l'employee : 2
pour terminer le saisie des employee taper 0
pour annuler l'employee precedent taper 1
0
employer ajoutez a la liste
employer ajoutez a la liste
1001  AZ300  200    15
1002  M40    500    5.2
1003  G120   20     120
----- +++ -----
choisir les pieces : 1001
saisir la quantité qui sera utilisée : 20
pour terminer le saisie des pieces taper 0
pour annuler la piece precedent taper 1
5
choisir les pieces : 1003
saisir la quantité qui sera utilisée : 50
ERREUR - la quantite en stock n'est pas suffisante
saisir la quantité qui sera utilisée : 2
pour terminer le saisie des pieces taper 0
pour annuler la piece precedent taper 1
1
la piece suivant a ete annule :
1003  G120  18     120
choisir les pieces : 1002
saisir la quantité qui sera utilisée : 450
ALARM - la quantité en stock de la piece va bientôt expire
pour terminer le saisie des pieces taper 0
pour annuler la piece precedent taper 1
0
piece ajoutez a la liste
piece ajoutez a la liste

```

## Terminer une réparation :

On affiche la liste raccourci des réparations active, on demande à l'utilisateur de choisir la réparation à terminer

Si la réparation existe, on remplit une nouvelle facture,

**Alert !** Le numéro de facture sera ajouté à la réparation

**Alert !** L'état de réparation est mise à « terminé »

**Alert !** La date de paiement de facture est mise à 0/0/0 et son état « non payée »

La facture sera ajoutée à la liste des factures du garage

```
7
1   2001   employees:2   pieces:3   active
2   3001   employees:2   pieces:2   active
----- +++ -----
choisir le numero du reparation a terminer
2
saisir le numero du facture      4501
saisir le montant du facture     0000
la facture est payée ? Y/N       N
nouveau choix
```

```
----- Menu FACTURE-----
taper 1 pour afficher la liste rapide des factures
taper 2 pour afficher la liste complete des factures
taper 3 pour modifier une facture
taper 4 pour rechercher une facture
taper 5 pour payee une facture
taper 0 pour retourner au menu principale
1
1   500    500    payee
2   650    650    non_payee
3   210    210    payee
4501 1750  1750    non_payee ←
nouveau choix
```

## La suppression d'une réparation:

On demande le numéro de la réparation, si elle existe elle sera affichée.

```
1   2001   employees:2   pieces:3   active
----- +++ -----
saisir le numero de reparation a supprimer2
la reparation est terminée - suppression impossible
+++
```

## La recherche d'une réparation:

On affiche la liste raccourci des réparations active et, on demande le numéro de la réparation, si elle existe elle sera affichée.

### Modification d'une réparation:

On affiche la liste raccourci des réparations afin que l'utilisateur puisse choisir un numéro de réparation à modifier,

Si le numéro saisi correspond à une réparation **active** la dernière va être affichée pour donner à l'utilisateur une idée sur l'état précédent de la réparation, après il sera modifier.

Il y a possibilité d'ajouter des employés

Il y a possibilité d'ajouter des pièces

```

si tu peut modifier voiture donner tapez 1
modifier piece tapez 2
modifier employer tapez 3 :2

saisir une piece
reference de piece est :1001

designation :LH50

quatiter dans les stock :800

prix de la piece 2.3

critique de stock 200
Piece:
reference de piece est :1002
designation :M40
quatiter dans les stock :500
prix de la piece 5.2
critique de stock100
quantiter qui sera utiliser 25
Piece:
reference de piece est :1001
designation :LH50
quatiter dans les stock :800
prix de la piece 2.3
critique de stock200
quantiter qui sera utiliser 0
Piece:
reference de piece est :1003
designation :G120
quatiter dans les stock :20
prix de la piece 120
critique de stock10
quantiter qui sera utiliser 1

si tu peut ajouter autre modification tapez 1 si non 0 0
nouveau choix

```

Il ya possibilité de modifier la voiture

```

si tu peut modifier voiture donner tapez 1
modifier piece tapez 2
modifier employer tapez 3 :1

quele type de voiture qui sera choisit 1 electrique 2 mecanque 1

id de Voiture Electrique est 8006

sa matricule :mat2

nombre de place :4

sa puissance 100

kilometre :200

marque:tesla

date_mise_en_service :saisir le jour 10
saisir le mois 10
saisir l'annee2021

couleur est vert

le type de batterieb2

temp de charge11

si tu peut ajouter autre modification tapez 1 si non 0 0
nouveau choix 0

```

```

----- saisir le numero de reparation a afficher -
1
NumÚro de rÚparation:      1
Date de rÚparation:       14/12/2015
Etat:                      active
aucune facture
La voiture est :

id de voiture est 2001
sa matricule :mat2
nombre de place :4
sa puissance 100
kilometre :200
marque:tesla
date_mise_en_service :10/10/2021
couleur est vert
le type de batterieb2
temp de charge11

```

**Alert !** Le numéro & numéro facture & état ne peut pas être modifié.



## 6. Gestion des factures:

### Deux types d'affichage des factures :

Affichage raccourci : offre une meilleure visibilité pour l'utilisation avec les infos nécessaires.

```
----- Menu FACTURE-----  
taper 1 pour afficher la liste rapide des factures  
taper 2 pour afficher la liste complete des factures  
taper 3 pour modifier une facture  
taper 4 pour rechercher une facture  
taper 5 pour payee une facture  
taper 0 pour retourner au menu principale  
1  
1 500 14/12/2015 payee  
2 650 0/0/0 non_payee  
3 210 7/1/2017 payee  
nouveau choix
```

Affichage complet : donne tous les infos relatives à la facture.

```
nouveau choix 2  
----- Facture -----  
le numero du facture est : 1  
le montant est : 500  
la date du payement est : 14/12/2015  
la facture est payée  
  
----- Facture -----  
le numero du facture est : 2  
le montant est : 650  
la facture n'est payée  
  
----- Facture -----  
le numero du facture est : 3  
le montant est : 210  
la date du payement est : 7/1/2017  
la facture est payée
```

## Modification d'une facture:

On affiche la liste raccourci des factures afin que l'utilisateur puisse choisir un num du facture,

Si le numéro saisi correspond a une facture **non payée** la dernière va être affichée pour donner à l'utilisateur une idée sue l'état précédent de la facture, après il sera modifier.

**Alert !** Le numéro sera conservé pour des raisons d'intégrité.

```
nouveau choix      3
1   500    14/12/2015    payee
2   650     0/0/0       non_payee
3   210    7/1/2017     payee
----- +++ -----
saisir le numero de facture a modifier1
la facture est payee - modification impossible
+++
```

```
nouveau choix      3
1  500    14/12/2015    payee
2  650    0/0/0        non_payee
3  210    7/1/2017     payee
-----  +++  -----
saisir le numero de facture a modifier2
l'etat actuel de la facture est :
le numero du facture est : 2
le montant est : 650
la facture n'est pay  
saisir le numero du facture      5555  *
saisir le montant du facture     4000   
la facture est pay   ? Y/N      Y
saisir la date :
saisir le jour      2
saisir le mois      5
saisir l'annee2023
l'etat actuel de la facture est :
le numero du facture est : 2
le montant est : 4000
la facture n'est pay  
```

## La recherche d'une facture:

On demande le numéro de la facture, si elle existe il sera afficher.

```
nouveau choix      4
-----+++-----
----- saisir le numero de facture a afficher
1
le numero du facture est : 1
le montant est : 500
la date du paiement est : 14/12/2015
la facture est payée
taper 0 pour terminer
```

## Payer une facture:

On affiche la liste raccourci des factures afin que l'utilisateur puisse choisir un num de facture à payer,

Si la facture existe et n'a pas été payé, on met l'état à «payé » et on demande la date de paiement.

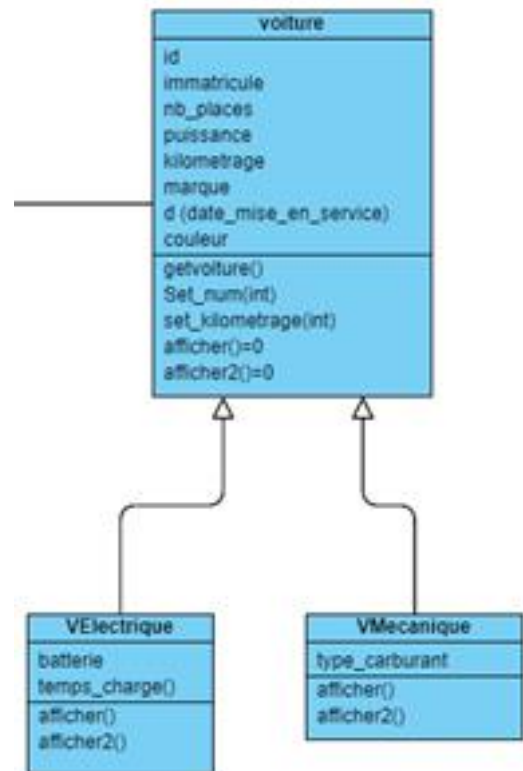
```
5
2  650    0/0/0    non_payee
-----+++-----
----- saisir le numero de facture a payee
2
saisir le jour      2
saisir le mois      5
saisir l'annee2023
le numero du facture est : 2
le montant est : 650
la date du paiement est : 2/5/2023
la facture est payée
taper 0 pour terminer
0
nouveau choix      1
1  500    14/12/2015    payee
2  650    2/5/2023     payee
3  210    7/1/2017     payee
nouveau choix
```

## IV. Evaluation du programme:

### 1. héritage:



Réalisé par : Ayoub Zerdoum



Réalisé par : Med Amine Kouki

### 2. polymorphisme:

```
474 // ----- class garage ----- //
475 class garage
476 {
477     vector<personne*> personnes; ←
478     vector<facture*> factures;
479     list<piece*> pieces;
480     vector<reparation*> reparations;
481     map<int,string> fichier_reparation;
482     vc voiture_client;
483     Queue<voiture*> file_attente;
484 public:
485     garage(){};
486     ~garage();
487     garage(const garage&);
488
489     virtual void ajouter_personne(personne&); ←
```

La classe garage contient un tableau dynamique composé des objets de type **personne** (en réalité **client** & **employée**)

```

902 void garage::ajouter_personne (personne& p)
903 {
904     personne *x;
905     if (typeid(p)==typeid(client))
906         x=new client (static_cast<const client&>(p));
907     else if (typeid(p)==typeid(employee))
908         x=new employee (static_cast<const employee&>(p));
909     personnes.push_back(x);
910 }

```

De même pour la classe voiture : dans la classe réparation il existe un pointeurs sur un objet voiture ( en réalité Electrique & Mécanique)

```

422 // ----- class reparation -----
423 class reparation
424 {
425     int num_reparation;
426     vector <employee*> employer;
427     vector <piece*> piec;
428     voiture* v; ←
429     date d;
430     int num_facture;
431     string etat;
432 public:
433     reparation(){};
434     reparation(const reparation& );
435     reparation(int, date, string);
436
437     void Set_vehicule(voiture &); ←
438

```

```

1939 void reparation::Set_vehicule(voiture &ve)
1940 {
1941     if (typeid(ve)==typeid(VElectrique))
1942         v=new VElectrique (static_cast<const VElectrique&>(ve));
1943     else if (typeid(ve)==typeid(VMechanique))
1944         v=new VMechanique (static_cast<const VMechanique&>(ve));
1945 }
1946

```

### 3. Surcharge des opérateurs:

Operateur >> et << pour tous les classes :

```
71      friend ostream& operator<<(ostream&,client&);
72      friend istream& operator>>(istream&,client&);
73
74      friend ostream& operator<<(ostream&,client*);
75      friend istream& operator>>(istream&,client*);
```

Operateur == et < :

```
89      bool operator==(const employee& e) const {return (id == e.id);}
90      bool operator<(const employee& e) const {return salaire < e.salaire;}
```

### 4. Gestion des fichiers :

La méthode d'enregistrement et récupération des données .

La classe garage contient toutes les données relatives au programme.

- En écrit les clients et les employés dans le même fichier, on les distingue par un nombre au début de la ligne (2 :pour les client /3 : pour les employés), le nom de fichier est « FichierPersonnes »

```
765      creer_fichier_personne(fl);
766
767      for(int i=0; i<personnes.size();i++)
768      {
769          if(typeid(*personnes[i]) == typeid(client))
770              fl<<"2 "<<&static_cast<client>(*personnes[i])<<endl;
771          else if(typeid(*personnes[i]) == typeid(employee))
772              fl<<"3 "<<&static_cast<employee>(*personnes[i])<<endl;
773      }
774      fl.close();
```

Enregistrement

fichierPersonne - Bloc-notes

Fichier	Edition	Format	Affichage	Aide						
3	1	25	ayoub	zerdoum	50718137	700	mec	tunis	Fichier	
3	2	23	aziz	ben_massoud	50265411	600	elec	bizerte		
3	3	29	mohamed	kouki	50456187	850	tol	nabeul		
2	1	32	hafeth	gadacha	26748961	789	14	12	2015	nabeul
2	2	40	mahmoud	elfelhaoui	26785632	502	20	3	2012	beja
2	3	26	mehdi	bouزيد	50756148	425	7	1	2017	mehdia

```

814 f1.open("d:\\fichierPersonne.text", ios::in);
815 if(!f1.is_open()) exit(-1);
816 while(1)
817 {
818     f1>>val;
819     if(f1.eof()) Récupérer
820         break;
821     if(val == 2)
822     {
823         client* c =new client;
824         f1>>*c;
825         personnes.push_back(c);
826     }
827     else if(val == 3)
828     {
829         employee* e =new employee;
830         f1>>*e;
831         personnes.push_back(e);
832     }
833 }
834 f1.close();


```

- En écrit les pièces dans un fichier « FichierPieces »

```

783
784     creer_fichier_piece(f3);
785     for(auto pi :pieces) Enregistrement
786         f3<<"1 " <<*pi<<endl;
787     f3.close();

```

 fichierPiece - Bloc-notes

Fichier	Edition	Format	Affichage	Aide			
1	1001	AZ300	200	15	50	12	<b>Fichier</b>
1	1002	M40	500	5.2	100	25	
1	1003	G120	20	120	10	1	

```

849 f3.open("d:\\fichierPiece.text", ios::in);
850 if(!f3.is_open()) exit(-1);
851 while(1)
852 {
853     f3>>val;
854     if(f3.fail())
855         break;
856     piece* p = new piece;
857     f3>>*p;
858     pieces.push_back(p);
859 }
860 f3.close();

```

**Récupérer**



- En écrit les factures dans un fichier « FichierFactures »

```

776     creer_fichier_factures(f2);
777     for(int i=0; i<factures.size();i++)
778     {
779         f2<<i<<" ";
780         f2<<&*factures[i]<<endl;
781     }
782     f2.close();

```

Enregistrement

fichierFacture - Bloc-notes

Fichier	Edition	Format	Affichage	Aide
0	1	500	payee	14 12 2015
1	2	650	non_payee	2 5 2023
2	3	210	payee	7 1 2017

Fichier

```

836     f2.open("d:\\fichierFacture.text", ios::in);
837     if(!f2.is_open()) exit(-1);
838     while(1)
839     {
840         f2>>val;
841         if(f2.fail())
842             break;
843         facture* f = new facture;
844         f2>>&*f;
845         factures.push_back(f);
846     }
847     f2.close();

```

Récupérer

- Pour les reparations , on enregistre chaque reparation dans sa propre fichier « Reparation + numero du reparation », la correspondance entre numero reparation et sa fichier est enregistrer dans une **map** , en peut recuprer les reparations à l'aide de cette map,

fichierPiece	02/05/2023 01:51	Fichier TEXT	1 Ko
fichierReparation	02/05/2023 01:51	Fichier TEXT	1 Ko
fichierReparation1	02/05/2023 01:51	Fichier TEXT	1 Ko
fichierReparation2	02/05/2023 01:51	Fichier TEXT	1 Ko
fichierVC	02/05/2023 01:51	Fichier TEXT	1 Ko

Fichier

\*fichierReparation1 - Bloc-notes

Fichier	Edition	Format	Affichage	Aide
0	1	14	12	2015
2	2001	T200	4	100
1	1	25	ayoub	zerdoum
1	2	23	aziz	ben_massoud
2	1001	AZ300	200	15
2	1002	M40	500	5.2
2	1003	G120	20	120

```

789 creer_fichier_reparations(f4);
790 for(int i=0; i<reparations.size();i++)
791 {
792     reparations[i]->enregistrer(i);
793     pair<int,string> c(reparations[i]->Get_num(),"d:\\fichierReparation" + to_string(reparations[i]->Get_num()) + ".text");
794     fichier_reparation.insert(c);
795 }
796 map<int,string>::iterator it;
797 for(it=fichier_reparation.begin() ; it!=fichier_reparation.end() ; ++it)
798 {
799     f4<<"0 " <<it->first<<" " <<it->second<<endl;
800 }
801 f4.close();

```

Enregistrement

```

863 pair<int,string> p;
864 f4.open("d:\\fichierReparation.text", ios::in);
865 while(1)
866 {
867     f4>>val;
868     if(f4.fail())
869         break;
870     f4>>p.first;
871     f4>>p.second;
872     fichier_reparation.insert(p);
873 }
874 f4.close();
875
876 reparation r;
877 map<int,string>::iterator it;
878 for(it=fichier_reparation.begin() ; it!=fichier_reparation.end() ; ++it)
879 {
880     r.recuperer(it->first);
881     reparation *x =new reparation(r);
882     reparations.push_back(x);
883 }

```

Récupérer

- La correspondance entre voiture et client sera aussi enregistrer dans un fichier « voitureClient »

```

803 creer_fichier_vc(f5);
804 f5<<&voiture_client;
805 f5.close();

```

Enregistrement



fichierVC - Bloc-notes

Fichier	Edition	Format
0	2001	1
0	3001	2

Fichier

```

885 int v,c;
886 f5.open("d:\\fichierVC.text", ios::in);
887 while(1)
888 {
889     f5>>val;
890     if(f5.fail())
891         break;
892     f5>>v;
893     f5>>c;
894     voiture_client.ajoutermap(v,c);
895 }
896 f5.close();

```

Récupérer

- La file d'attente ne sera pas enregistrer
- la récupération des voitures est réalisé au même temps que les réparations, il sera redondant de les re\_enregistrer dans un autre fichier.

## Le changement dans le fichier :

Il est possible de réalisé des changements sur les fichiers de réparation

```

2270 void reparation::modifier_reparation_fichier(int aa)
2271 {
2272     reparation r;
2273     fstream f;
2274     string ch;
2275     ch = "fichierReparation" + to_string(aa) + ".text";
2276     f.open(ch, ios::in);
2277     r.recuperer(aa);
2278     f.close();
2279     f.open(ch, ios::in|ios::trunc);
2280     r.modifier_reparation();
2281     r.enregistrer(aa);
2282     f.close();
2283 }

```

## 5. Gestion des exceptions :

Exemple (saisie de classe client) :

```

185 istream& operator>>(istream& i, client& c)
186 {
187     string msg;
188     string msg1="ERREUR - le client existe deja";
189     string msg2="ERREUR - l'id doit etre un entier";
190     string msg3="ERREUR - l'id doit etre positive";
191     string msg4="ERREUR - l'age doit etre positive";
192     string msg5="ERREUR - l'age doit etre superieur a 18";
193     string msg6="ERREUR - l'age doit etre un entier";
194     string msg7="ERREUR - telephone non valide";
195     string msg8="ERREUR - le telephone est composé de 8 caractères";
196     string msg9="ERREUR - le numero de permis doit etre positive";
197     string msg10="ERREUR - le numero non valide";
198     while(true)
199     {
200         try
201         {
202             cout<<"saisir l'id" <<endl;
203             i>>c.id;
204             if(cin.fail()) throw 1;
205             if(c.id<=0) throw msg3;
206             break;
207         }
208         catch(string msg)
209         {
210             cout<<msg<<endl;
211         }
212         catch(int i)
213         {
214             cout <<msg2<<endl;
215             cin.clear();
216             cin.ignore(numeric_limits<streamsize>::max(), '\n');
217             continue;
218         }
219     }
220 }

```

## 6. Template :

**Template pile** : elle sera utilisé dans le démarrage d'une réparation pour offrir une possibilité de retour en arrière lors de saisie des employés et des pièces, le dernière employé choisit pourrait être annulé (voir démarrage de réparation)

```
124 //----- template pile -----,
125 template<class T>
126 struct element
127 {
128     T info;
129     element* suivant;
130 };
131
132 template<class T>
133 class pile
134 {
135     element<T>* sommet;
136 public:
137     pile(){sommet=NULL;};
138     ~pile()
139     {
140
141
142
143
144     void empiler(T t)
145     {
146
147
148
149
150
151
152     T depiler()
153     {
154
155
156
157
158
159
160
161
162
163
164     bool estVide(){return(sommet==NULL);};
165
166     int NbElements()
167     {
168
169
170
171
172
173
174
175
176
177     void afficher()
178     {
179
180
181
182
183
184
185
186
187
188
189
190
191
192     pile(const pile<T>& w)
193     {
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210     pile& operator=(pile<T>& w)
211     {
212
213
214
215
216
217
218
219
220
221
222
223
224     };
```

**Template Queue :** elle sera utilisé pour la file d'attente des voitures, la première voiture qui entre, va sortir la première

```
397 //----- template file -----//
398 template<typename T>
399 class Queue {
400 private:
401     vector<T> data;
402 public:
403     void push(const T& item) {data.push_back(item);}
404
405     void pop() {
406         if (!empty()) {data.erase(data.begin());}
407     T& front() {return data.front();}
408     T& back() {return data.back();}
409     bool empty() const {return data.empty();}
410     size_t size() const {return data.size();}
411
412     void copy(Queue<voiture*>& source, Queue<voiture*>& destination)
413     {
414         while (!source.empty())
415         {
416             destination.push(source.front());
417             source.pop();
418         }
419     }
420 };
```

**Template VC :** rassemble à une map , elle est utilisé pour enregistrer le couple « id\_voiture ,id\_client ». Elle nous permet de savoir le propriétaire d'une voiture.

```
340 //----- class map -----//
341 class vc
342 {
343 public:
344     map<int,int> a;
345     map<int,int>::iterator it;
346 public:
347     vc(){};
348     vc(const vc& );
349     void ajoutermap(int,int);
350     void affichermap();
351     void modifiermap(int,int);
352     int val(int v){return a[v];};
353     void recherchermap(int);
354     friend ostream& operator<<(ostream& os, const vc& obj) {
355
356
357     friend istream& operator>>(istream& is, vc& obj) {
358
359     friend ostream& operator<<(ostream& os, const vc* obj)
360     {
361
362     friend istream& operator>>(istream& is, vc* obj)
363     {
364
365     void suprimmap(int);
366 };
```

## 7. classe abstraite :

Les classes **personne & voiture** sont des classe abstraits : il ne seront pas utilisés dans l'exécution du programme.

## 8. conteneurs séquentiels :

Vector & list :

```
470 class garage
471 {
472     vector<personne*> personnes;
473     vector<facture*> factures;
474     list<piece*> pieces;
475     vector<reparation*> reparations;
```

## 9. conteneur associatif (map) :

```
341 class vc
342 {
343     public:
344         map<int,int> a;
345         map<int,int>::iterator it;
346         vector<reparation*> reparations;
347         map<int,string> fichier_reparation;
348         vc voiture_client;
```

## 10. algorithmes :

Sort & find :

```
1927 auto foundIterator = find(employer.begin(), employer.end(), em);
1988 sort(employer.begin(), employer.end(), [](employee* c, employee* b) { return *c < *b; });
```

## 11. tableaux dynamiques d'objets :

Classe garage :

Classe réparation :

```
470 class garage
471 {
472     vector<personne*> personnes;
473     vector<facture*> factures;
474     list<piece*> pieces;
475     vector<reparation*> reparations;
419 // ----- class reparation ---
420 class reparation
421 {
422     int num_reparation;
423     vector <employee*> employer;
424     vector <piece*> piec;
```

## V. header.h:

```
#ifndef HEADER10_H_INCLUDED
#define HEADER10_H_INCLUDED

#include <iostream>
#include <vector>
#include <string>
#include <fstream>
#include <iomanip>
#include <map>
#include <algorithm>
#include <stack>
#include <list>
#include <exception>
#include <limits>

using namespace std;
// ----- class date ----- //
class date
{
    int jour;
    int mois;
    int annee;
public:
    date(int,int,int);
    date(){};
    friend ostream& operator<<(ostream&,date&);
    friend istream& operator>>(istream&,date&);

    friend ostream& operator<<(ostream&,date*);
    friend istream& operator>>(istream&,date*);
};

// ----- class personne ----- //
class personne
{
protected:
    int id;
```



```

    int age;
    string nom;
    string prenom;
    int telephone;
public:
    personne(int,int,string,string,int);
    personne(){};
    personne(const personne&);
    int Get_id(){return id;};
    void Set_id(int i){this->id =i;};
    virtual void afficher();
    virtual void afficher2()= 0;
    void Set_telephone(int t){this->telephone=t;};
    friend ostream& operator<<(ostream&,personne&);
    friend istream& operator>>(istream&,personne&);

    friend ostream& operator<<(ostream&,personne*);
    friend istream& operator>>(istream&,personne*);
};

// ----- class client ----- //
class client:public personne
{
    int num_permis;
    date date_obtention;
    string ville;
public:
    client(int,int,string,string,int,int,date,string);
    client(){};
    void afficher();
    void afficher2(){cout<<id<<" "<<nom<<" "<<prenom<<"
"<<num_permis<<endl;};
    friend ostream& operator<<(ostream&,client&);
    friend istream& operator>>(istream&,client&);

    friend ostream& operator<<(ostream&,client*);
    friend istream& operator>>(istream&,client*);
};

// ----- class employee ----- //
class employee:public personne

```

```

{
    float salaire;;
    string specialite;
    string address;
public:
    employee(int,int,string,string,int,float,string,string);
    employee(){};
    void afficher();
    void afficher2(){cout<<id<<" "<<nom<<" "<<prenom<<"
"<<salaire<<" "<<specialite<<endl;};
    bool operator==(const employee& e) const {return (id == e.id);}
    bool operator<(const employee& e) const {return salaire <
e.salaire;}
    float Get_salaire(){return salaire;};
    void Set_salaire(float s){this->salaire=s;};
    friend ostream& operator<<(ostream&,employee&);
    friend istream& operator>>(istream&,employee&);

    friend ostream& operator<<(ostream&,employee*);
    friend istream& operator>>(istream&,employee*);
};

// ----- class facture ----- //
class facture
{
    int num_facture;
    date date_payement;
    float montant;
    string etat;
public:
    facture(int,date,float,string);
    facture(){};
    void afficher2(){cout<<num_facture<<" "<<montant<<"
"<<date_payement<<" "<<etat<<endl;};
    int Get_num(){return num_facture;};
    void Set_num(int i){num_facture = i;};
    void Set_etat(string e){this->etat=e;};
    string Get_etat(){return etat;};
    void Set_date(date& d){date_payement = d;}
    void Set_montant(float m){montant=m;};
    friend ostream& operator<<(ostream&,facture&);

```

```

friend istream& operator>>(istream&,facture&);

friend ostream& operator<<(ostream&,facture*);
friend istream& operator>>(istream&,facture*);
};

//----- template pile -----//
template<class T>
struct element
{
    T info;
    element* suivant;
};

template<class T>
class pile
{
    element<T>* sommet;
public:
    pile(){sommet=NULL;};
    ~pile()
    {
        while(sommet!=NULL)
            depiler();
    };

    void empiler(T t)
    {
        element<T> *e = new element<T>;
        e->suivant=sommet;
        e->info=t;
        sommet=e;
    };

    T depiler()
    {
        if(sommet != NULL )
        {
            element<T>*e =sommet;
            T data=e->info;
            sommet=sommet->suivant;

```

```

        delete e;
        return data;
    }
};

```

```

bool estVide(){return(sommet==NULL);};

```

```

int NbElements()
{
    element<T>*e=sommet;
    int n=0;
    while(e!=NULL)
    {
        e=e->suivant;
        n++;
    }
};

```

```

void afficher()
{
    if(sommet==NULL)
        cout<<"\n la pile est vide"<<endl;
    else
    {
        element<T>*e=sommet;
        while(e!=NULL)
        {
            cout<<e<<" "<<e->info<<endl;
            e=e->suivant;
        }
    }
};

```

```

pile(const pile<T>& w)
{
    sommet = NULL;
    element<T>*e;
    element<T>*courant;
    element<T>*tmp = NULL;
    courant =w.sommet;
    while(courant!=NULL)

```

```

    {
        e =new element<T>(*courant);
        if(sommet == NULL)
            sommet =e;
        else
            tmp->suivant=e;
        tmp = e;
        courant=courant->suivant;
    }
};
pile& operator=(pile<T>& w)
{
    if(this!=&w)
    {
        while(sommet!=NULL)
            depiler();
        sommet = NULL;
        element<T>*e;
        element<T>*courant;
        element<T>*tmp = NULL;
        courant =w.sommet;
        while(courant!=NULL)
        {
            e =new element<T>(*courant);
            if(sommet == NULL)
                sommet =e;
            else
                tmp->suivant=e;
            tmp = e;
            courant=courant->suivant;
        }
    }
    return *this;
};

// ----- class voiture ----- //
class voiture
{
protected:
    int id;
    string immatricule;

```

```

    int nb_place;
    int puissance;
    int kilometrage;
    string marque;
    date d;//date_mise_en_service
    string couleur;
public:
    voiture(int,string,int,int,int,string,date,string);
    voiture(){};
    voiture(const voiture&);
    int getvoiture(){return id;};
    void Set_num(int i){id = i;};
    void set_kilometrage(int a){kilometrage=a;};
    virtual void afficher();
    virtual void afficher2() = 0;

    virtual void ecrire(fstream&);

    friend ostream& operator<<(ostream&,voiture*);
    friend istream& operator>>(istream&,voiture*);

    friend ostream& operator<<(ostream&,voiture&);
    friend istream& operator>>(istream&,voiture&);
};
// ----- class voitureElectrique ----- //
class VElectrique :public voiture
{
    string batterie;
    int temps_charge;
public:
    VElectrique(){};
    // VElectrique(const VElectrique&);
    void afficher();
    void afficher2(){cout<<id<<" "<<immatricule<<" "<<marque<<"
"<<batterie<<endl;};
    VElectrique(int,string,int,int,int,string,date,string,string,int);
    void set_kilometrage(int a){kilometrage=a;};
    void ecrire(fstream&);
    friend ostream& operator<<(ostream&,VElectrique*);
    friend istream& operator>>(istream&,VElectrique*);

```

```

    friend ostream& operator<<(ostream&,VElectrique&);
    friend istream& operator>>(istream&,VElectrique&);
};

// ----- class voitureMecanique ----- //
class VMecanique : public voiture
{
    string type_carburant;
public:
    VMecanique(){};
    void afficher();
    void afficher2(){cout<<id<<" "<<immatricule<<" "<<marque<<"
"<<type_carburant<<endl;};
    VMecanique(int,string,int,int,int,string,date,string,string);
    friend ostream& operator<<(ostream&,VMecanique*);
    friend istream& operator>>(istream&,VMecanique*);
    friend ostream& operator<<(ostream&,VMecanique&);
    friend istream& operator>>(istream&,VMecanique&);
};

// ----- class piece ----- //
class piece
{
    int reference_piece;
    string designation ;
    int quantiter_stock;
    float prix;
    int critique;
    int quantiter;
public:
    piece(){};
    piece(int,string,int,float,int,int);
    void afficher();
    void afficher2(){cout<<reference_piece<<" "<<designation<<"
"<<quantiter_stock<<" "<<prix<<endl;};
    void afficher3(){cout<<reference_piece<<" "<<designation<<"
"<<prix<<" "<<quantiter<<endl;};
    bool operator==(const piece& p) const {return reference_piece ==
p.reference_piece ;}
    bool operator<(const piece& p) const {return prix< p.prix;}
    int get_quantiter(){return quantiter;};

```

```

int get_stock(){return quantiter_stock;};
int Get_critique(){return critique;};
float get_prix(){return prix;};
int Get_num(){return reference_piece;};
void Set_reference(int i){this->reference_piece=i;};
void set_quantiter_stock(int a){quantiter_stock=a;};
void set_quantiter(int a)
{ if (quantiter_stock-a<0)
    cout<<"\n\ncette piece est n'a pas available";
  else
  {
    quantiter =a;
    set_quantiter_stock(quantiter_stock-a);
  }
};
void set_critique(int a){critique=a;};
void set_prix(float a){prix=a;};
friend ostream& operator<<(ostream&,piece&);
friend istream& operator>>(istream&,piece&);
friend ostream& operator<<(ostream&,piece*);
friend istream& operator>>(istream&,piece*);
};
//----- class map -----//
class vc
{
public:
    map<int,int> a;
    map<int,int>::iterator it;
public:
    vc();
    vc(const vc& );
    void ajoutermap(int,int);
    void affichermap();
    void modifiermap(int,int);
    int val(int v){return a[v];};
    void recherchermap(int);
    friend ostream& operator<<(ostream& os, const vc& obj) {
        for (auto & [key, val] : obj.a) {
            os <<"id voiture :"<< key << " : " << "id client :"<<val << endl;
        }
        return os;
    }
};

```



```

}

friend istream& operator>>(istream& is, vc& obj) {
    int x, key, value;

    do{
        cout<<"donner id voiture/client  ";
        is >> key >> value;
        obj.a[key] = value;
        cout<<"\ntapez 1 pour ajouter autre couple  ";
        is>>x;
    }while (x);
    return is;
}

friend ostream& operator<<(ostream& os, const vc* obj)
{
    for (auto & [key, val] : obj->a) {
        os <<"0 " << key <<" " <<val << endl;
    }
    return os;
}

friend istream& operator>>(istream& is, vc* obj)
{
    int val, key, value;
    if(val == 0)
    {
        is >> key >> value;
        obj->a[key] = value;
    }
}

void suprimmap(int);
};

```

```

//----- template file -----//
template<typename T>
class Queue {
private:
    vector<T> data;
public:

```

```

void push(const T& item) {data.push_back(item);}

void pop() {
    if (!empty()) {data.erase(data.begin());}
    T& front() {return data.front();}
    T& back() {return data.back();}
    bool empty() const {return data.empty();}
    size_t size() const {return data.size();}

void copy(Queue<voiture*>& source, Queue<voiture*>& destination)
{
    while (!source.empty())
    {
        destination.push(source.front());
        source.pop();
    }
}
};

// ----- class reparation ----- //
class reparation
{
    int num_reparation;
    vector <employee*> employer;
    vector <piece*> piec;
    voiture* v;
    date d;
    int num_facture;
    string etat;
public:
    reparation(){};
    reparation(const reparation& );
    reparation(int,date,string);

    void Set_vehicule(voiture &);

    void set_etat(string e){etat =e;};
    string Get_etat(){return etat;};
    void Set_date(date d){this->d = d;};
    int Get_num(){return num_reparation;};
    void Set_num(int n){this->num_reparation = n;};

```

```

int Get_facture(){return num_facture;};
void Set_num_facture(int i){this->num_facture = i;};
int getvoiture(){return v->getvoiture();};
void afficher();
void afficher_v2();
void afficher2(){cout<<num_reparation<<" "<<v->getvoiture()<<"
employees:"<<employer.size()<<" pieces:"<<piec.size()<<"
"<<etat<<endl;};};

```

```

void enregistrer(int);
void recuperer(int);
void creer_fichier_reparation(fstream &);

```

```

void ajouter_employeur(employee&);
friend istream& operator>>(istream&,reparation*);
friend ostream& operator<<(ostream&,reparation*);
friend istream& operator>>(istream&,reparation &);
friend ostream& operator<<(ostream&,reparation &);
void afficher_employees();
void ajouter_piece(piece&);
void afficher_pieces();
void afficher_voiture();
float calcul_montant();
~reparation();

```

```

void modifier_reparation_fichier(int);
void modifier_reparation();

```

```
};
```

```
// ----- class garage ----- //
```

```
class garage
```

```
{
```

```

vector<personne*> personnes;
vector<facture*> factures;
list<piece*> pieces;
vector<reparation*> reparations;
map<int,string> fichier_reparation;
vc voiture_client;
Queue<voiture*> file_attente;

```

```
public:
```

```
garage(){};
~garage();
garage(const garage&);
```

```
virtual void ajouter_personne(personne&);
```

```
void enregistrer();
void recuperer();
static void creer_fichier_personne(fstream &);
static void creer_fichier_factures(fstream &);
static void creer_fichier_piece(fstream &);
static void creer_fichier_reparations(fstream &);
static void creer_fichier_vc(fstream &);
```

```
void ajouterCV(int v,int c){voiture_client.ajoutermap(v,c);};
```

```
void supprimer_personne(int);
void afficher_clients(int);
void afficher_employees(int);
```

```
void afficher_clients2();
void afficher_employees2();
void afficherVC(){voiture_client.affichermap();};
```

```
void ajouter_facture(facture&);
void supprimer_facture(int);
void afficher_facture(int);
void afficher_factures2();
void afficher_factures2_non_payee();
```

```
void ajouter_piece(piece&);
void supprimer_piece(int);
void afficher_piece(int);
void afficher_pieces2();
```

```
void ajouter_reparation(reparation&);
void supprimer_reparation(int);
void afficher_reparation(int);
void afficher_reparation2();
void afficher_reparation2_active();
```

```
void afficher_file2();  
void afficher_file();  
void ajouter_voiture();  
void afficher_voitures();
```

```
friend void ajouter_client(garage &g);  
friend void supprimer_client(garage &g);  
friend void modifier_client(garage &g);  
friend void recherche_client(garage &g);
```

```
friend void ajouter_employee(garage &g);  
friend void supprimer_employee(garage &g);  
friend void modifier_employee(garage &g);  
friend void recherche_employee(garage &g);
```

```
friend void ajouter_piece(garage &g);  
friend void supprimer_piece(garage &g);  
friend void modifier_piece(garage &g);  
friend void ajouter_stock_piece(garage &g);  
friend void recherche_piece(garage &g);
```

```
friend void modifier_voiture(garage&g);  
friend void recherche_proprietaire(garage&g);
```

```
friend void demarrer_reparation(garage& g);  
friend void supprimer_reparation(garage& g);  
friend void modifier_reparation(garage& g);  
friend void recherche_reparation(garage& g);  
friend void terminer_reparation(garage& g);
```

```
friend void modifier_facture(garage& g);  
friend void recherche_facture(garage& g);  
friend void payer_facture(garage& g);  
friend void menu_facture(garage& g);
```

```
friend void afficher_menu_principale(garage&g);
```

```
};
```

```
void ajouter_client(garage &g);  
void supprimer_client(garage &g);  
void modifier_client(garage &g);  
void recherche_client(garage &g);  
void menu_client(garage& g);
```

```
void ajouter_employee(garage &g);  
void supprimer_employee(garage &g);  
void modifier_employee(garage &g);  
void recherche_employee(garage &g);  
void menu_employee(garage& g);
```

```
void ajouter_piece(garage &g);  
void supprimer_piece(garage &g);  
void modifier_piece(garage &g);  
void ajouter_stock_piece(garage &g);  
void recherche_piece(garage &g);  
void menu_piece(garage& g);
```

```
void modifier_voiture(garage&g);  
void recherche_proprietaire(garage&g);  
void menu_voiture(garage& g);
```

```
void demarrer_reparation(garage& g);  
void supprimer_reparation(garage& g);  
void modifier_reparation(garage& g);  
void recherche_reparation(garage& g);  
void terminer_reparation(garage& g);  
void menu_reparation(garage& g);
```

```
void modifier_facture(garage& g);  
void recherche_facture(garage& g);  
void payer_facture(garage& g);  
void menu_facture(garage& g);
```

```
void afficher_menu_principale(garage& g);
```

```
#endif // HEADER10_H_INCLUDED
```

## VI. source.cpp:

```
#include "header10.h"
#include "algorithm"
#include <algorithm>
#include <stack>
#include <vector>
#include <list>

// ----- class date -----//
date::date(int j,int m,int a)
{
    this->jour=j;
    this->mois=m;
    this->annee=a;
}

ostream& operator<<(ostream& o,date& d)
{
    o<<d.jour<<"/"<<d.mois<<"/"<<d.annee;
    return o;
}

istream& operator>>(istream& i,date& d)
{
    string msg;
    string msg1="ERREUR - le jour doit etre superieur a 0";
    string msg2="ERREUR - le jour doit etre inferieur a 31";
    string msg3="ERREUR - le mois doit etre superieur a 0";
    string msg4="ERREUR - le mois doit etre inferieur a 12";
    string msg5="ERREUR - l'annee doit etre superieur a 0";
    while(true)
    {
        try
        {
            cout<<"saisir le jour  ";
            i>>d.jour;
            if(d.jour<=0) throw msg1;
            if(d.jour>31) throw msg2;
            break;
        }
    }
}
```

```

    }
    catch(string msg)
    {
        cout<<msg<<endl;
    }
}
while(true)
{
    try
    {
        cout<<"saisir le mois   ";
        i>>d.mois;
        if(d.mois<=0) throw msg3;
        if(d.mois>12) throw msg4;
        break;
    }
    catch(string msg)
    {
        cout<<msg<<endl;
    }
}
while(true)
{
    try
    {
        cout<<"saisir l'annee";
        i>>d.annee;
        if(d.annee<=0) throw msg5;
        break;
    }
    catch(string msg)
    {
        cout<<msg<<endl;
    }
}
return i;
}

ostream& operator<<(ostream& o,date* d)
{

```



```

        o<<setw(5)<<d->jour<<" "<<setw(5)<<d->mois<<" "<<setw(5)<<d-
>annee;
        return o;
    }

    istream& operator>>(istream& i,date* d)
    {
        i>>d->jour;
        i>>d->mois;
        i>>d->annee;
        return i;
    }

    // ----- class personne -----//
    personne::personne(int id,int age,string nom ,string prenom,int tel)
    {
        this->id=id;
        this->age=age;
        this->nom=nom;
        this->prenom=prenom;
        this->telephone=tel;
    }

    personne::personne(const personne& p)
    {
        this->id=p.id;
        this->age=p.age;
        this->nom=p.nom;
        this->prenom=p.prenom;
        this->telephone=p.telephone;
    }

    void personne::afficher()
    {
        cout<<"l'id est : "<<id<<endl;
        cout<<"l'age est : "<<age<<endl;
        cout<<"le nom est : "<<nom<<endl;
        cout<<"le prenom est : "<<prenom<<endl;
        cout<<"le numero du telephone est : "<<telephone<<endl;
    }

```

```
ostream& operator<<(ostream& o,personne& p)
{
    o<<"l'id est : "<<p.id<<endl;
    o<<"l'age est : "<<p.age<<endl;
    o<<"le nom est : "<<p.nom<<endl;
    o<<"le prenom est : "<<p.prenom<<endl;
    o<<"le numero du telephone est : "<<p.telephone<<endl;
    return o;
}
```

```
ostream& operator<<(ostream& o,personne* p)
{
    o<<setw(5)<<p->id<<" ";
    o<<setw(5)<<p->age<<" ";
    o<<setw(5)<<p->nom<<" ";
    o<<setw(5)<<p->prenom<<" ";
    o<<setw(5)<<p->telephone<<endl;
    return o;
}
```

```
istream& operator>>(istream& i,personne* p)
{
    i>>p->id;
    i>>p->age;
    i>>p->nom;
    i>>p->prenom;
    i>>p->telephone;
    return i;
}
```

```
// ----- class client -----//
client::client(int id,int age,string nom ,string prenom,int tel,int
num,date d,string ville):personne(id,age,nom,prenom,tel)
{
    this->num_permis=num;
    this->date_obtention=d;
    this->ville=ville;
}
```

```
void client::afficher()
{
    cout<<"l'id est : "<<id<<endl;
```

```

cout<<"l'age est : "<<age<<endl;
cout<<"le nom est : "<<nom<<endl;
cout<<"le prenom est : "<<prenom<<endl;
cout<<"le numero du telephone est : "<<telephone<<endl;

cout<<"le numero du permis est : "<<num_permis<<endl;
cout<<"la date est : "<<date_obtention<<endl;
cout<<"la ville est : "<<ville<<endl;
}

```

```

ostream& operator<<(ostream& o,client& c)
{
    o<<"l'id est : "<<c.id<<endl;
    o<<"l'age est : "<<c.age<<endl;
    o<<"le nom est : "<<c.nom<<endl;
    o<<"le prenom est : "<<c.prenom<<endl;
    o<<"le numero du telephone est : "<<c.telephone<<endl;

    o<<"le numero du permis est : "<<c.num_permis<<endl;
    o<<"la date est : "<<c.date_obtention<<endl;
    o<<"la ville est : "<<c.ville<<endl;
    return o;
}

```

```

istream& operator>>(istream& i,client& c)
{
    string msg;
    string msg1="ERREUR - le client existe deja";
    string msg2="ERREUR - l'id doit etre un entier";
    string msg3="ERREUR - l'id doit etre positive";
    string msg4="ERREUR - l'age doit etre positive";
    string msg5="ERREUR - l'age doit etre superieur a 18";
    string msg6="ERREUR - l'age doit etre un entier";
    string msg7="ERREUR - telephone non valide";
    string msg8="ERREUR - le telephone est composé de 8 caractères";
    string msg9="ERREUR - le numero de permis doit etre positive";
    string msg10="ERREUR - le numero non valide";
    while(true)
    {
        try

```

```

{
    cout<<"saisir l'id          ";
    i>>c.id;
    if(cin.fail()) throw 1;
    if(c.id<=0)    throw msg3;
    break;
}
catch(string msg)
{
    cout<<msg<<endl;
}
catch(int i)
{
    cout <<msg2<<endl;
    cin.clear();
    cin.ignore(numeric_limits<streamsize>::max(), '\n');
    continue;
}
}

```

```

while(true)
{
    try
    {
        cout<<"saisir l'age          ";
        i>>c.age;
        if(cin.fail()) throw 1;
        else if(c.age<0)    throw msg4;
        else if(c.age<18)    throw msg5;
        break;
    }
    catch(string msg)
    {
        cout<<msg<<endl;
    }
    catch(int i)
    {
        cout <<msg6<<endl;
        cin.clear();
        cin.ignore(numeric_limits<streamsize>::max(), '\n');
        continue;
    }
}

```

```
    }  
}
```

```
while(true)  
{  
    try  
    {  
        cout<<"saisir le nom          ";  
        i>>c.nom;  
        cout<<"saisir le prenom        ";  
        i>>c.prenom;  
        break;  
    }  
    catch(string msg)  
    {  
        cout<<msg<<endl;  
    }  
}
```

```
while(true)  
{  
    try  
    {  
        cout<<"saisir le numero du telephone  ";  
        i>>c.telephone;  
        if(cin.fail()) throw 1;  
        else if(c.telephone<10000000 || c.telephone>99999999)  
throw msg8;  
        break;  
    }  
    catch(string msg)  
    {  
        cout<<msg<<endl;  
    }  
    catch(int i)  
    {  
        cout <<msg7<<endl;  
        cin.clear();  
        cin.ignore(numeric_limits<streamsize>::max(), '\n');  
        continue;  
    }  
}
```

```
}
```

```
while(true)
```

```
{
```

```
    try
```

```
    {
```

```
        cout<<"saisir le numero du permis    ";
```

```
        i>>c.num_permis;
```

```
        if(cin.fail()) throw 1;
```

```
        else if(c.num_permis<0)    throw msg9;
```

```
        break;
```

```
    }
```

```
    catch(string msg)
```

```
    {
```

```
        cout<<msg<<endl;
```

```
    }
```

```
    catch(int i)
```

```
    {
```

```
        cout <<msg10<<endl;
```

```
        cin.clear();
```

```
        cin.ignore(numeric_limits<streamsize>::max(), '\n');
```

```
        continue;
```

```
    }
```

```
}
```

```
    cout<<"saisir la date d'obtention du permis"<<endl;
```

```
    i>>c.date_obtention;
```

```
    cout<<"saisir la ville    ";
```

```
    i>>c.ville;
```

```
    return i;
```

```
}
```

```
ostream& operator<<(ostream& o,client* c)
```

```
{
```

```
    o<<setw(5)<<c->id<<"  ";
```

```
    o<<setw(5)<<c->age<<"  ";
```

```
    o<<setw(5)<<c->nom<<"  ";
```

```
    o<<setw(5)<<c->prenom<<"  ";
```

```
    o<<setw(5)<<c->telephone<<"  ";
```

```
    o<<setw(5)<<c->num_permis<<"  ";
```

```
    o<<setw(10)<<&c->date_obtention<<"  ";
```

```

        o<<setw(5)<<c->ville<<endl;
        return o;
    }

istream& operator>>(istream& i,client* c)
{
    i>>c->id;
    i>>c->age;
    i>>c->nom;
    i>>c->prenom;
    i>>c->telephone;

    i>>c->num_permis;
    i>>&c->date_obtention;
    i>>c->ville;
    return i;
}
// ----- class employee -----//
employee::employee(int id,int age,string nom ,string prenom,int
tel,float salaire,string specialite,string
address):personne(id,age,nom,prenom,tel)
{
    this->salaire=salaire;
    this->specialite=specialite;
    this->address= address;
}

void employee::afficher()
{
    cout<<"l'id est : "<<id<<endl;
    cout<<"l'age est : "<<age<<endl;
    cout<<"le nom est : "<<nom<<endl;
    cout<<"le prenom est : "<<prenom<<endl;
    cout<<"le numero du telephone est : "<<telephone<<endl;

    cout<<"la salaire est : "<<salaire<<endl;
    cout<<"la specialité est : "<<specialite<<endl;
    cout<<"l'address est : "<<address<<endl;
}

ostream& operator<<(ostream& o,employee& e)

```

```

{
    o<<"l'id est : "<<e.id<<endl;
    o<<"l'age est : "<<e.age<<endl;
    o<<"le nom est : "<<e.nom<<endl;
    o<<"le prenom est : "<<e.prenom<<endl;
    o<<"le numero du telephone est : "<<e.telephone<<endl;

    o<<"la salaire est : "<<e.salaire<<endl;
    o<<"la specialité est : "<<e.specialite<<endl;
    o<<"l'address est : "<<e.address<<endl;
    return o;
}

```

```

istream& operator>>(istream& i,employee& e)

```

```

{
    string msg;
    string msg1="ERREUR - l'employee existe deja";
    string msg2="ERREUR - l'id doit etre un entier";
    string msg3="ERREUR - l'id doit etre positive";
    string msg4="ERREUR - l'age doit etre positive";
    string msg5="ERREUR - l'age doit etre superieur a 18";
    string msg6="ERREUR - l'age doit etre un entier";
    string msg7="ERREUR - telephone non valide";
    string msg8="ERREUR - le telephone est composé de 8 caractères";
    string msg9="ERREUR - la salaire doit etre positive";
    string msg10="ERREUR - salaire non valide";
    string msg11="ERREUR - specialite n'existe pas";

    while(true)
    {
        try
        {
            cout<<"saisir l'id          ";
            i>>e.id;
            if(cin.fail()) throw 1;
//            if(employee_existe(e.id)) throw msg1;
            if(e.id<=0)  throw msg3;
            break;
        }
        catch(string msg)
        {

```



```

        cout<<msg<<endl;
    }
    catch(int i)
    {
        cout <<msg2<<endl;
        cin.clear();
        cin.ignore(numeric_limits<streamsize>::max(), '\n');
        continue;
    }
}

```

```

while(true)
{
    try
    {
        cout<<"saisir l'age          ";
        i>>e.age;
        if(cin.fail()) throw 1;
        else if(e.age<0)   throw msg4;
        else if(e.age<18)  throw msg5;
        break;
    }
    catch(string msg)
    {
        cout<<msg<<endl;
    }
    catch(int i)
    {
        cout <<msg6<<endl;
        cin.clear();
        cin.ignore(numeric_limits<streamsize>::max(), '\n');
        continue;
    }
}

```

```

while(true)
{
    try
    {
        cout<<"saisir le nom          ";
        i>>e.nom;
    }
    catch(string msg)
    {
        cout<<msg<<endl;
    }
    catch(int i)
    {
        cout <<msg7<<endl;
        cin.clear();
        cin.ignore(numeric_limits<streamsize>::max(), '\n');
        continue;
    }
}

```

```

        cout<<"saisir le prenom          ";
        i>>e.prenom;
        break;
    }
    catch(string msg)
    {
        cout<<msg<<endl;
    }
}

while(true)
{
    try
    {
        cout<<"saisir le numero du telephone  ";
        i>>e.telephone;
        if(cin.fail()) throw 1;
        else if(e.telephone<10000000 || e.telephone>99999999)
throw msg8;
        break;
    }
    catch(string msg)
    {
        cout<<msg<<endl;
    }
    catch(int i)
    {
        cout <<msg7<<endl;
        cin.clear();
        cin.ignore(numeric_limits<streamsize>::max(), '\n');
        continue;
    }
}

while(true)
{
    try
    {
        cout<<"saisir la salaire          ";
        i>>e.salaire;
        if(cin.fail()) throw 1;

```

```

        else if(e.salaire<=0)    throw msg9;
        break;
    }
    catch(string msg)
    {
        cout<<msg<<endl;
    }
    catch(int i)
    {
        cout <<msg10<<endl;
        cin.clear();
        cin.ignore(numeric_limits<streamsize>::max(), '\n');
        continue;
    }
}

while(true)
{
    try
    {
        cout<<"saisir la specialité      ";
        i>>e.specialite;
        if(e.specialite!="mec" && e.specialite!="elec" &&
e.specialite!="tol") throw msg11;
        break;
    }
    catch(string msg)
    {
        cout<<msg<<endl;
    }
}

cout<<"saisir l'address      ";
i>>e.address;
return i;
}

ostream& operator<<(ostream& o,employee* e)
{
    o<<setw(5)<<e->id<<" ";
    o<<setw(5)<<e->age<<" ";

```

```

        o<<setw(5)<<e->nom<<" ";
        o<<setw(5)<<e->prenom<<" ";
        o<<setw(5)<<e->telephone<<" ";
        o<<setw(5)<<e->salaire<<" ";
        o<<setw(5)<<e->specialite<<" ";
        o<<setw(5)<<e->address<<endl;;
        return o;
    }

```

```

istream& operator>>(istream& i,employee* e)

```

```

{
    i>>e->id;
    i>>e->age;
    i>>e->nom;
    i>>e->prenom;
    i>>e->telephone;

```

```

    i>>e->salaire;
    i>>e->specialite;
    i>>e->address;
    return i;
}

```

```

// ----- class facture -----//

```

```

facture::facture(int num,date d,float montant,string etat)

```

```

{
    this->num_facture=num;
    this->date_payement=d;
    this->montant=montant;
    this->etat=etat;
}

```

```

ostream& operator<<(ostream& o,facture& f)

```

```

{
    o<<"le numero du facture est : "<<f.num_facture<<endl;
    o<<"le montant est : "<<f.montant<<endl;
    if(f.etat=="payee")
    {
        o<<"la date du paiement est : "<<f.date_payement<<endl;
        o<<"la facture est payée"<<endl;
    }
    else if(f.etat=="non_payee")

```

```

        o<<"la facture n'est payée"<<endl;

    return o;
}

istream& operator>>(istream& i,facture& f)
{
    string reponse;

    string msg1="ERREUR - le numero de facture existe deja";
    string msg2="ERREUR - numero de facture non valide";
    string msg3="ERREUR - le numero de facture doit etre positive";
    string msg5="ERREUR - montant de facture non valide";
    string msg4="ERREUR - le montant de facture doit etre positive";
    string msg6="ERREUR - reponse non valide";
    while(true)
    {
        try
        {
            cout<<"saisir le numero du facture    ";
            i>>f.num_facture;
            if(cin.fail()) throw 1;
//            if(facture_existe(f.num_facture)) throw msg1;
            if(f.num_facture<0)    throw msg3;
            break;
        }
        catch(string msg)
        {
            cout<<msg<<endl;
        }
        catch(int i)
        {
            cout <<msg2<<endl;
            cin.clear();
            cin.ignore(numeric_limits<streamsize>::max(), '\n');
            continue;
        }
    }

    while(true)
    {

```

```

try
{
    cout<<"saisir le montant du facture   ";
    i>>f.montant;
    if(cin.fail()) throw 1;
    if(f.montant<0)   throw msg4;
    break;
}
catch(string msg)
{
    cout<<msg<<endl;
}
catch(int i)
{
    cout <<msg5<<endl;
    cin.clear();
    cin.ignore(numeric_limits<streamsize>::max(), '\n');
    continue;
}
}

while(true)
{
    try
    {
        cout<<"la facture est payée ? Y/N       ";
        i>>reponse;
        if(reponse != "Y" && reponse != "N")   throw msg6;
        break;
    }
    catch(string msg)
    {
        cout<<msg<<endl;
    }
}

if(reponse=="Y")
{
    cout<<"saisir la date :"<<endl;;
    i>>f.date_payement;
    f.etat="payee";
}

```

```

    }
    else if(reponse=="N")
    {
        date d(0,0,0);
        f.etat="non_payee";
        f.date_payement =d;
    }

    return i;
}

ostream& operator<<(ostream& o,facture* f)
{
    o<<f->num_facture<<" ";
    o<<setw(10)<<f->montant<<" ";
    o<<setw(10)<<f->etat<<" ";
    o<<setw(10)<<&f->date_payement<<endl;;
    return o;
}

istream& operator>>(istream& i,facture* f)
{
    i>>f->num_facture;
    i>>f->montant;
    i>>f->etat;
    i>>&f->date_payement;
    return i;
}

// ----- class garage -----//
garage::~garage()
{
    for(int i=0;i<personnes.size();i++)
        delete personnes[i];
    this->personnes.clear();

    for(int i=0;i<factures.size();i++)
        delete factures[i];
    this->factures.clear();
}

```

```

    for (auto itr = pieces.begin(); itr != pieces.end(); itr++)
        delete *itr;
    this->pieces.clear();

    for(int i=0;i<reparations.size();i++)
        delete reparations[i];
    this->reparations.clear();
}

garage::garage(const garage& g)
{
    int i;
    personne *p;
    for(i=0;i<g.personnes.size();i++)
    {
        if(typeid(*g.personnes[i])==typeid(client))
            p = new client(static_cast<const client&>(*g.personnes[i]));
        if(typeid(*g.personnes[i])==typeid(employee))
            p = new employee(static_cast<const
employee&>(*g.personnes[i]));
        personnes.push_back(p);
    }

    facture *f;
    for(i=0;i<g.factures.size();i++)
    {
        f = new facture(*g.factures[i]);
        factures.push_back(f);
    }

    for (auto itr = pieces.begin(); itr != pieces.end(); itr++)
        this->pieces.push_back(*itr);

    reparation *r;
    for(i=0;i<g.reparations.size();i++)
    {
        r = new reparation(*g.reparations[i]);
        reparations.push_back(r);
    }
}

```



```
}
```

```
void garage::creer_fichier_personne(fstream & f)
{
    f.open("d:\\fichierPersonne.text", ios::in | ios::out | ios::trunc);
    if(! f.is_open()) exit;
}
```

```
void garage::creer_fichier_factures(fstream & f)
{
    f.open("d:\\fichierFacture.text", ios::in | ios::out | ios::trunc);
    if(! f.is_open()) exit;
}
```

```
void garage::creer_fichier_piece(fstream & f)
{
    f.open("d:\\fichierPiece.text", ios::in | ios::out | ios::trunc);
    if(! f.is_open()) exit;
}
```

```
void garage::creer_fichier_reparations(fstream & f)
{
    f.open("d:\\fichierReparation.text", ios::in | ios::out | ios::trunc);
    if(! f.is_open()) exit;
}
```

```
void garage::creer_fichier_vc(fstream & f)
{
    f.open("d:\\fichierVC.text", ios::in | ios::out | ios::trunc);
    if(! f.is_open()) exit;
}
```

```
void garage::enregistrer()
{
    fstream f1,f2,f3,f4,f5;

    creer_fichier_personne(f1);

    for(int i=0; i<personnes.size();i++)
    {
```

```

        if(typeid(*personnes[i]) == typeid(client))
            f1<<"2 "<<&static_cast<client>(*personnes[i])<<endl;
        else if(typeid(*personnes[i]) == typeid(employee))
            f1<<"3 "<<&static_cast<employee>(*personnes[i])<<endl;
    }
    f1.close();

    creer_fichier_factures(f2);
    for(int i=0; i<factures.size();i++)
    {
        f2<<i<<" ";
        f2<<&*factures[i]<<endl;
    }
    f2.close();

    creer_fichier_piece(f3);
    for(auto pi :pieces)
        f3<<"1 "<<&*pi<<endl;
    f3.close();

    creer_fichier_reparations(f4);
    for(int i=0; i<reparations.size();i++)
    {
        reparations[i]->enregistrer(i);
        pair<int,string> c(reparations[i]-
>Get_num(),"d:\\fichierReparation" + to_string(reparations[i]-
>Get_num()) + ".text");
        fichier_reparation.insert(c);
    }
    map<int,string>::iterator it;
    for(it=fichier_reparation.begin() ; it!=fichier_reparation.end() ; ++it)
    {
        f4<<"0 "<<it->first<<" "<<it->second<<endl;
    }
    f4.close();

    creer_fichier_vc(f5);
    f5<<&voiture_client;
    f5.close();
}

```

```

void garage::recuperer()
{
    fstream f1,f2,f3,f4,f5;
    int val;
    int k = 0;

    f1.open("d:\\fichierPersonne.text", ios::in);
    if(!f1.is_open()) exit(-1);
    while(1)
    {
        f1>>val;
        if(f1.eof())
            break;
        if(val == 2)
        {
            client* c =new client;
            f1>>&*c;
            personnes.push_back(c);
        }
        else if(val == 3)
        {
            employee* e =new employee;
            f1>>&*e;
            personnes.push_back(e);
        }
    }
    f1.close();

    f2.open("d:\\fichierFacture.text", ios::in);
    if(!f2.is_open()) exit(-1);
    while(1)
    {
        f2>>val;
        if(f2.fail())
            break;
        facture* f = new facture;
        f2>>&*f;
        factures.push_back(f);
    }
    f2.close();
}

```

```

f3.open("d:\\fichierPiece.text", ios::in);
if(!f3.is_open()) exit(-1);
while(1)
{
    f3>>val;
    if(f3.fail())
        break;
    piece* p = new piece;
    f3>>&*p;
    pieces.push_back(p);
}
f3.close();

```

```

pair<int,string> p;
f4.open("d:\\fichierReparation.text", ios::in);
while(1)
{
    f4>>val;
    if(f4.fail())
        break;
    f4>>p.first;
    f4>>p.second;
    fichier_reparation.insert(p);
}
f4.close();

```

```

reparation r;
map<int,string>::iterator it;
for(it=fichier_reparation.begin() ; it!=fichier_reparation.end() ; ++it)
{
    r.recuperer(it->first);
    reparation *x =new reparation(r);
    reparations.push_back(x);
}

```

```

int v,c;
f5.open("d:\\fichierVC.text", ios::in);
while(1)
{
    f5>>val;

```

```

        if(f5.fail())
            break;
        f5>>v;
        f5>>c;
        voiture_client.ajoutermap(v,c);
    }
    f5.close();

}

void garage::ajouter_personne(personne& p)
{
    personne *x;
    if(typeid(p)==typeid(client))
        x=new client(static_cast<const client&>(p));
    else if(typeid(p)==typeid(employee))
        x=new employee(static_cast<const employee&>(p));
    personnes.push_back(x);
}

void garage::supprimer_personne(int num)
{
    for(int i=0;i<personnes.size();i++)
        if(personnes[i]->Get_id()==num)
            personnes.erase(personnes.begin()+i);
}

void garage::supprimer_reparation(int num)
{
    for(int i=0;i<reparations.size();i++)
        if(reparations[i]->Get_num()==num)
            reparations.erase(reparations.begin()+i);
}

void garage::supprimer_facture(int num)
{
    for(int i=0;i<factures.size();i++)
        if(factures[i]->Get_num()==num)
            factures.erase(factures.begin()+i);
}

```

```
}
```

```
void garage::afficher_clients(int num)
{
    if(num == 0)
    {
        for(int i=0;i<personnes.size();i++)
        {
            if(typeid(*personnes[i])== typeid(client))
            {
                cout<<"----- Client -----"<<endl;
                cout<<static_cast<client>(*personnes[i])<<endl;
                cout<<endl;
            }
        }
    }
    else
    {
        for(int i=0;i<personnes.size();i++)
        {
            if(typeid(*personnes[i])== typeid(client) && personnes[i]-
>Get_id() == num)
            {
                cout<<"----- Client -----"<<endl;
                cout<<static_cast<client>(*personnes[i])<<endl;
                cout<<endl;
            }
        }
    }
}
```

```
void garage::afficher_clients2()
{
    for(int i=0;i<personnes.size();i++)
    {
        if(typeid(*personnes[i])== typeid(client))
            personnes[i]->afficher2();
    }
}
```

```
}
```

```
void garage::afficher_employees(int num)
```

```
{
    if(num == 0)
    {
        for(int i=0;i<personnes.size();i++)
        {
            if(typeid(*personnes[i])== typeid(employee))
            {
                cout<<"----- Employee -----"<<endl;
//      personnes[i]->afficher();
                cout<<static_cast<employee>>(*personnes[i])<<endl;
                cout<<endl;
            }
        }
    }
    else
    {
        for(int i=0;i<personnes.size();i++)
        {
            if(typeid(*personnes[i])== typeid(employee) && personnes[i]-
>Get_id() == num)
            {
                cout<<"----- Employee -----"<<endl;
                cout<<static_cast<employee>(*personnes[i])<<endl;
                cout<<endl;
            }
        }
    }
}
```

```
void garage::afficher_employees2()
```

```
{
    for(int i=0;i<personnes.size();i++)
    {
        if(typeid(*personnes[i])== typeid(employee))
            personnes[i]->afficher2();
    }
}
```

```

void garage::ajouter_facture(facture &f)
{
    facture *x = new facture(f);
    factures.push_back(x);
}

```

```

void garage::afficher_facture(int num)
{
    if(num == 0)
    {
        for(int i=0;i<factures.size();i++)
        {
            cout<<"----- Facture -----"<<endl;
            cout<<*factures[i]<<endl;;
            cout<<endl;
        }
    }
    else
    {
        for(int i=0;i<factures.size();i++)
        {
            if(factures[i]->Get_num() == num)
            {
                cout<<"----- Facture -----"<<endl;
                cout<<*factures[i]<<endl;;
                cout<<endl;
            }
        }
    }
}

```

```

void garage::afficher_factures2()
{
    for(int i=0;i<factures.size();i++)
        factures[i]->afficher2();
}

```

```

void garage::afficher_factures2_non_payee()
{
    for(int i=0;i<factures.size();i++)
    {

```



```

        if(factures[i]->Get_etat() == "non_payee")
            factures[i]->afficher2();
    }
}

```

```

void garage::ajouter_piece(piece &p)
{
    piece *x = new piece(p);
    pieces.push_back(x);
}

```

```

void garage::afficher_piece(int num)
{
    if(num == 0)
    {
        for(auto p :pieces)
        {
            cout<<"----- Piece -----"<<endl;
            cout<<*p<<endl;;
            cout<<endl;
        }
    }
    else
    {
        for(auto p :pieces)
        {
            if(p->Get_num() == num)
            {
                cout<<"----- Piece -----"<<endl;
                cout<<*p<<endl;;
                cout<<endl;
            }
        }
    }
}

```

```

void garage::afficher_pieces2()
{
    for(auto p :pieces)
        p->afficher2();
}

```

```
}
```

```
void garage::ajouter_reparation(reparation &r)
```

```
{  
    reparation *x = new reparation(r);  
    reparations.push_back(x);  
}
```

```
void garage::afficher_reparation(int num)
```

```
{  
    if(num == 0)  
    {  
        for(int i=0;i<reparations.size();i++)  
        {  
            cout<<"----- Reparation -----"<<endl;  
            cout<<*reparations[i]<<endl;;  
            cout<<endl;  
        }  
    }  
    else  
    {  
        for(int i=0;i<reparations.size();i++)  
        {  
            if(reparations[i]->Get_num() == num)  
            {  
                cout<<"----- Reparation -----"<<endl;  
                cout<<*reparations[i]<<endl;;  
                cout<<endl;  
            }  
        }  
    }  
}
```

```
void garage::supprimer_piece(int n)
```

```
{  
    for (auto it = pieces.begin(); it != pieces.end(); ++it)  
    {  
        if ((*it)->Get_num() == n)  
        {  
            pieces.erase(it);  
            break;  
        }  
    }  
}
```

```

    }
}
}

```

```

void garage::afficher_reparation2()
{
    for(int i=0;i<reparations.size();i++)
        reparations[i]->afficher2();
}

```

```

void garage::afficher_reparation2_active()
{
    for(int i=0;i<reparations.size();i++)
    {
        if(reparations[i]->Get_etat()=="active")
            reparations[i]->afficher2();
    }
}

```

```

void garage::afficher_file()
{
    voiture *x ;
    Queue<voiture*> q;
    q.copy(file_attente,q);
    while(!q.empty())
    {
        cout<<" ----- affichage file -----"<<endl;
        q.front()->afficher();
        file_attente.push(q.front());
        q.pop();
    }
}

```

```

void garage::afficher_file2()
{
    voiture *x ;
    Queue<voiture*> q;
    q.copy(file_attente,q);
    while(!q.empty())
    {

```

```

        q.front()->afficher2();
        file_attente.push(q.front());
        q.pop();
    }
}
void garage::ajouter_voiture()
{
    string reponse;
    VElectrique ve;
    VMecanique vm;
    while(1)
    {
        try
        {
            cout<<"donner le type de voiture E:elecrique /
M:mecanique"<<endl;
            cin>>reponse;
            if(reponse != "E" && reponse != "M") throw 1;
            break;
        }
        catch(int i)
        {
            cout<<"ERREUR - reponse non validee, les type sont
E:elecrique / M:mecanique"<<endl;
        }
    }
    if(reponse == "E")
    {
        cin>>ve;
        VElectrique *x = new VElectrique(ve);
        file_attente.push(x);
    }
    if(reponse == "M")
    {
        cin>>vm;
        VMecanique *x = new VMecanique(vm);
        file_attente.push(x);
    }
    afficher_clients2();
    int id_client;
    cout<<"donner l'id du client proprietaire de cette voiture"<<endl;

```

```

        cin>>id_client;
        voiture_client.ajoutermap(file_attente.front()-
>getvoiture(),id_client);
};

```

```

void garage::afficher_voitures()
{
    for(int i=0;i<reparations.size();i++)
    {
        reparations[i]->afficher_v2();
    }
};

```

//----- voiture -----

```

void voiture::afficher()
{
    cout<<"\nid de voiture est "<<id;
    cout<<"\nsa matricule : "<<immatricule;
    cout<<"\nnombre de place : "<<nb_place;
    cout<<"\nsa puissance "<<puissance;
    cout<<"\nkilometre : "<<kilometrage;
    cout<<"\nmarque:"<<marque;
    cout<<"\ndate_mise_en_service : "<<d;
    cout<<"\ncouleur est "<<couleur;

}

ostream& operator<<(ostream&o,voiture*v)
{
    o<<v->id<<" ";
    o<<setw(10)<<v->immatricule<<" ";
    o<<setw(10)<<v->nb_place<<" ";
    o<<setw(10)<<v->puissance<<" ";
    o<<setw(10)<<v->kilometrage<<" ";
    o<<setw(10)<<v->marque<<" ";
    o<<setw(10)<<&v->d<<" ";
    o<<setw(10)<<v->couleur<<" ";
    return o;
}

ostream& operator<<(ostream&o,voiture&v)

```

```

{
    cout<<"\nid de voiture est ";
    o<<v.id;
    cout<<"\nsa matricule :";
    o<<v.immatricule;
    cout<<"\nnombre de place :";
    o<<v.nb_place;
    cout<<"\nsa puissance ";
    o<<v.puissance;
    cout<<"\nkilometre :";
    o<<v.kilometrage;
    cout<<"\nmarque:";
    o<<v.marque;
    cout<<"\ndate_mise_en_service :";
    o<<v.d;
    cout<<"\ncouleur est ";
    o<<v.couleur;
    return o;
}

voiture::voiture(int id,string im,int nb,int p,int k ,string m,date d,string
c)
{
    this->id=id;
    immatricule =im;
    nb_place=nb;
    puissance=p;
    kilometrage=k;
    marque=m;
    this->d=d;
    couleur=c;
}

voiture::voiture(const voiture&v)
{
    this->id=v.id;
    this->immatricule =v.immatricule;
    this->nb_place=v.nb_place;
    this->puissance=v.puissance;
    this->kilometrage=v.kilometrage;
    this->marque=v.marque;
    this->d=v.d;

```

```

        this->couleur=v.couleur;
    }

istream&operator>>(istream&i,voiture&v)
{
    cout<<"\nid de voiture est ";
    i>>v.id;
    cout<<"\nsa matricule :";
    i>>v.immatricule;
    cout<<"\nnombre de place :";
    i>>v.nb_place;
    cout<<"\nsa puissance ";
    i>>v.puissance;
    cout<<"\nkilometre :";
    i>>v.kilometrage;
    cout<<"\nmarque:";
    i>>v.marque;
    cout<<"\ndate_mise_en_service :";
    i>>v.d;
    cout<<"\ncouleur est ";
    i>>v.couleur;
    return i;
}

```

```

istream&operator>>(istream&i,voiture*v)
{
    i>>v->id;
    i>>v->immatricule;
    i>>v->nb_place;
    i>>v->puissance;
    i>>v->kilometrage;
    i>>v->marque;
    i>>&v->d;
    i>>v->couleur;
    return i;
};

```

// ----- class VMecanique ----- //

```

VMecanique::VMecanique(int id,string im,int nb,int p,int k ,string
m,date d,string c,string type_carburant):voiture( id, im, nb, p, k , m, d,
c)

```

```
{
    this->type_carburant=type_carburant;
}
```

```
istream&operator>>(istream&i,VMecanique&v)
```

```
{
    string msg;
    string msg1="ERREUR - la voiture existe deja";
    string msg2="ERREUR - l'id doit etre positive";
    string msg3="ERREUR - id non valide";
    string msg4="ERREUR - le nombre de place doit etre positive";
    string msg5="ERREUR - nombre de place non valide";
    string msg6="ERREUR - la puissance doit etre positive";
    string msg7="ERREUR - puissance non valide";
    string msg8="ERREUR - le kilometrage doit etre positive";
    string msg9="ERREUR - kilometrage non valide";
```

```
while(true)
```

```
{
    try
    {
        cout<<"\nid de Voiture Mecanique est ";
        i>>v.id;
        if(cin.fail()) throw 1;
        else if(v.id<0) throw msg2;
        break;
    }
    catch(string msg)
    {
        cout<<msg<<endl;
    }
    catch(int i)
    {
        cout <<msg3<<endl;
        cin.clear();
        cin.ignore(numeric_limits<streamsize>::max(), '\n');
        continue;
    }
}
```

```
cout<<"\nsa matricule :";
```



```
i>>v.immatricule;
```

```
while(true)
{
    try
    {
        cout<<"\nnombre de place :";
        i>>v.nb_place;
        if(cin.fail()) throw 1;
        else if(v.nb_place<0)    throw msg4;
        break;
    }
    catch(string msg)
    {
        cout<<msg<<endl;
    }
    catch(int i)
    {
        cout <<msg5<<endl;
        cin.clear();
        cin.ignore(numeric_limits<streamsize>::max(), '\n');
        continue;
    }
}
```

```
while(true)
{
    try
    {
        cout<<"\nsa puissance ";
        i>>v.puissance;
        if(cin.fail()) throw 1;
        else if(v.puissance<0)    throw msg6;
        break;
    }
    catch(string msg)
    {
        cout<<msg<<endl;
    }
    catch(int i)
    {

```

```

        cout <<msg7<<endl;
        cin.clear();
        cin.ignore(numeric_limits<streamsize>::max(), '\n');
        continue;
    }
}

while(true)
{
    try
    {
        cout<<"\nkilometre :";
        i>>v.kilometrage;
        if(cin.fail()) throw 1;
        else if(v.kilometrage<0)    throw msg8;
        break;
    }
    catch(string msg)
    {
        cout<<msg<<endl;
    }
    catch(int i)
    {
        cout <<msg9<<endl;
        cin.clear();
        cin.ignore(numeric_limits<streamsize>::max(), '\n');
        continue;
    }
}

cout<<"\nmarque:";
i>>v.marque;
cout<<"\ndate_mise_en_service :";
i>>v.d;
cout<<"\ncouleur est ";
i>>v.couleur;
cout<<"le type de carburant";
i>>v.type_carburant;
return i;
}

istream&operator>>(istream&i,VMecanique*v)

```

```

{
    i>>v->id;
    i>>v->immatricule;
    i>>v->nb_place;
    i>>v->puissance;
    i>>v->kilometrage;
    i>>v->marque;
    i>>&v->d;
    i>>v->couleur;
    i>>v->type_carburant;
    return i;
};

void VMecanique::afficher()
{
    voiture::afficher();
    cout<<"\ntype de carburant"<<type_carburant<<endl;

}

ostream& operator<<(ostream& o,VMecanique& v)
{
    cout<<"\nid de voiture mecnique est ";
    o<<v.id;
    cout<<"\nsa matricule :";
    o<<v.immatricule;
    cout<<"\nnombre de place :";
    o<<v.nb_place;
    cout<<"\nsa puissance ";
    o<<v.puissance;
    cout<<"\nkilometre :";
    o<<v.kilometrage;
    cout<<"\nmarque:";
    o<<v.marque;
    cout<<"\ndate_mise_en_service :";
    o<<v.d;
    cout<<"\ncouleur est ";
    o<<v.couleur;
    cout<<"\ntype de carburant";
    o<<v.type_carburant;
    return o;
}

```

```
ostream& operator<<(ostream&o,VMecanique*v)
{
    o<<v->id<<" ";
    o<<setw(10)<<v->immatricule<<" ";
    o<<setw(10)<<v->nb_place<<" ";
    o<<setw(10)<<v->puissance<<" ";
    o<<setw(10)<<v->kilometrage<<" ";
    o<<setw(10)<<v->marque<<" ";
    o<<setw(10)<<&v->d<<" ";
    o<<setw(10)<<v->couleur<<" ";
    o<<setw(10)<<v->type_carburant<<" ";
    return o;
}
```

// ----- class VElectrique ----- //

```
istream&operator>>(istream&i,VElectrique&v)
{
    string msg;
    string msg1="ERREUR - la voiture existe deja";
    string msg2="ERREUR - l'id doit etre positive";
    string msg3="ERREUR - id non valide";
    string msg4="ERREUR - le nombre de place doit etre positive";
    string msg5="ERREUR - nombre de place non valide";
    string msg6="ERREUR - la puissance doit etre positive";
    string msg7="ERREUR - puissance non valide";
    string msg8="ERREUR - le kilometrage doit etre positive";
    string msg9="ERREUR - kilometrage non valide";
    string msg10="ERREUR - le temps de recharge doit etre positive";
    string msg11="ERREUR - temps de recharge non valide";

    while(true)
    {
        try
        {
            cout<<"\nid de Voiture Electrique est ";
            i>>v.id;
            if(cin.fail()) throw 1;
            else if(v.id<0) throw msg2;
            break;
        }
    }
}
```

```

    }
    catch(string msg)
    {
        cout<<msg<<endl;
    }
    catch(int i)
    {
        cout <<msg3<<endl;
        cin.clear();
        cin.ignore(numeric_limits<streamsize>::max(), '\n');
        continue;
    }
}

```

```

cout<<"\nsa matricule :";
i>>v.immatricule;

```

```

while(true)
{
    try
    {
        cout<<"\nnombre de place :";
        i>>v.nb_place;
        if(cin.fail()) throw 1;
        else if(v.nb_place<0) throw msg4;
        break;
    }
    catch(string msg)
    {
        cout<<msg<<endl;
    }
    catch(int i)
    {
        cout <<msg5<<endl;
        cin.clear();
        cin.ignore(numeric_limits<streamsize>::max(), '\n');
        continue;
    }
}

```

```

while(true)

```

```

{
    try
    {
        cout<<"\nsa puissance ";
        i>>v.puissance;
        if(cin.fail()) throw 1;
        else if(v.puissance<0)    throw msg6;
        break;
    }
    catch(string msg)
    {
        cout<<msg<<endl;
    }
    catch(int i)
    {
        cout <<msg7<<endl;
        cin.clear();
        cin.ignore(numeric_limits<streamsize>::max(), '\n');
        continue;
    }
}

```

```

while(true)
{
    try
    {
        cout<<"\nkilometre :";
        i>>v.kilometrage;
        if(cin.fail()) throw 1;
        else if(v.kilometrage<0)    throw msg8;
        break;
    }
    catch(string msg)
    {
        cout<<msg<<endl;
    }
    catch(int i)
    {
        cout <<msg9<<endl;
        cin.clear();
        cin.ignore(numeric_limits<streamsize>::max(), '\n');
    }
}

```

```

        continue;
    }
}

cout<<"\nmarque:";
i>>v.marque;
cout<<"\ndate_mise_en_service :";
i>>v.d;
cout<<"\ncouleur est ";
i>>v.couleur;
cout<<"\nle type de batterie";
i>>v.batterie;
while(true)
{
    try
    {
        cout<<"\ntemp de charge";
        i>>v.temps_charge;
        if(cin.fail()) throw 1;
        else if(v.temps_charge<0)    throw msg10;
        break;
    }
    catch(string msg)
    {
        cout<<msg<<endl;
    }
    catch(int i)
    {
        cout <<msg11<<endl;
        cin.clear();
        cin.ignore(numeric_limits<streamsize>::max(), '\n');
        continue;
    }
}

return i;
}

istream& operator>>(istream&i,VElectrique*v)
{
    i>>v->id;

```

```

        i>>v->immatricule;
        i>>v->nb_place;
        i>>v->puissance;
        i>>v->kilometrage;
        i>>v->marque;
        i>>&v->d;
        i>>v->couleur;
        i>>v->batterie;
        i>>v->temps_charge;
        return i;
};

```

```

void VElectrique::afficher()
{
    voiture::afficher();
    cout<<"\nle type de batterie"<<batterie;
    cout<<"\ntemp de charge"<<temps_charge<<endl;
}

ostream& operator<<(ostream&o,VElectrique&v)
{
    cout<<"\nid de voiture electrique est ";
    o<<v.id;
    cout<<"\nsa matricule :";
    o<<v.immatricule;
    cout<<"\nnombre de place :";
    o<<v.nb_place;
    cout<<"\nsa puissance ";
    o<<v.puissance;
    cout<<"\nkilometre :";
    o<<v.kilometrage;
    cout<<"\nmarque:";
    o<<v.marque;
    cout<<"\ndate_mise_en_service :";
    o<<v.d;
    cout<<"\ncouleur est ";
    o<<v.couleur;
    cout<<"\nla type de batterie";
    o<<v.batterie;
    cout<<"\ntemp de charge";
    o<<v.temps_charge;
}

```



```

    return o;
}
ostream& operator<<(ostream&o,VElectrique*v)
{
    o<<v->id<<" ";
    o<<setw(10)<<v->immatricule<<" ";
    o<<setw(10)<<v->nb_place<<" ";
    o<<setw(10)<<v->puissance<<" ";
    o<<setw(10)<<v->kilometrage<<" ";
    o<<setw(10)<<v->marque<<" ";
    o<<setw(10)<<&v->d<<" ";
    o<<setw(10)<<v->couleur<<" ";
    o<<setw(10)<<v->batterie<<" ";
    o<<setw(10)<<v->temps_charge<<" ";
    return o;
}

```

```

void voiture::ecrire(fstream& f)
{
    f<<id<<" ";
    f<<setw(10)<<immatricule<<" ";
    f<<setw(10)<<nb_place<<" ";
    f<<setw(10)<<puissance<<" ";
    f<<setw(10)<<kilometrage<<" ";
    f<<setw(10)<<marque<<" ";
    f<<setw(10)<<&d<<" ";
    f<<setw(10)<<couleur<<" "<<endl;
}

```

```

void VElectrique::ecrire(fstream& f)
{
    f<<id<<" ";
    f<<setw(10)<<immatricule<<" ";
    f<<setw(10)<<nb_place<<" ";
    f<<setw(10)<<puissance<<" ";
    f<<setw(10)<<kilometrage<<" ";
    f<<setw(10)<<marque<<" ";
    f<<setw(10)<<&d<<" ";
    f<<setw(10)<<couleur<<" "<<endl;
    f<<setw(10)<<batterie<<" ";
    f<<setw(10)<<temps_charge<<" "<<endl;
}

```

```

}
VElectrique::VElectrique(int id,string im,int nb,int p,int k ,string m,date
d,string c,string batterie,int temps):voiture( id, im, nb, p, k , m, d, c)
{
    this->batterie=batterie;
    temps_charge=temps;
}

// ----- piece -----//
piece::piece(int reference_piece,string designation,int
quantiter_stock,float prix,int critique,int quantiter)
{
    this->reference_piece=reference_piece;
    this->designation=designation;
    this->quantiter_stock=quantiter_stock;
    this->prix=prix;
    this->critique=critique;
    this->quantiter=quantiter;
}

void piece::afficher()
{
    cout<<"\nreference de piece : "<<reference_piece;
    cout<<"\nsa designation  "<<designation;
    cout<<"\nquantiter en stock  "<<quantiter_stock;
    cout<<"\nsa prix  "<<prix;
    cout<<"\nsa critique stock  "<<critique;
    cout<<"\nquantiter  "<<quantiter;

}

istream& operator>>(istream&i,piece*p)
{
    i>>p->reference_piece;
    i>>p->designation;
    i>>p->quantiter_stock;
    i>>p->prix;
    i>>p->critique;
    i>>p->quantiter;
    return i;
}

```

```

istream& operator>>(istream&i,piece&p)
{
    string msg;
    string msg1="ERREUR - la piece existe deja";
    string msg2="ERREUR - la reference doit etre positive";
    string msg3="ERREUR - reference non valide";
    string msg4="ERREUR - la quantite en stock doit etre positive";
    string msg5="ERREUR - quantite en stock non valide";
    string msg6="ERREUR - le prix doit etre positive";
    string msg7="ERREUR - prix non valide";
    string msg8="ERREUR - le seuil critique doit etre positive";
    string msg9="ERREUR - seuil critique non valide";

    while(true)
    {
        try
        {
            cout<<"\nreference de piece est :";
            i>>p.reference_piece;
            if(cin.fail()) throw 1;
            else if(p.reference_piece<0) throw msg2;
            break;
        }
        catch(string msg)
        {
            cout<<msg<<endl;
        }
        catch(int i)
        {
            cout <<msg3<<endl;
            cin.clear();
            cin.ignore(numeric_limits<streamsize>::max(), '\n');
            continue;
        }
    }
    cout<<"\ndesignation :";i>>p.designation;
    while(true)
    {
        try

```

```

{
    cout<<"\nquantiter dans les stock :";i>>p.quantiter_stock;
    if(cin.fail()) throw 1;
    else if(p.quantiter_stock<0)  throw msg4;
    break;
}
catch(string msg)
{
    cout<<msg<<endl;
}
catch(int i)
{
    cout <<msg5<<endl;
    cin.clear();
    cin.ignore(numeric_limits<streamsize>::max(), '\n');
    continue;
}
}

```

```

while(true)
{
    try
    {
        cout<<"\nprix de la piece ";i>>p.prix;
        if(cin.fail()) throw 1;
        else if(p.prix<0)  throw msg6;
        break;
    }
    catch(string msg)
    {
        cout<<msg<<endl;
    }
    catch(int i)
    {
        cout <<msg7<<endl;
        cin.clear();
        cin.ignore(numeric_limits<streamsize>::max(), '\n');
        continue;
    }
}
}

```

```

while(true)
{
    try
    {
        cout<<"\ncritique de stock ";i>>p.critique;
        if(cin.fail()) throw 1;
        else if(p.critique<0)  throw msg8;
        break;
    }
    catch(string msg)
    {
        cout<<msg<<endl;
    }
    catch(int i)
    {
        cout <<msg9<<endl;
        cin.clear();
        cin.ignore(numeric_limits<streamsize>::max(), '\n');
        continue;
    }
}
p.quantiter = 0;
return i;
}

ostream& operator<<(ostream&o,piece*p)
{
    o<<setw(5)<<p->reference_piece<<" ";
    o<<setw(5)<<p->designation<<" ";
    o<<setw(5)<<p->quantiter_stock<<" ";
    o<<setw(5)<<p->prix<<" ";
    o<<setw(5)<<p->critique<<" ";
    o<<setw(5)<<p->quantiter<<endl;
    return o;
}

ostream& operator<<(ostream&o,piece&p)
{
    o<<"\nreference de piece est : "<<p.reference_piece;
    o<<"\ndesignation : "<<p.designation;
    o<<"\nquatiter dans les stock : "<<p.quantiter_stock;
    o<<"\nprix de la piece "<<p.prix;
    o<<"\ncritique de stock"<<p.critique;

```

```

    o<<"\nquantiter qui sera utiliser "<<p.quantiter;
    return o;
}

// ----- reparation -----//
void reparation::ajouter_piece(piece& p)
{
    piece *q=new piece(p);
    auto foundlterator = find(piec.begin(), piec.end(),q);
    if (foundlterator != piec.end()) {
        cout << "piece existe deja dans la liste! "<< endl;
    } else {
        this->piec.push_back(q);
        cout << "piece ajoutee a la liste "<< endl;
    }
}

}

void reparation::ajouter_employer(employee&e)
{
    employee *em=new employee(e);

    auto foundlterator = find(employer.begin(), employer.end(), em);
    if (foundlterator != employer.end()) {
        cout << "employer existe deja dans la liste!"<< endl;
    } else {
        this->employer.push_back(em);
        cout << "employer ajoutee a la liste"<< endl;
    }
}

}

void reparation::Set_vehicule(voiture &ve)
{
    if(typeid(ve)==typeid(VElectrique))
        v=new VElectrique(static_cast<const VElectrique&>(ve));
    else if(typeid(ve)==typeid(VMecanique))
        v=new VMecanique(static_cast<const VMecanique&>(ve));
}

}

reparation::reparation(int num, date d, string etat)

```

```

{
    this->num_reparation = num;
    this->d = d;
    this->etat = etat;
    this->num_facture = 0;
}
reparation::~~reparation()
{
    for (auto itr = employer.begin(); itr != employer.end(); itr++)
        delete *itr;
    employer.clear();

    for (int i = 0; i < piec.size(); i++) {
        delete piec[i];
    }
    piec.clear();
    delete v;
}
float reparation::calcul_montant() {
    float total = 0;
    for (int i = 0; i < piec.size(); i++) {
        total += (float)piec[i]->get_quantiter()*piec[i]->get_prix();
    }

    for (auto const& emp : employer)
        total += emp->Get_salaire();
    return total;
}

void reparation::afficher()
{
    cout << "Numéro de réparation:   " << num_reparation << endl;
    cout << "Date de réparation:      " << d << endl;
    cout << "Etat:                          " << etat << endl;
    if(num_facture==0)
        cout<<"aucune facture"<<endl;
    else
        cout<<"num facture :           " << num_facture<<endl;
    cout << "La voiture est :           " << endl;
    v->afficher();
}

```

```

        sort(employer.begin(),employer.end(),[](employee* c, employee* b)
{ return *c < *b; });
        cout << "Liste des employes: ";
        int i = 1;
        for (employee* p : employer) {
            cout << "employe  "<<i<<": " << endl<< *p << endl;i++;
        }
        sort(piec.begin(), piec.end(), [](piece* c, piece* b){return *c < *b;});
        cout << "Liste des piece: ";
        int z = 1;
        for (piece* p : piec) {
            cout << "Piece  "<<z<<": " << endl<< *p << endl;z++;
        }

        cout << "\nMontant total: " << calcul_montant() << endl;
    }

```

```

istream& operator>>(istream& i, reparation* r)
{
    i >> r->num_reparation;
    i >> &r->d;
    r->etat = "active";
    i>>r->num_facture;
    i >>r->v;
    return i;
}

```

```

istream& operator>>(istream&i,reparation &r)
{
    int reponse;
    VElectrique ve;
    VMecanique vm;
    voiture *x;

    cout<<"donner le numero du reparation est : ";
    i>>r.num_reparation;
    cout<<"donner la date de reparation :   ";
    i>>r.d;
    r.etat = "active";
    r.num_facture = 0;
}

```



```

        cout<<"saisir type de voiture , 1/voiture electrique, 2/voiture
mecanique : ";
        i>>reponse;
        if(reponse == 1)
        {
            i>>ve;
            x=new VElectrique(ve);
        }
        else if(reponse == 2)
        {
            i>>vm;
            x=new VMecanique(vm);
        }
        r.v = x;
        return i;
    }

```

```

reparation::reparation(const reparation& r)
{
    int i;
    num_reparation = r.num_reparation;
    d = r.d;
    etat = r.etat;
    num_facture = r.num_facture;

    if (dynamic_cast<VMecanique*>(r.v) != nullptr)
    {
        v = new VMecanique(*(static_cast<VMecanique*>(r.v)));
    }
    else if (dynamic_cast<VElectrique*>(r.v) != nullptr)
    {
        v = new VElectrique(*(static_cast<VElectrique*>(r.v)));
    }

    for (auto itr = r.employer.begin(); itr != r.employer.end(); itr++)
        this->employer.push_back(*itr);

    piece *p;
    for(i=0;i<r.piec.size();i++)
    {
        p = new piece(*r.piec[i]);
    }
}

```

```

        piec.push_back(p);
    }

    this->num_reparation = r.num_reparation;
    this->d = r.d;
    this->etat = r.etat;
}

ostream& operator<<(ostream&o,reparation &r)
{
    cout<<"\n*****affichage de reparation*****\n";
    o<<"\nle numero du reparation est : "<<r.num_reparation;
    o<<"\nla date de reparation :   "<<r.d<<endl;

    if(r.num_facture==0)
        o<<"aucune facture"<<endl;
    else
        o<<"num facture :       "<<r.num_facture<<endl;

    o<<"la voiture est : "<<endl;

    if(typeid(*r.v) == typeid(voiture))
        o<<*r.v;
    else if(typeid(*r.v) == typeid(VElectrique))
        o<<static_cast<VElectrique&>(*r.v); // -----
----- this actually work HHHHHHH -----
    else if(typeid(*r.v) == typeid(VMecanique))
        o<<static_cast<VMecanique&>(*r.v);

    sort(r.employer.begin(),r.employer.end(),[](employee* c, employee*
b) { return *c < *b; });
    o<<"\nles employees participant a la reparation : "<<endl;
    int i = 1;
    for (employee* p : r.employer)
        o << "employe "<<i<<": \n\n" << *p << endl;i++;

    sort(r.piec.begin(), r.piec.end(), [](piece* c, piece* b){return *c <
*b;});
    o<<"les pieces utilise dans la reparation est :";
    int z = 1;

```

```

        for (piece* p : r.piec) {
            o << "Piece  "<<z<<": " << endl<< *p << endl;z++;
        }

        if(r.etat == "active")
            o<<"la reparation est active"<<endl;
        else if(r.etat == "fini")
            o<<"la reparation est termine"<<endl;
        cout<<"\nla montant de reparation  "<<r.calcul_montant();

        return o;
    }

    void reparation::afficher_employees()
    {
        for(auto em :employer)
            cout<<*em<<endl;
    }

    void reparation::afficher_pieces()
    {
        for(int i=0;i<piec.size();i++)
            cout<<*piec[i]<<endl;
    }

    void reparation::afficher_voiture()
    {
        cout<<*v;
    }

    void reparation::afficher_v2()
    {
        v->afficher2();
    }

    void reparation::enregistrer(int i)
    {
        fstream f;
        string ch;
        creer_fichier_reparation(f);
    }

```

```

f<<i<<" ";
f<<num_reparation<<" ";
f<<setw(10)<<&d<<" ";
f<<setw(10)<<etat<<" "<<endl;
f<<setw(10)<<num_facture<<" "<<endl;

if(typeid(*v) == typeid(voiture))
    f<<"1 "<<&*v<<endl;
else if(typeid(*v) == typeid(VElectrique))
    f<<"2 "<<&static_cast<VElectrique>(*v)<<endl;
else if(typeid(*v) == typeid(VMecanique))
    f<<"3 "<<&static_cast<VMecanique>(*v)<<endl;

for(auto em :employer)
    f<<"1 "<<&*em<<endl;
for(int i=0;i<piec.size();i++)
    f<<"2 "<<&*piec[i]<<endl;
f.close();

};
void reparation::recuperer(int num)
{
    int val;
    fstream f;
    string ch;
    ch = "d:\\fichierReparation" + to_string(num) + ".text";
    f.open(ch, ios::in);
    if(!f.is_open()) exit(-1);
    f>>val;
    f>>num_reparation;
    f>>&d;
    f>>etat;
    f>>num_facture;

    f>>val;
    if(val == 2)
    {
        VElectrique* ve =new VElectrique;
        f>>&*ve;
        v = ve;
    }
}

```

```

else if(val == 3)
{

    VMecanique* vm =new VMecanique;
    f>>&*vm;
    v = vm;
}

while(1)
{
    f>>val;
    if(f.fail())
        break;
    if(val == 1)
    {
        employee* e =new employee;
        f>>&*e;
        employer.push_back(e);
    }
    else if(val == 2)
    {
        piece* p =new piece;
        f>>&*p;
        piec.push_back(p);
    }
}
f.close();
}

void reparation::modifier_reparation()
{
    int x;
    do{
        cout<<"\n\n\n\nsi tu peut modifier voiture donner tapez 1\nmodifier
piece tapez 2 \nmodifier employer tapez 3 :";
        int y,w,a1,i;
        cin>>y;
        if (y==1)
        {
            cout<<"\nquele type de voiture qui sera choisit 1 electrique 2
mecanique ";

```

```

int a2;
cin>>a2;

if(a2==1)
{
    VElectrique ve;
    cin>>ve;
    ve.Set_num(v->getvoiture());
    v=new VElectrique(ve);
}
if(a2==2)
{
    VMecanique ve;
    cin>>ve;
    ve.Set_num(v->getvoiture());
    v=new VMecanique(ve);
}
}
if(y==2)
{
    cout<<"\n saisir une piece    ";
    piece p2;
    cin>>p2;
    piece* p1= new piece(p2);
    replace_if(piec.begin(), piec.end(), [p1](piece* p){ return *p == *p1;
}, p1);
    for (piece* p : piec)
        cout << "Piece: " << *p << endl;
}
else if(y==3)
{
    cout<<"\n saisir un employee";
    employee p2;
    cin>>p2;
    employee* p1= new employee(p2);
    replace_if(employer.begin(), employer.end(), [p1](employee* p){
return *p == *p1; }, p1);
    for (employee* p : employer)
        cout << "employee: " << *p << endl;
}
cout<<"\n si tu peut ajouter autre modification tapez 1 si non 0 ";
cin>>x;

```

```

    }while (x);

}

void reparation::modifier_reparation_fichier(int aa)
{
    reparation r;
    fstream f;
    string ch;
    ch = "fichierReparation" + to_string(aa) + ".text";
    f.open(ch,ios::in);
    r.recuperer(aa);
    f.close();
    f.open(ch,ios::in|ios::trunc) ;
    r.modifier_reparation();
    r.enregistrer(aa);
    f.close();
}

void reparation::creer_fichier_reparation(fstream & f)
{
    string ch;
    ch = "d:\\fichierReparation" + to_string(num_reparation) + ".text";
    f.open(ch, ios::in | ios::out | ios::trunc);
    if(! f.is_open()) exit;
};

//----- class cv -----//
vc::vc(const vc& other)
{
    for (auto it = other.a.begin(); it != other.a.end(); ++it)
        a[it->first] = it->second;

    it = a.begin();
}

void vc::ajoutermap(int v,int c)
{
    a.insert(make_pair(v,c));
}

```

```

void vc::affichermmap()
{
    for (it=a.begin();it!=a.end();it++)
        cout<<"id voiture : "<<it->first<<"    "<<"id client : "<<it->second<<endl;
};

```

```

void vc::modifiermap(int c,int v)
{
    it = a.find(v);

    if (it != a.end()) {
        a[c] = it->second;
        a.erase(it);
    }
}

```

```

void vc::recherchermmap(int b)
{
    it=a.find(b);
    if (it!=a.end()) cout<<"cette voiture existe ";
    else cout<<"cette voiture n'existe pas ";
};

```

```

void vc::suprimmap(int b)
{
    it=a.find(b);
    if (it!=a.end()) a.erase(b);
};
//-----menu -----//

```

```

void afficher_menu_principale(garage& g)
{
    int choix;
    cout<<" ----- Menu -----"<<endl;
    cout<<"taper 1 pour gerer les client "<<endl;
    cout<<"taper 2 pour gerer les employee "<<endl;
    cout<<"taper 3 pour gerer les piece "<<endl;
    cout<<"taper 4 pour faire une reparation "<<endl;
}

```



```
cout<<"taper 5 pour gerer la liste d'attentes de voitures"<<endl;
cout<<"taper 6 pour gerer les factures "<<endl;
```

```
cin>>choix;
switch (choix)
{
    case 1: menu_client(g);
    case 2: menu_employee(g);
    case 3: menu_piece(g);
    case 4: menu_reparation(g);
    case 5: menu_voiture(g);
    case 6: menu_facture(g);
}
}
```

```
void ajouter_client(garage &g)
{
    client c;
    int r = 1;
    bool indicator = false;
    while (r != 0)
    {
        cout<<"----- saisir un client -----"<<endl;
        cin>>c;
        for(int i=0;i<g.personnes.size();i++)
        {
            if(g.personnes[i]->Get_id()==c.Get_id())
            {
                cout<<"!!!!!! le client existe deja !!!!!!"<<endl;
                indicator = true;
            }
        }
        if(indicator == false)
        {
            g.ajouter_personne(c);
            g.enregistrer();
            cout<<endl;
        }
    }
}
```

```

        cout<<"taper 0 pour terminer "<<endl;
        cin>>r;
    }
}

void supprimer_client(garage &g)
{
    int r = 1;
    int id;
    g.afficher_clients2();
    while (r != 0)
    {
        bool indicator = false;

        cout<<"-----+++-----"<<endl;
        cout<<"----- saisir l'id de client a supprimer -----"<<endl;
        cin>>id;
        for(int i=0;i<g.personnes.size();i++)
        {
            if(g.personnes[i]->Get_id()==id && typeid(*g.personnes[i])==
            typeid(client))
            {
                g.supprimer_personne(id);
                cout<<"supprimée"<<endl;
                g.enregistrer();
                indicator = true;
            }
        }
        if(indicator == false)
        {
            cout<<"!!! le client n'existe pas !!!"<<endl;
            cout<<endl;
        }
        cout<<"taper 0 pour terminer "<<endl;
        cin>>r;
    }
}

```

```

void modifier_client(garage &g)
{
    int r = 1;

```

```

int id;
client c;
bool indicator = false;
g.afficher_clients2();
cout<<"-----+++"<<endl;
while (r != 0)
{
    cout<<"----- saisir l'id de client a modifier -----"<<endl;
    cin>>id;
    for(int i=0;i<g.personnes.size();i++)
    {
        if(g.personnes[i]->Get_id()==id && typeid(*g.personnes[i])==
typeid(client))
        {
            g.supprimer_personne(id);
            cin>>c;
            c.Set_id(id);
            g.ajouter_personne(c);
            g.enregistrer();
            indicator = true;
        }
    }
    if(indicator == false)
    {
        cout<<"!!! le client n'existe pas !!!"<<endl;
        cout<<endl;
    }
    cout<<"taper 0 pour terminer "<<endl;
    cin>>r;
}
}

```

```

void recherche_client(garage &g)
{
    int r = 1;
    int id;
    bool indicator = false;
    cout<<"-----+++"<<endl;
    while (r != 0)
    {
        cout<<"----- saisir l'id de client a afficher -----"<<endl;

```

```

        cin>>id;
        for(int i=0;i<g.personnes.size();i++)
        {
            if(g.personnes[i]->Get_id()==id && typeid(*g.personnes[i])==
typeid(client))
            {
                g.personnes[i]->afficher();
                indicator = true;
            }
        }
        if(indicator == false)
        {
            cout<<"!!! le client n'existe pas !!!"<<endl;
            cout<<endl;
        }
        cout<<"taper 0 pour terminer "<<endl;
        cin>>r;
    }
}

```

```

void menu_client(garage& g)
{
    int choix;
    cout<<" ----- Menu CLIENT-----"<<endl;
    cout<<"taper 1 pour afficher la liste rapide des client "<<endl;
    cout<<"taper 2 pour afficher la liste complete des client "<<endl;
    cout<<"taper 3 pour ajouter un client "<<endl;
    cout<<"taper 4 pour supprimer un client "<<endl;
    cout<<"taper 5 pour modifier un client "<<endl;
    cout<<"taper 6 pour rechercher un client "<<endl;
    cout<<"taper 0 pour retourner au menu principale "<<endl;
    cin>>choix;

    do
    {
        switch (choix)
        {
            case 1: g.afficher_clients2() ;break;
            case 2: g.afficher_clients(0);break;
            case 3: ajouter_client(g);break;
            case 4: supprimer_client(g);break;

```

```

        case 5: modifier_client(g);break;
        case 6: recherche_client(g);break;
        case 0: afficher_menu_principale(g);break;
    }
    cout<<"nouveau choix   ";
    cin>>choix;
}while(choix != 0);
}
//----- menu employee -----
-//
void ajouter_employee(garage &g)
{
    employee e;
    int r = 1;
    bool indicator = false;
    while (r != 0)
    {
        cout<<"----- saisir un employee -----"<<endl;
        cin>>e;
        for(int i=0;i<g.personnes.size();i++)
        {
            if(g.personnes[i]->Get_id()==e.Get_id() &&
typeid(*g.personnes[i])== typeid(employee))
            {
                cout<<"!!!!!! le employee existe deja !!!!!!"<<endl;
                indicator = true;
            }
        }
        if(indicator == false)
        {
            g.ajouter_personne(e);
            g.enregistrer();
            cout<<endl;
        }
        cout<<"taper 0 pour terminer "<<endl;
        cin>>r;
    }
}

void supprimer_employee(garage &g)
{

```

```

int r = 1;
int id;
g.afficher_employees2();
while (r != 0)
{
    bool indicator = false;

    cout<<"-----+++-----"<<endl;
    cout<<"----- saisir l'id de employee a supprimer -----"<<endl;
    cin>>id;
    for(int i=0;i<g.personnes.size();i++)
    {
        if(g.personnes[i]->Get_id()==id && typeid(*g.personnes[i])==
typeid(employee))
        {
            g.supprimer_personne(id);
            cout<<"supprimée"<<endl;
            g.enregistrer();
            indicator = true;
        }
    }
    if(indicator == false)
    {
        cout<<"!!! le employee n'existe pas !!!"<<endl;
        cout<<endl;
    }
    cout<<"taper 0 pour terminer "<<endl;
    cin>>r;
}
}

```

```

void modifier_employee(garage &g)
{
    int r = 1;
    int id;
    employee e;
    bool indicator = false;
    g.afficher_employees2();
    cout<<"-----+++-----"<<endl;
    while (r != 0)
    {

```

```

        cout<<"----- saisir l'id de employee a modifier -----"<<endl;
        cin>>id;
        for(int i=0;i<g.personnes.size();i++)
        {
            if(g.personnes[i]->Get_id()==id && typeid(*g.personnes[i])==
typeid(employee))
            {
                g.supprimer_personne(id);
                cin>>e;
                e.Set_id(id);
                g.ajouter_personne(e);
                g.enregistrer();
                indicator = true;
            }
        }
        if(indicator == false)
        {
            cout<<"!!! l'employee n'existe pas !!!"<<endl;
            cout<<endl;
        }
        cout<<"taper 0 pour terminer "<<endl;
        cin>>r;
    }
}

```

```

void recherche_employee(garage &g)
{
    int r = 1;
    int id;
    bool indicator = false;
    cout<<"-----+++"<<endl;
    while (r != 0)
    {
        cout<<"----- saisir l'id de employee a afficher -----"<<endl;
        cin>>id;
        for(int i=0;i<g.personnes.size();i++)
        {
            if(g.personnes[i]->Get_id()==id && typeid(*g.personnes[i])==
typeid(employee))
            {
                g.personnes[i]->afficher();
            }
        }
    }
}

```

```

        indicator = true;
    }
}
if(indicator == false)
{
    cout<<"!!! l'employee n'existe pas !!!"<<endl;
    cout<<endl;
}
cout<<"taper 0 pour terminer "<<endl;
cin>>r;
}
}

```

```

void menu_employee(garage& g)
{
    int choix;
    cout<<" ----- Menu EMPLOYEE-----"<<endl;
    cout<<"taper 1 pour afficher la liste rapide des employees "<<endl;
    cout<<"taper 2 pour afficher la liste complete des employees
"<<endl;
    cout<<"taper 3 pour ajouter un employee "<<endl;
    cout<<"taper 4 pour supprimer un employee "<<endl;
    cout<<"taper 5 pour modifier un employee "<<endl;
    cout<<"taper 6 pour rechercher un employee "<<endl;
    cout<<"taper 0 pour retourner au menu principale "<<endl;
    cin>>choix;

    do
    {
        switch (choix)
        {
            case 1: g.afficher_employees2() ;break;
            case 2: g.afficher_employees(0);break;
            case 3: ajouter_employee(g);break;
            case 4: supprimer_employee(g);break;
            case 5: modifier_employee(g);break;
            case 6: recherche_employee(g);break;
            case 0: afficher_menu_principale(g);break;
        }
        cout<<"nouveau choix  ";
        cin>>choix;
    }
}

```



```

    }while(choix != 0);
}

```

```

//----- menu piece -----//

```

```

void ajouter_piece(garage &g)
{
    piece p;
    int r = 1;
    bool indicator = false;
    while (r != 0)
    {
        cout<<"----- saisir une piece -----"<<endl;
        cin>>p;
        for(auto pi :g.pieces)
        {
            if(pi->Get_num()==p.Get_num())
            {
                cout<<"!!!!!! la piece existe deja !!!!!!"<<endl;
                indicator = true;
            }
        }
        if(indicator == false)
        {
            g.ajouter_piece(p);
            g.enregistrer();
            cout<<endl;
        }
        cout<<"taper 0 pour terminer "<<endl;
        cin>>r;
    }
}

```

```

void supprimer_piece(garage &g)
{
    int r = 1;
    int i;
    int id;
    g.afficher_pieces2();
    while (r != 0)
    {

```

```

    bool indicator = false;

    cout<<"-----+++-----"<<endl;
    cout<<"----- saisir la reference du piece a supprimer -----
"<<endl;
    cin>>id;
    for(auto pi :g.pieces)
    {
        if(pi->Get_num()==id)
        {
            g.supprimer_piece(id);
            cout<<"supprimée"<<endl;
            g.enregistrer();
            indicator = true;
            break;
        }
    }
    if(indicator == false)
    {
        cout<<"!!! la piece n'existe pas !!!"<<endl;
        cout<<endl;
    }
    cout<<"taper 0 pour terminer "<<endl;
    cin>>r;
}
}

```

```

void modifier_piece(garage &g)
{
    int r = 1;
    int id;
    piece p;
    bool indicator = false;
    g.afficher_pieces2();
    cout<<"-----+++-----"<<endl;
    while (r != 0)
    {
        cout<<"----- saisir la reference du piece a modifier -----
"<<endl;
        cin>>id;
        for(auto pi :g.pieces)

```

```

    {
        if(pi->Get_num()==id)
        {
            g.supprimer_piece(id);
            cin>>p;
            p.Set_reference(id);
            g.ajouter_piece(p);
            g.enregistrer();
            indicator = true;
        }
    }
    if(indicator == false)
    {
        cout<<"!!! la piece n'existe pas !!!"<<endl;
        cout<<endl;
    }
    cout<<"taper 0 pour terminer "<<endl;
    cin>>r;
}
}

```

```

void recherche_piece(garage &g)
{
    int r = 1;
    int id;
    bool indicator = false;
    cout<<"-----+++-----"<<endl;
    while (r != 0)
    {
        cout<<"----- saisir la reference du piece a afficher -----
"<<endl;
        cin>>id;
        for(auto pi :g.pieces)
        {
            if(pi->Get_num()==id)
            {
                pi->afficher();
                indicator = true;
            }
        }
    }
    if(indicator == false)

```

```

    {
        cout<<"!!! la piece n'existe pas !!!"<<endl;
        cout<<endl;
    }
    cout<<"taper 0 pour terminer "<<endl;
    cin>>r;
}
}

```

```

void ajouter_stock_piece(garage& g)
{
    int r = 1;
    int id,q,v;
    string msg;
    string msg2="ERREUR - la reference doit etre positive";
    string msg3="ERREUR - reference non valide";
    string msg4="ERREUR - la quantite en stock doit etre positive";
    string msg5="ERREUR - quantite en stock non valide";
    bool indicator = false;
    g.afficher_pieces2();
    cout<<"-----+++"<<endl;
    while (r != 0)
    {
        while(true)
        {
            try
            {
                cout<<"----- saisir la reference du piece -----
"<<endl;

                cin>>id;
                if(cin.fail()) throw 1;
                else if(q<0) throw msg2;
                break;
            }
            catch(string msg)
            {
                cout<<msg<<endl;
            }
            catch(int i)
            {
                cout <<msg3<<endl;
            }
        }
    }
}

```

```

        cin.clear();
        cin.ignore(numeric_limits<streamsize>::max(), '\n');
        continue;
    }
}

for(auto pi :g.pieces)
{
    if(pi->Get_num()==id)
    {
        while(true)
        {
            try
            {
                cout<<"saisir la quantité a ajouter : ";
                cin>>q;
                if(cin.fail()) throw 1;
                else if(q<0) throw msg4;
                break;
            }
            catch(string msg)
            {
                cout<<msg<<endl;
            }
            catch(int i)
            {
                cout <<msg5<<endl;
                cin.clear();
                cin.ignore(numeric_limits<streamsize>::max(), '\n');
                continue;
            }
        }
        v = pi->get_stock();
        pi->set_quantiter_stock(q + v);
        cout<<"la nouvelle quantite en stock est: "<<pi-
>get_stock()<<endl;
        indicator = true;
    }
}

if(indicator == false)
{

```

```

        cout<<"!!! la piece n'existe pas !!!"<<endl;
        cout<<endl;
    }
    cout<<"taper 0 pour terminer "<<endl;
    cin>>r;
}
}

void menu_piece(garage& g)
{
    int choix;
    cout<<" ----- Menu PIECE-----"<<endl;
    cout<<"taper 1 pour afficher la liste rapide des pieces "<<endl;
    cout<<"taper 2 pour afficher la liste complete des pieces "<<endl;
    cout<<"taper 3 pour ajouter une piece "<<endl;
    cout<<"taper 4 pour supprimer une piece"<<endl;
    cout<<"taper 5 pour modifier une piece"<<endl;
    cout<<"taper 6 pour mise a jour la quantité en stock une
piece"<<endl;
    cout<<"taper 7 pour rechercher une piece"<<endl;
    cout<<"taper 0 pour retourner au menu principale "<<endl;
    cin>>choix;

    do
    {
        switch (choix)
        {
            case 1: g.afficher_pieces2() ;break;
            case 2: g.afficher_piece(0);break;
            case 3: ajouter_piece(g);break;
            case 4: supprimer_piece(g);break;
            case 5: modifier_piece(g);break;
            case 6: ajouter_stock_piece(g);break;
            case 7: recherche_piece(g);break;
            case 0: afficher_menu_principale(g);break;
        }
        cout<<"nouveau choix  ";
        cin>>choix;
    }while(choix != 0);
}

```

```
//----- menu voiture -----//
void modifier_voiture(garage& g)
{
    int num;
    string msg;
    string msg2="ERREUR - le numero du voiture doit etre positive";
    string msg3="ERREUR - numero non valide";
    string msg4="ERREUR - la reponse doit etre E ou M";
    g.afficher_voitures();
    cout<<"-----+++-----"<<endl;
    while(true)
    {
        try
        {
            cout<<"donner le numero du voiture a changer"<<endl;
            cin>>num;
            if(cin.fail()) throw 1;
            else if(num<0) throw msg2;
            break;
        }
        catch(string msg)
        {
            cout<<msg<<endl;
        }
        catch(int i)
        {
            cout <<msg3<<endl;
            cin.clear();
            cin.ignore(numeric_limits<streamsize>::max(), '\n');
            continue;
        }
    }

    for(int i =0;i<g.reparations.size();i++)
    {
        if(g.reparations[i]->getvoiture()==num)
        {
            string a2;
            while(true)
            {
                try
```

```

        {
            cout<<"\nquele type de voiture qui sera choisit E
electrique// M mecanque ";
            cin>>a2;
            if(a2!="E" && a2!="M") throw msg4;
            break;
        }
        catch(string msg)
        {
            cout<<msg<<endl;
        }
    }

    if(a2=="E")
    {
        VElectrique ve;
        cin>>ve;
        ve.Set_num(num);
        g.reparations[i]->Set_vehicule(ve);
    }
    if(a2=="M")
    {
        VMecanique vm;
        cin>>vm;
        vm.Set_num(num);
        g.reparations[i]->Set_vehicule(vm);
    }
}
}
g.enregistrer();
}

```

```

void recherche_proprietaire(garage& g)
{
    int num,id_client;
    bool indicator = false;
    g.afficher_voitures();
    cout<<"-----+++-----"<<endl;
    cout<<"donner le numero du voiture"<<endl;
    cin>>num;
    id_client = g.voiture_client.val(num);
}

```



```

    for(int i=0;i<g.personnes.size();i++)
    {
        if(g.personnes[i]->Get_id()==id_client &&
typeid(*g.personnes[i])== typeid(client))
        {
            g.personnes[i]->afficher();
            indicator = true;
        }
    }
    if(indicator == false)
    {
        cout<<"!!! le client n'existe pas !!!"<<endl;
        cout<<endl;
    }
}

```

```

}

```

```

void menu_voiture(garage& g)
{
    int choix;
    cout<<" ----- Menu Voiture-----"<<endl;
    cout<<"taper 1 pour afficher la liste rapide des voitures en
attentes"<<endl;
    cout<<"taper 2 pour afficher la liste complete des voitures en
attentes "<<endl;
    cout<<"taper 3 pour ajouter une voiture a la file d'attentes "<<endl;
    cout<<"taper 4 pour rechercher toutes les voitures"<<endl;
    cout<<"taper 5 pour modifier une voiture"<<endl;
    cout<<"taper 6 pour rechercher proprietaire"<<endl;
    cout<<"taper 0 pour retourner au menu principale "<<endl;
    cin>>choix;

    do
    {
        switch (choix)
        {
            case 1: g.afficher_file2() ;break;
            case 2: g.afficher_file();break;
            case 3: g.ajouter_voiture();break;
            case 4: g.afficher_voitures();break;

```

```

        case 5: modifier_voiture(g);break;
        case 6: recherche_proprietaire(g);break;
        case 0: afficher_menu_principale(g);break;
    }
    cout<<"nouveau choix    ";
    cin>>choix;
}while(choix != 0);
}

//----- menu reparation -----//
void demarrer_reparation(garage& g)
{
    reparation r;
    int num,k=1,m=1;
    string reponse;
    string msg;
    string msg1="ERREUR - le numero de reparation doit etre
positive)";
    string msg2="ERREUR - le numero de reparation existe deja)";
    string msg3="ERREUR - numero de reparation non valide)";
    string msg4="ERREUR - la quntité doit etre positive)";
    string msg5="ERREUR - la quantite en stock n'est pas suffissante";
    string msg6="ERREUR - saisir un entier";
    while (m != 0)
    {
        g.file_attente.front()->afficher2();
        cout<<"----- +++ -----"<<endl;
        cout<<"demarrer la reparation avec cette voiture? Y/N  :";
        while(1)
        {
            try
            {
                cin>>reponse;
                if(reponse!= "Y" && reponse != "N") throw 1;
                break;
            }
            catch (int i)
            {
                cout<<"ERREUR - reponse non valise (Y/N)"<<endl;
            }
        }
    }
}

```

```

if(reponse == "Y")
{
    r.Set_vehicule(*g.file_attente.front());
    g.file_attente.pop();
}
if(reponse == "N")
{
    g.file_attente.pop();
    demarrer_reparation(g);
}
while(1)
{
    try
    {
        cout<<"donner le numero du reparation est : ";
        cin>>num;
        r.Set_num(num);
        if(cin.fail()) throw 1;
        else if(num < 0) throw msg1;
        else
        {
            for(int i=0;i<g.reparations.size();i++)
            {
                if(r.Get_num() == g.reparations[i]->Get_num()) throw
msg2;
            }
        }
        break;
    }
    catch(string msg)
    {
        cout<<msg<<endl;
    }
    catch(int i)
    {
        cout<<msg3<<endl;
    }
}

cout<<"donner la date de reparation : ";
date d;

```

```

cin>>d;
r.Set_date(d);
r.set_etat("active");
r.Set_num_facture(0);

bool existe1=false,existe2=false;
pile<employee> pile1,pile2;
pile<piece> pile3,pile4;
g.afficher_employees2();
cout<<"----- +++ -----"<<endl;
while(k!=0)
{
    cout<<"choisir l'employee : ";
    cin>>num;

    pile2 = pile1;
    while(!pile2.estVide())
    {
        employee e1;
        e1 = pile2.depiler();
        if(e1.Get_id()==num)
        {
            existe2 = true;
        }
    }
    for(int i=0; i<g.personnes.size();i++)
    {
        if(g.personnes[i]->Get_id()==num &&
typeid(*g.personnes[i])== typeid(employee))
        {
            existe1 = true;
            if(existe2 == false)
            {
                pile1.empiler(static_cast<const
employee&>(*g.personnes[i]));
            }
        }
    }
    cout<<"pour terminer le saisie des employee taper 0 "<<endl;
    cout<<"pour annuler l'employee precedent taper 1 "<<endl;
    cin>>k;
}

```

```

        if(k==1)
        {
            employee e2 = pile1.depiler();
            cout<<"l'employee suivant a ete annulé : "<<endl;
            e2.afficher2();
        }
    }
    while(!pile1.estVide())
    {
        employee e3 = pile1.depiler();
        r.ajouter_employer(e3);
    }

    k=1;
    g.afficher_pieces2();
    cout<<"----- +++ -----"<<endl;
    while(k!=0)
    {
        cout<<"choisir les pieces : ";
        cin>>num;

        pile4 = pile3;
        while(!pile4.estVide())
        {
            piece p1;
            p1 = pile4.depiler();
            if(p1.Get_num()==num)
            {
                existe2 = true;
            }
        }
        for(auto pi :g.pieces)
        {
            if(pi->Get_num()==num)
            {
                existe1 = true;
                if(existe2 == false)
                {
                    piece p1 = *pi;
                    int q;

```

```

//----- saisie de quantité -----
----

while(1)
{
    try
    {
        cout<<"saisir la quantité qui sera utilisé : ";
        cin>>q;
        if(cin.fail()) throw 1;
        else if(q<=0) throw msg4;
        else if(q>p1.get_stock()) throw msg5;
        break;
    }
    catch(string msg)
    {
        cout<<msg<<endl;
    }
    catch(int i)
    {
        cout<<"ERREUR - quantite non valide"<<endl;
    }
}
//----- update contité en stock -
-----

p1.set_quantiter(q);
pi->set_quantiter_stock(pi->get_stock() - q);
//----- gestion d'alarm -----
--

if(pi->get_stock()<=pi->Get_critique())
    cout<<"ALARM - la quantité en stock de la piece va
bientot expire"<<endl;
    pile3.empiler(p1);
}
}
}
cout<<"pour terminer le saisie des pieces taper 0 "<<endl;
cout<<"pour annuler la piece precedent taper 1 "<<endl;
cin>>k;
if(k==1)
{
    piece p2 = pile3.depiler();

```

```

        cout<<"la piece suivant a ete annule :"<<endl;
        p2.afficher2();
    }
}
while(!pile3.estVide())
{
    piece p3 = pile3.depiler();
    r.ajouter_piece(p3);
}
g.ajouter_reparation(r);
while(1)
{
    try
    {
        cout<<"taper 0 pour terminer "<<endl;
        cin>>m;
        if(cin.fail()) throw 1;
        break;
    }
    catch(int i)
    {
        cout<<msg6<<endl;
    }
}
if(m == 0) break;
}
}

void terminer_reparation(garage& g)
{
    int id;
    bool indicator = false;
    g.afficher_reparation2_active();
    cout<<"----- +++ -----"<<endl;
    cout<<"choisir le numero du reparation a terminer "<<endl;
    cin>>id;
    for(int i=0;i<g.reparations.size();i++)
    {
        if(g.reparations[i]->Get_num()==id && g.reparations[i]-
>Get_etat()=="active")
        {

```

```

        facture f;
        cin>>f;
        f.Set_etat("non_payee");
        f.Set_montant(g.reparations[i]->calcul_montant());
        g.reparations[i]->set_etat("terminé");
        g.reparations[i]->Set_num_facture(f.Get_num());
        g.ajouter_facture(f);
        g.enregistrer();
        indicator = true;
    }
}
if(indicator == false)
{
    cout<<"!!! le numero de reparation ne correspond a aucun
reparation active !!!"<<endl;
    cout<<endl;
}
}

```

```

void modifier_reparation(garage& g)
{
    int id;
    bool indicator = false;
    g.afficher_reparation2();
    cout<<"----- +++ -----"<<endl;
    cout<<"saisir le numero de reparation a modifier";
    cin>>id;
    for(int i=0;i<g.reparations.size();i++)
    {
        if(g.reparations[i]->Get_num()==id && g.reparations[i]-
>Get_etat()=="active")
        {
            cout<<"l'etat actuel de la repartion est : "<<endl;
            cout<<*g.reparations[i];
            g.reparations[i]->modifier_reparation();
            g.enregistrer();
            indicator = true;
        }
        else if(g.reparations[i]->Get_num()==id && g.reparations[i]-
>Get_etat()=="terminé")

```



```

        cout<<"la reparation est terminé - modification
impossible"<<endl;
    }
    if(indicator == false)
    {
        cout<<"!!! la reparation n'existe pas !!!"<<endl;
        cout<<endl;
    }
}

void supprimer_reparation(garage& g)
{
    int id;
    bool indicator = false;
    g.afficher_reparation2_active();
    cout<<"----- +++ -----"<<endl;
    cout<<"saisir le numero de reparation a supprimer";
    cin>>id;
    for(int i=0;i<g.reparations.size();i++)
    {
        if(g.reparations[i]->Get_num()==id && g.reparations[i]-
>Get_etat()=="active")
        {
            cout<<"||| supression de reparation |||"<<endl;
            g.supprimer_reparation(id);
            g.enregistrer();
            indicator = true;
        }
        else if(g.reparations[i]->Get_num()==id && g.reparations[i]-
>Get_etat()=="terminé")
            cout<<"la reparation est terminé - suppression
impossible"<<endl;
    }
    if(indicator == false)
    {
        cout<<"!!! la reparation n'existe pas !!!"<<endl;
        cout<<endl;
    }
}

void recherche_reparation(garage& g)

```

```

{
    int r = 1;
    int id;
    bool indicator = false;
    cout<<"-----++-----"<<endl;
    while (r != 0)
    {
        cout<<"----- saisir le numero de reparation a afficher -----
"<<endl;
        cin>>id;
        for(int i=0;i<g.reparations.size();i++)
        {
            if(g.reparations[i]->Get_num()==id)
            {
                g.reparations[i]->afficher();
                indicator = true;
            }
        }
        if(indicator == false)
        {
            cout<<"!!! la reparation n'existe pas !!!"<<endl;
            cout<<endl;
        }
        cout<<"taper 0 pour terminer "<<endl;
        cin>>r;
    }
}

```

```

void menu_reparation(garage& g)
{
    int choix;
    cout<<" ----- Menu REPARATION-----"<<endl;
    cout<<"taper 1 pour afficher la liste rapide des reparations"<<endl;
    cout<<"taper 2 pour afficher la liste complete des
reparations"<<endl;
    cout<<"taper 3 pour demarrer une reparation"<<endl;
    cout<<"taper 4 pour modifier une reparation"<<endl;
    cout<<"taper 5 pour supprimer une reparation"<<endl;
    cout<<"taper 6 pour rechercher une reparation"<<endl;
    cout<<"taper 7 pour terminer reparation && rediger facture"<<endl;
    cout<<"taper 0 pour retourner au menu principale "<<endl;
}

```

```

cin>>choix;

do
{
    switch (choix)
    {
        case 1: g.afficher_reparation2() ;break;
        case 2: g.afficher_reparation(0);break;
        case 3: demarrer_reparation(g);break;
        case 4: modifier_reparation(g);break;
        case 5: supprimer_reparation(g);break;
        case 6: recherche_reparation(g);break;
        case 7: terminer_reparation(g);break;
        case 0: afficher_menu_principale(g);break;
    }
    cout<<"nouveau choix  ";
    cin>>choix;
}while(choix != 0);
}

//----- menu facture -----//
void modifier_facture(garage& g)
{
    int id,m=1;
    facture f;
    bool indicator = false;
    string msg;
    string msg2="ERREUR - le numero de facture doit etre positive>";
    string msg3="ERREUR - id non valide>";
    string msg6="ERREUR - saisir un entier>";

    while (m != 0)
    {
        g.afficher_factures2();
        cout<<"----- +++ -----"<<endl;
        while(1)
        {
            try
            {
                cout<<"saisir le numero de facture a modifier>";
                cin>>id;
            }
            catch (...)
            {
                cout<<msg6<<endl;
            }
        }
    }
}

```

```

        if(cin.fail()) throw 1;
        else if(id < 0) throw msg2;
        break;
    }
    catch(string msg)
    {
        cout<<msg<<endl;
    }
    catch(int i)
    {
        cout<<msg3<<endl;
    }
}
for(int i=0;i<g.factures.size();i++)
{
    if(g.factures[i]->Get_num()==id && g.factures[i]-
>Get_etat()=="non_payee")
    {
        cout<<"l'etat actuel de la facture est :"<<endl;
        cout<<*g.factures[i];
        cin>>f;
        f.Set_etat("non_payee");
        f.Set_num(g.factures[i]->Get_num());
        g.supprimer_facture(id);
        g.ajouter_facture(f);
        g.enregistrer();
        indicator = true;
    }
    else if(g.factures[i]->Get_num()==id && g.factures[i]-
>Get_etat()=="payee")
        cout<<"la facture est payee - modification
impossible"<<endl;
    }
    if(indicator == false)
    {
        cout<<"!!! la facture n'existe pas !!!"<<endl;
        cout<<endl;
    }
}
while(1)
{
    try

```

```

    {
        cout<<"taper 0 pour terminer "<<endl;
        cin>>m;
        if(cin.fail()) throw 1;
        break;
    }
    catch(int i)
    {
        cout<<msg6<<endl;
    }
}
if(m == 0) break;
}
}

```

```

void recherche_facture(garage& g)
{
    int r = 1;
    int id;
    bool indicator = false;
    cout<<"-----+++"<<endl;
    while (r != 0)
    {
        cout<<"----- saisir le numero de facture a afficher -----
"<<endl;
        cin>>id;
        for(int i=0;i<g.factures.size();i++)
        {
            if(g.factures[i]->Get_num()==id)
            {
                cout<<*g.factures[i];
                indicator = true;
            }
        }
        if(indicator == false)
        {
            cout<<"!!! la facture n'existe pas !!!"<<endl;
            cout<<endl;
        }
        cout<<"taper 0 pour terminer "<<endl;
        cin>>r;
    }
}

```

```

    }
};

void payer_facture(garage& g)
{
    int id,r;
    date d;
    bool indicator = false;
    g.afficher_factures2_non_payee();
    cout<<"-----+++-----"<<endl;
    while (r != 0)
    {
        cout<<"----- saisir le numero de facture a payee -----"<<endl;
        cin>>id;
        for(int i=0;i<g.factures.size();i++)
        {
            if(g.factures[i]->Get_num()==id && g.factures[i]-
>Get_etat()=="non_payee")
            {
                g.factures[i]->Set_etat("payee");
                cin>>d;
                g.factures[i]->Set_date(d);
                cout<<*g.factures[i];
                indicator = true;
                g.enregistrer();
            }
            else if(g.factures[i]->Get_num()==id && g.factures[i]-
>Get_etat()=="payee")
            {
                cout<<"la facture est deja payee ";
                indicator = true;
            }
        }
        if(indicator == false)
        {
            cout<<"!!! la facture n'existe pas !!!"<<endl;
            cout<<endl;
        }
        cout<<"taper 0 pour terminer "<<endl;
        cin>>r;
    }
}

```

```
}
```

```
void menu_facture(garage& g)
{
    int choix;
    cout<<" ----- Menu FACTURE-----"<<endl;
    cout<<"taper 1 pour afficher la liste rapide des factures"<<endl;
    cout<<"taper 2 pour afficher la liste complete des factures"<<endl;
    cout<<"taper 3 pour modifier une facture"<<endl;
    cout<<"taper 4 pour rechercher une facture "<<endl;
    cout<<"taper 5 pour payee une facture"<<endl;
    cout<<"taper 0 pour retourner au menu principale "<<endl;
    cin>>choix;

    do
    {
        switch (choix)
        {
            case 1: g.afficher_factures2() ;break;
            case 2: g.afficher_facture(0);break;
            case 3: modifier_facture(g);break;
            case 4: recherche_facture(g);break;
            case 5: payer_facture(g);break;
            case 0: afficher_menu_principale(g);break;
        }
        cout<<"nouveau choix  ";
        cin>>choix;
    }while(choix != 0);
}
```

## VII. main :

```
#include <iostream>
#include "header10.h"
int main()
{
    garage g;
    g.recuperer();
    afficher_menu_principale(g);

    cout<<"terminé"<<endl;
}
```