

Berthet Laurent
Pehme Taavi

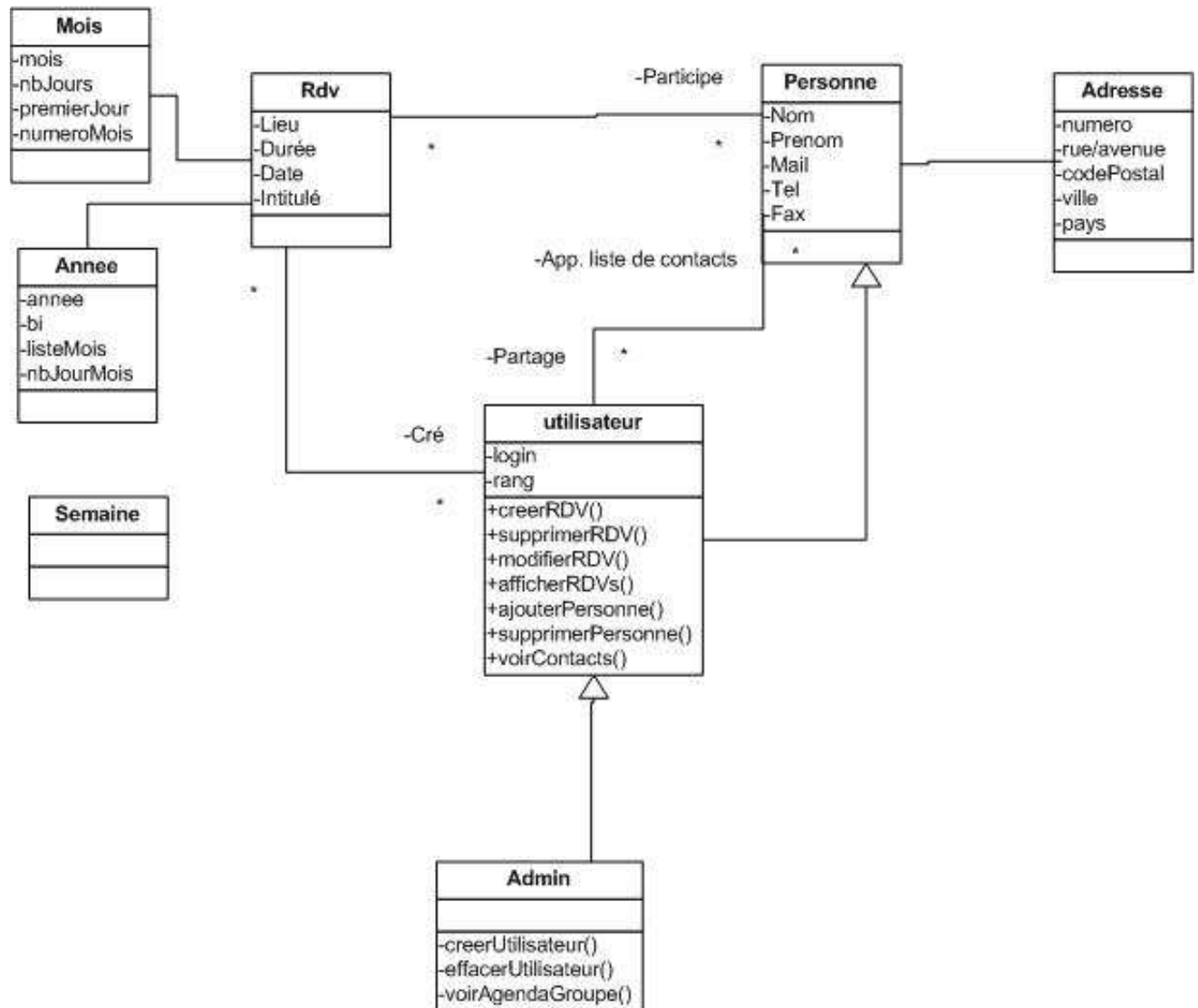
**COMPTE RENDU DU TP DE
SMALLTALK
Agenda**

Janvier 2007

Introduction

Le but de ce TP était de réaliser un agenda en Smalltalk. Pour réaliser ce TP, nous avons utilisé visualWorks 7.4.1 non commercial. Nous avons utilisé cet outils car il est assez simple d'utilisation car très ressemblant à ses versions antérieures.

Diagramme UML

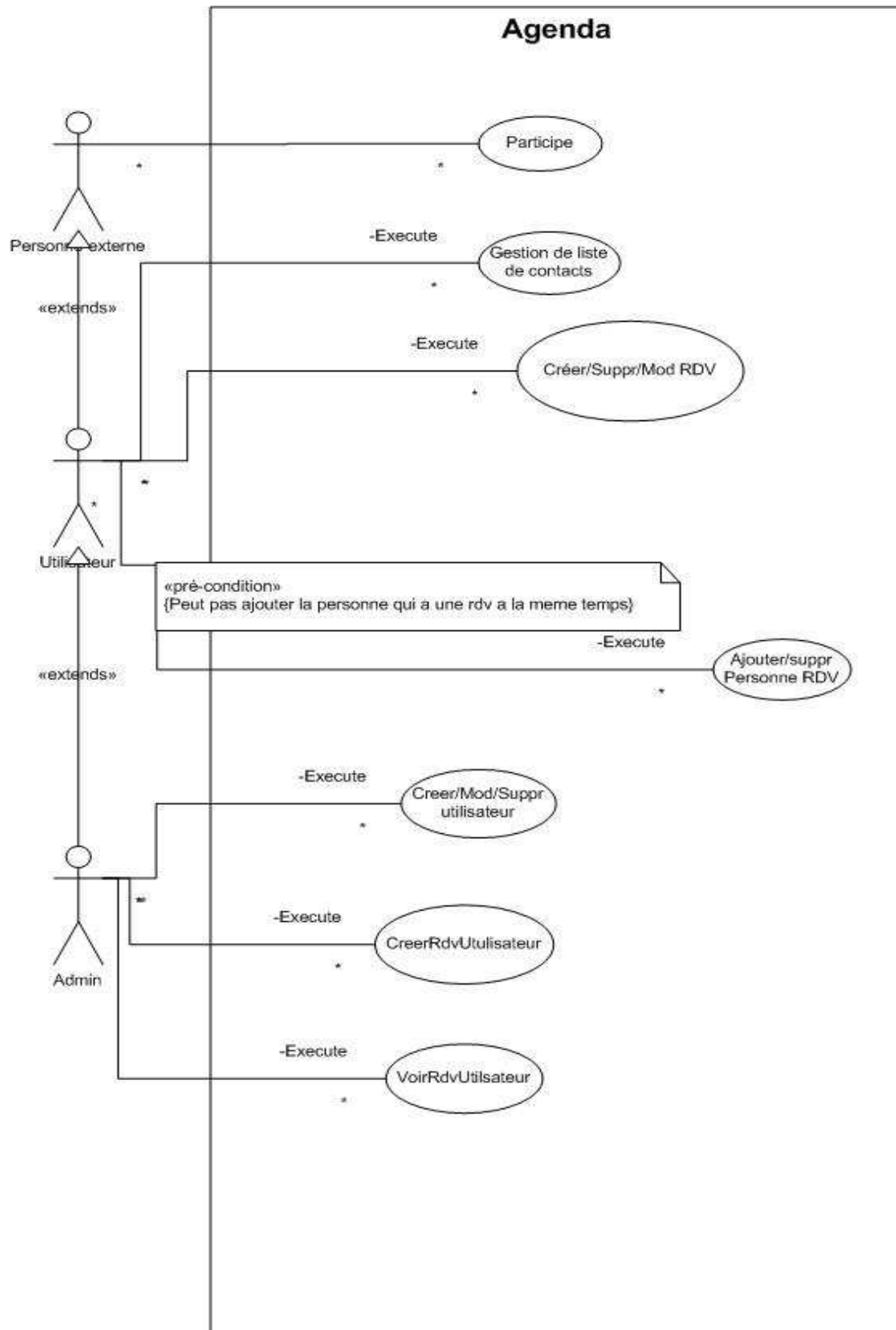


Voici le diagramme UML que nous avons réalisé pour concevoir notre agenda.

Dans notre application, il n'y a qu'un seul admin qui peut créer autant d'utilisateur qu'il souhaite. Il peut aussi supprimer un utilisateur soit en gardant les rdv soit en les supprimant aussi. Un admin peut aussi créer un Rdv pour un utilisateur. Il peut aussi visionner les Rdv d'un utilisateur.

Un utilisateur lui peut ajouter des contacts qu'il pourra ajouter dans ses rdv. Il peut aussi créer des rdv les modifier et les supprimer.

Use Case



Voici les Use case pour cette application. Les use case montre toutes les actions que les utilisateurs admin et personnes peuvent réaliser.

Diagramme de séquences

Voici un diagramme de séquence montrant la création d'un RDV. La vérification des participants l'action la plus complexe de ce diagramme.

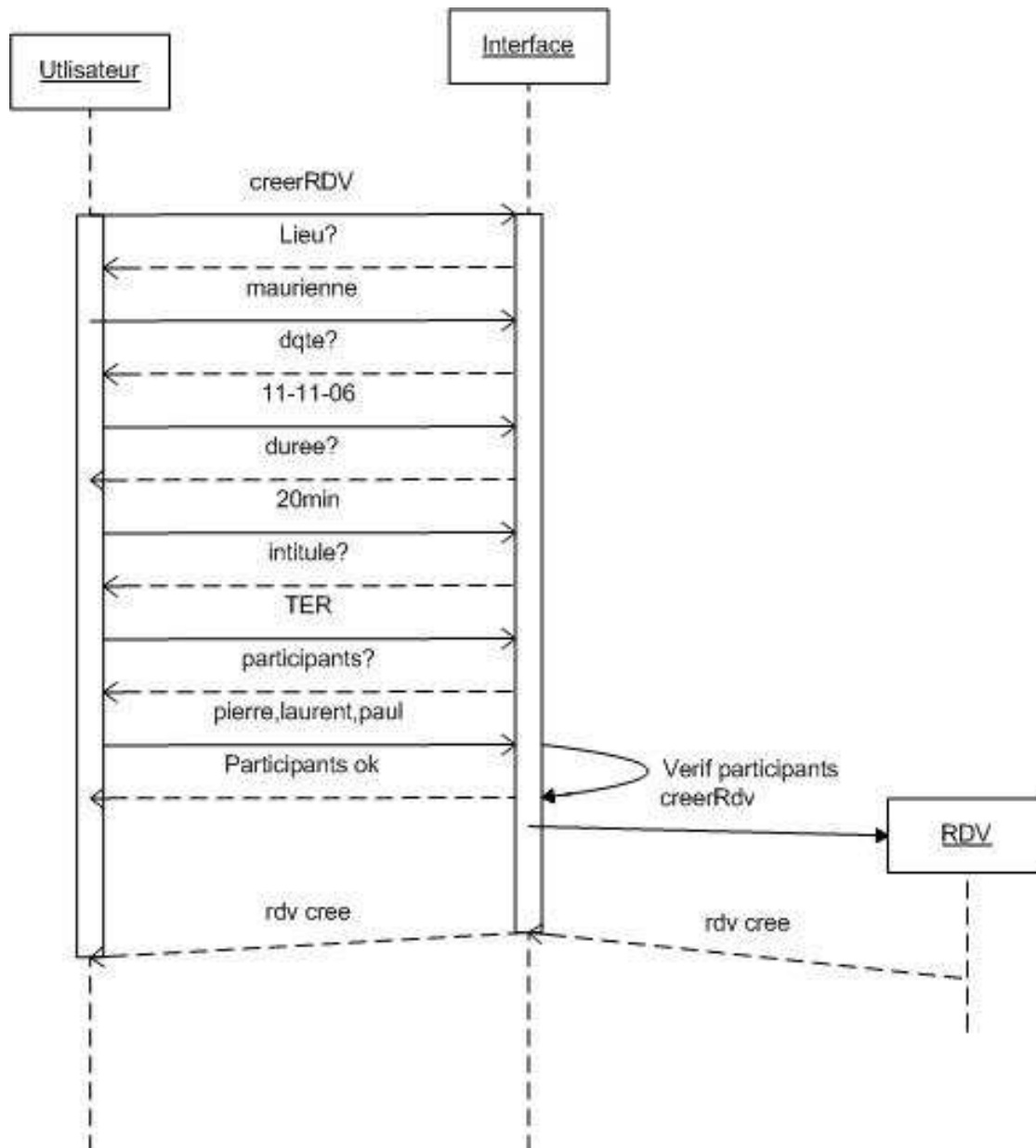
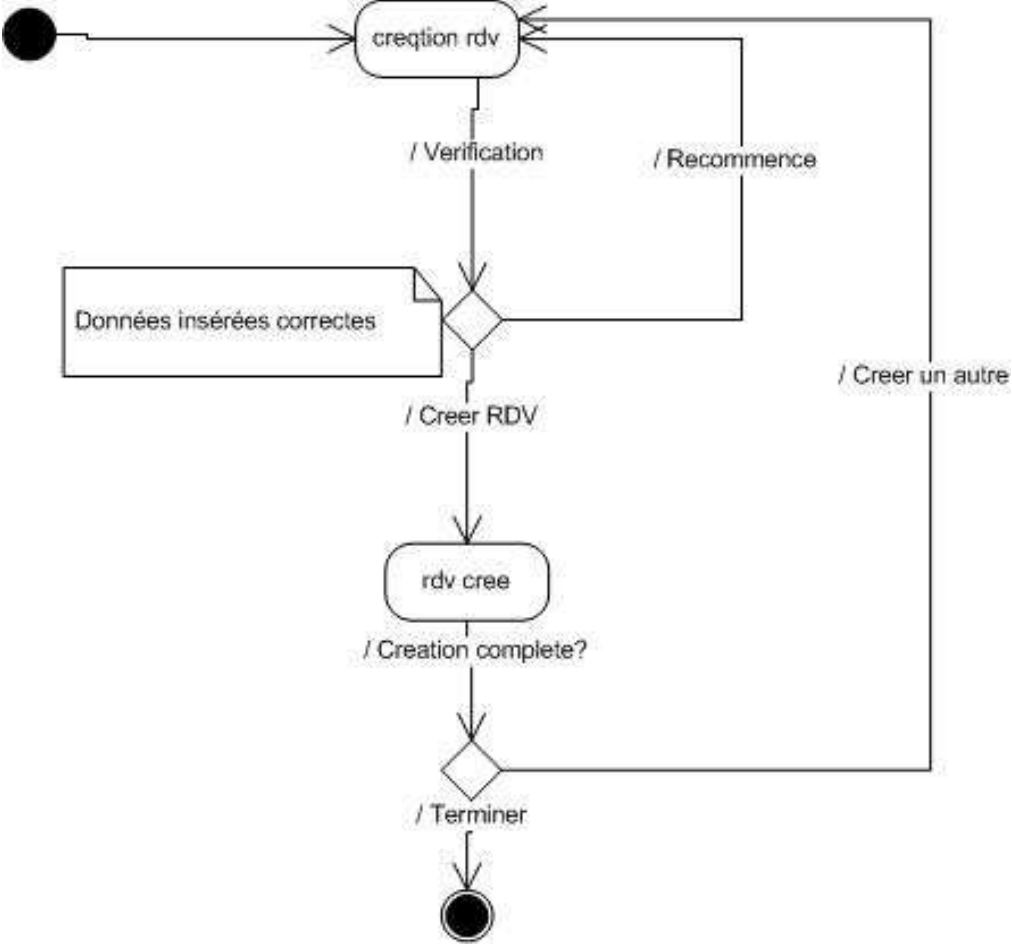


Diagramme d'état

Voici un diagramme d'état montrant la création d'un Rdv



Conclusion

Ce TP nous a permis d'apprendre rapidement le langage Smalltalk qui n'est pas l'un des plus simples ainsi que de voir ce qu'un vrai langage objet.

Avec plus de temps nous aurions pu ajouter d'autres fonctionnalités ainsi que améliorer l'interface graphique.

Sources du model:

Admin

Methodes d'instance

initialize

```
listeUtilisateurs :=List new.  
listeRdv := List new.  
adresse := Adresse new.  
login := ".  
password := "
```

initialize: u

```
self nom: u nom.  
self prenom: u prenom.  
self login: u login.  
self password: u password.  
listeUtilisateurs :=List new.  
listeRdv := List new.  
adresse := Adresse new.
```

initialize: n and: p

```
nom := n.  
prenom := p.  
listeUtilisateurs :=List new.  
listeRdv := List new.  
adresse := Adresse new.  
login := ".  
password := "
```

listeUtilisateurs

```
^listeUtilisateurs
```

listeUtilisateurs :l

```
listeUtilisateurs := l.
```

creerUtilisateur: n and: p and: l and: pass

```
| u |  
u := Utilisateur new:n and: p and: l and: pass.  
listeUtilisateurs add:u.  
Dialog warn: 'utilisateur crée '.
```

effacerUtilisateur: u

```
listeUtilisateurs remove: u.
```

Methodes de classe

new: n and: p

^super new initialize: n and: p.

new

^super new initialize

new: u

^super new initialize: u

Adresse

Methodes d'instance**initAdresse**

numero:=".

ville:=".

codePostal := ".

pays:=".

rue := ".

initAdresse: n and: r and: cp and: v and: p

numero:=n.

ville:=v.

codePostal := cp.

pays:=p.

rue := r.

Methodes de classe**new: n and: r and: cp and: v and: p**

^super new initAdresse:n and: r and: cp and: v and: p.

new

^super new initAdresse

Année

Methodes d'instance**init_annee: a**

annee := a.

listeMois := #('January' 'February' 'March' 'April' 'May' 'June' 'July' 'August' 'September' 'October' 'November' 'December').

nbJoursMois := #(31 28 31 30 31 30 31 31 30 31 30 31).

init_annee

annee := 2006.

bi := False.

```
listeMois := #('January' 'February' 'March' 'April' 'May' 'June' 'July' 'August' 'September'
'October' 'November' 'December').
nbJoursMois := #(31 28 31 30 31 30 31 31 30 31 30 31 ).
```

```
numeroMois: m
^listeMois indexOf:m
```

Methodes de classe

```
new:a
^super new init_annee:a.
```

```
new
^super new init_annee
```

Mois

Methodes d'instance

```
init: m
mois := m.
```

```
calculePremierJour: jour and:js
| j s jourS premier ls |
j := jour\\7.
" creation d'une semaine"
s := Semaine new.
" recuperation de la liste des jours d'une semaine"
ls := s listeJours.
" jourS est egal a la position de js dans ls"
jourS := ls indexOf: js.
" calcul du premier jour"
premier := jourS - j.
" premier doit etre positif"
premier > 0 ifFalse: [ premier := premier +7].
^ls at:premier.
```

Methode de classe

```
new: m
^super new init: m.
```

RDV

Methodes d'instances

```
init
```

lieu:="".
intitule:="".
heure:="".
jour:="".

init: j and: m and: a and: h and: min and: l and: i and: d

lieu:= l.
intitule:= i.
heure:=h.
minute := min.
jour:=j.
mois := m.
annee := a.
duree := d.

init: j and: m and: a and: h and: min and: l and: i and: d and: liste

lieu:= l.
intitule:= i.
heure:=h.
minute := min.
jour:=j.
annee:= a.
mois := m.
duree := d.
listePersonnes := liste.

Methodes de classe

new:j and: m and: a and: h and: min and: l and: i and:d and: liste

^super new init:j and: m and: a and: h and: min and: l and: i and: d and: liste.

new:j and: m and: a and: h and: min and: l and: i and:d

^super new init:j and: m and: a and: h and: min and: l and: i and: d.

new

^super new init

Semaine

Methodes d'instances

init

numero:=0.

listeJours := #('Monday' 'Tuesday' 'Wednesday' 'Thursday' 'Friday' 'Saturday' 'Sunday').

getJour: j

"retourne le jour d'indice j "

| i |

i:= listeJours indexOf:j.

^i

Methode de classe

new

^super new init.

Utilisateur

Methode d'instance

initialize

nom := "".

prenom := "".

login := "".

password := "".

listeRdv := List new.

mesContacts := List new.

initialize:n

nom :=n.

prenom := "".

login := "".

password := "".

listeRdv := List new.

mesContacts := List new.

initialize:n and: p and: l and: pass

nom :=n.

prenom := p.

login := l.

password := pass.

listeRdv := List new.

mesContacts := List new.

modifierRdv: rdv and: j and: m and: a and: h and: min and: l and: i and: d

" pour modifier un rdv, on efface le rdv puis on en recrée un "

listeRdv remove: rdv.

self nouveauRdv: j and: m and: a and: h and: min and: l and: i and: d and: rdv listePersonnes.

nouveauRdv: rdv

(self verifRdv: rdv) ifTrue:[

 listeRdv add: rdv.

].

**nouveauRdv: jour and: mois and: annee and: heure and: minute and: lieu and: intitule
and: duree**

| rdv |

(jour = 0) ifFalse: [

```
rdv := Rdv new: jour and: mois and: annee and: heure and: minute and: lieu and:
intitule and: duree.
```

```
“ on verifie que l’on peut ajouter le rdv “
(self verifRdv: rdv) ifTrue: [
    listeRdv add: rdv.
    Dialog warn:'rdv crée'.
]
ifFalse: [
    “ il y a deja un rdv a la meme heure “
    (Dialog confirm:'vous avez deja un rdv à cette heure voulez quand meme
ajouté ce rdv?') ifTrue:[
        listeRdv add: rdv.
        Dialog warn:'rdv ajouté'.
    ].
].
]
ifTrue: [
    Dialog warn:'La date est incorrecte'.
].
```

nouveauRdv: jour and: mois and: annee and: heure and: minute and: lieu and: intitule and: duree and: liste

```
| rdv |
(jour = 0) ifFalse: [
    rdv := Rdv new: jour and: mois and: annee and: heure and: minute and: lieu and:
intitule and: duree and: liste.
    “ on verifie que l’on peut ajouter le rdv “
    (self verifRdv: rdv) ifTrue: [
        listeRdv add: rdv.
        "self ajouteRdvListe: rdv and: liste."
        Dialog warn:'rdv crée'.
    ]
    ifFalse: [    il y a deja un rdv a la meme heure “
        (Dialog confirm:'vous avez deja un rdv à cette heure voulez quand meme
ajouté ce rdv?') ifTrue:[
            listeRdv add: rdv.
            Dialog warn:'rdv ajouté'.
        ].
    ].
]
ifTrue: [
    Dialog warn:'La date est incorrecte'.
].
```

supprimerRdv: rdv

```
" on supprime le rdv de la liste "
listeRdv remove: rdv.
```

verifRdv: rdv

```

| indice pasTrouve taille rdvCourant |
pasTrouve := true.
(listeRdv size >0) ifTrue: [
    indice := 1.
    " on parcourt tous les rdv pour verifier que le rdv est correct "
    " on verifie qu'il n'y a pas deja un rdv a la meme date "
    taille := listeRdv size.
    [(indice <= taille) & (pasTrouve) ] whileTrue:    [
        rdvCourant := listeRdv at:indice.
        " on verifie la date et l'heure du rdv "
        (((rdv jour) = (rdvCourant jour)) & ((rdv mois) = (rdvCourant mois)) & ((rdv
annee) = (rdvCourant annee)) & ((rdv heure) = (rdvCourant heure)) & ((rdv minute) =
(rdvCourant minute))))ifTrue:[
            pasTrouve := false.
        ].
        indice := indice + 1.
    ].
].
^pasTrouve.

```

Methodes de classe

new: n and: p and: l and: pass

^super new initialize: n and:p and: l and: pass.

new: n

^super new initialize: n.

new

^super new initialize

Interface graphique :

Enregistrement

initialize

fichierAdmin := 'admin.bos' asFilename.

annuler

self closeRequest.

ok

```
] admin fichier stream bos liste indice pasTrouve p str b |
  l := login value.
  p := password value.
  " en fonction de la valeur du login on verifie si c'est un utilisateur ou un admin"
" si le fichier admin existe"
(fichierAdmin exists) ifTrue:[
  str := fichierAdmin readStream.
  b := BinaryObjectStorage onOld: str.
  admin := b next.
  b close.
  str close.
  " on verifie si le login et le pass sont ceux de l'admin "
  ((l = admin login) & (p = admin password)) ifTrue:[
    admin login: l.
    admin password: p.
    self closeRequest.
    Menu_principal new: admin.
  ]
  " ce n'est pas un admin "
  ifFalse:[
    " on va verifier si le login est un login d'un utilisateur "

    fichier := 'utilisateur.bos' asFilename.
    " on verifie que le fichier existe "
    (fichier exists) ifTrue:[
      stream := fichier readStream.
      bos := BinaryObjectStorage onOld: stream.
      liste := bos next.
      stream close.
      bos close.
      " on va parcourir toute la liste "
      indice := 1.
      pasTrouve := true.
      " on verifie tous les login et pass jusqu'a trouve le bon "
      [((indice <= liste size) & (pasTrouve))] whileTrue: [
        utilisateur := liste at:indice.
        ((utilisateur login = l) & (utilisateur password = p)) ifTrue:[
          pasTrouve := false.
        ]
      ].
    ]
  ]
]
```

```

        indice := indice + 1.
    ].
    " on a pas trouvé donc il faut entrer un nouveau login et pass "
    (pasTrouve) ifTrue:[
        Dialog warn: 'login incorrect'.
        login value:".
        password value:".
    ]
    " le login et le pass sont bon donc on ouvre une nouvelle fenetre "
    ifFalse:[
        Menu_utilisateur new:utilisateur.
        " on fenetre la fenetre courante "
        self closeRequest.
    ].
]
" le fichier n'existe pas donc aucun utilisateur ne peut se logger "
ifFalse:[
    Dialog warn:'login incorrect '.
    login value:".
    password value:".
].
].
]
" le fichier admin n'existe pas "
ifFalse:[
    " on verifie si c'est un admin "
    ((l = 'admin') & (p = 'password')) ifTrue: [
        " on va créer un admin "
        admin := Admin new: 'chef' and:'chef'.
        " on ouvre la fenetre menu principal "
        Menu_principal new:admin.
        " on fenetre la fenetre courante "
        self closeRequest.
    ]
    ifFalse:[
        " on va verifier si le login est un login d'un utilisateur "

        fichier := 'utilisateur.bos' asFilename.
        " on verifie que le fichier existe "
        (fichier exists) ifTrue:[
            stream := fichier readStream.
            bos := BinaryObjectStorage onOld: stream.
            liste := bos next.
            stream close.
            bos close.
            " on va parcourir toute la liste "
            indice := 1.
            pasTrouve := true.
            " on verifie tous les login et pass jusqu'a trouve le bon "
            [((indice <= liste size) & (pasTrouve))] whileTrue: [

```

```

        utilisateur := liste at:indice.
        ((utilisateur login = l) & (utilisateur password = p)) ifTrue:[
            pasTrouve := false.
        ].
        indice := indice +1.
    ].
    " on a pas trouvé donc il faut entrer un nouveau login et pass "
    (pasTrouve) ifTrue:[
        Dialog warn: 'login incorrect'.
        login value:".
        password value:".
    ]
    " le login et le pass sont bon donc on ouvre une nouvelle fenetre"
    ifFalse:[
        Menu_utilisateur new:utilisateur.
        " on fenetre la fenetre courante "
        self closeRequest.
    ].
]
" le fichier n'existe pas donc aucun utilisateur ne peut se logger "
ifFalse:[
    Dialog warn: 'login incorrect'.
    login value:".
    password value:".
].
].
].

```

Menu_principal

initialize: a

```

admin := a.
self recupListe.
" on ouvre la fenetre "
self open.

```

recupListe

```

| fichier stream bos tabUtilisateur indice |
    fichier := 'utilisateur.bos' asFilename.
fichier exists ifTrue:[
    " on recupere la liste d'utilisateur qui est dans le fichier bos "
    stream := fichier readStream.
    bos := BinaryObjectStorage onOld:stream.
    tabUtilisateur := bos next.
    stream close.
    bos close.
    " on ajoute tous les utilisateurs dans une liste "
    indice := 1.
    admin listeUtilisateurs: List new.

```

```
        [(indice <= tabUtilisateur size)] whileTrue: [  
            admin listeUtilisateurs add: (tabUtilisateur at: indice).  
            indice := indice+1.  
        ].  
    ].
```

Actions

ajouterRdv

```
self closeRequest.  
MesInterfaces new:self.
```

modif

```
self closeRequest.  
Donnees_perso new:self.
```

newRdv

```
self closeRequest.  
NewRdvPersonne new: admin.
```

nouv_util

```
self closeRequest.  
Nouvel_utilisateur new: admin.
```

quitter

```
self closeRequest.  
Enregistrement open.
```

rdv

```
self closeRequest.  
ListeRdv new:self.
```

Methode de classe

new: a

```
^super new initialize:a.
```

Menu_utilisateur

initialize: u

```
utilisateur := u.  
"on ouvre la fenetre "  
self open.
```

Actions

modifPassword

```
self closeRequest.  
Modif_password new: self.
```

newContact

self closeRequest.

Nouveau_contact new: self.

nouveauRdv

self closeRequest.

MesInterfaces new:self.

quitter

self closeRequest.

Enregistrement open.

voirRdv

self closeRequest.

ListeRdv new:self.

Methode de classe

new: u

^super new initialize:u.

Donnees_perso**intialize: a**

| adresse |

" on recupere la fenetre parent "

parent := a.

admin := a personne.

fichier := 'admin.bos' asFilename.

" on ouvre la fenetre "

self open.

" on initialise tous les champs avec les valeurs de l'admin "

champNom value: admin nom.

champPrenom value: admin prenom.

champLogin value: admin login.

champPass value: admin password.

champMail value: admin mail.

champTel value: admin tel.

adresse := admin adresse.

"champNumero value: (adresse numero)."

"champCp value: (adresse codePostal)."

champRue value: (adresse rue).

champVille value: adresse ville.

champPays value: adresse pays.

valider

| str b ad |

((champLogin value=") | (champPass value = ")) ifTrue:[

```

        Dialog warn: 'vous devez remplir les champs login et password '
    ]
    ifFalse:[
        admin nom: champNom value.
        admin prenom: champPrenom value.
        admin login: champLogin value.
        admin password: champPass value.
        ad := Adresse new:champNumero value and: champRue value and: champCp and:
        champVille value and: champPays value.
        " on enregistre les donnée de l'admin "
        str := fichier writeStream.
        b := BinaryObjectStorage onNew: str.
        b nextPut: admin.
        str close.
        b close.
        " on enregistre l'adresse dans un fichier séparé "
        admin adresse: ad.
        "fichierAdresse := 'adresseAdmin.bos ' asFilename.
        str := fichierAdresse writeStream.
        b := BinaryObjectStorage onNew: str.
        b nextPut: admin.
        str close.
        b close."
        self closeRequest.
        parent initialize: admin.
    ].

```

raz

```

" dans le cas ou l'admin change reelement"
(Dialog confirm:'Etes vous sur de vouloir remettre à zero?') ifTrue:[
    fichier delete.
    fichierAdresse delete.
    self closeRequest.
    Enregistrement open.
].

```

annuler

```

self closeRequest.
parent initialize: admin.

```

ListeRdv

initialize: p

```

" on recupere utilisateur qui vient de se connecter "
parent := p.
utilisateur := p personne.
" on recupere la liste des ses rdv "
chemin := 'Rdv\'.
chemin := chemin,utilisateur nom,'_'.

```

```

chemin := chemin,utilisateur prenom.
chemin := chemin,'.bos'.
fichier := chemin asFilename.
(fichier exists) ifTrue:[
    stream := fichier readStream.
    bos := BinaryObjectStorage onOld: stream.
    utilisateur listeRdv: bos next.
    bos close.
    stream close.
    nbPersonnes := 0.
    nbRdv := utilisateur listeRdv size.
    self creerTableRdv.
    self initSightingsTableRdv.
    self remplirTableRdv.
    self metLabelRdv.
    self creerTablePersonnes.
    self initSightingsTablePersonnes.
    self metLabelPersonne.

    self open.
]
ifFalse:[
    parent initialize: utilisateur.
    Dialog warn:'il faut creer un Rdv '.
].

```

creerTablePersonnes

```

contenuTablePersonnes := TwoDList columns:2 rows:nbPersonnes.
self tableListePersonnes table: contenuTablePersonnes.

```

creerTableRdv

```

contenuTableListeRdv := TwoDList columns: 4 rows: nbRdv.
self tableRdv table: contenuTableListeRdv.

```

initSightingsTablePersonnes

```

sightingsTablePersonnes := SelectionInTable with: contenuTablePersonnes.
tableListePersonnes := TableInterface new selectionInTable: sightingsTablePersonnes.

```

initSightingsTableRdv

```

sightingsTableRdv := SelectionInTable with: contenuTableListeRdv.
tableRdv := TableInterface new selectionInTable: sightingsTableRdv.

```

metLabelPersonne

```

tableListePersonnes columnLabelsArray: #('Nom' 'Prenom').

```

metLabelRdv

```

tableRdv columnLabelsArray: #('Date' 'Lieu' 'Intitule' 'Duree').

```

remplirTablePersonnes

```

| indice i p|

```

```

indice := 1.
i := 1.
[(indice <= nbPersonnes)] whileTrue:[
    p := rdvSelect listePersonnes at: indice.
    contenuTablePersonnes at: i put: p nom.
    i := i+1.
    contenuTablePersonnes at: i put: p prenom.
    i := i + 1.
    indice := indice +1.
]

```

remplirTableRdv

```

| indice a rdvCourant numero date i |
    indice := 1.
    i := 1.
    a := Annee new.
    [(indice <= nbRdv)] whileTrue: [
        rdvCourant := utilisateur listeRdv at:indice.
        numero := a numeroMois: rdvCourant mois asString.
        date := rdvCourant jour printString.
        date := date, '/', numero printString.
        date := date, '/', rdvCourant annee printString.
        contenuTableListeRdv at: i put: date.
        i := i+1.
        contenuTableListeRdv at: i put: rdvCourant lieu.
        i := i+1.
        contenuTableListeRdv at: i put: rdvCourant intitule.
        i := i+1.
        contenuTableListeRdv at: i put: rdvCourant duree.
        i := i+1.
        indice := indice +1.
    ].

```

Actions

annuler

```

self closeRequest.
parent initialize:utilisateur.

```

dbClick

```

| v ligne |
    " on recupere le numero de la ligne sur laquelle ona cliqué "
    v := self sightingsTableRdv selectionIndex.
    ligne := v y.
    " on recupere le rdv de cette ligne "
    rdvSelect := utilisateur listeRdv at:ligne.
    " on affecte tous les champs avec les valeurs du rdv "
    champAnnee value: rdvSelect annee.
    champMois value: rdvSelect mois asString.
    champJour value: rdvSelect jour.

```

```

champIntitule value: rdvSelect intitule.
champHeure value: rdvSelect heure.
champMinute value: rdvSelect minute.
champDuree value: rdvSelect duree.
champLieu value: rdvSelect lieu.
Dialog warn: rdvSelect listePersonnes size printString,' personnes vont assister au rdv'.
" on remplit la table des personnes "
nbPersonnes := rdvSelect listePersonnes size.
self creerTablePersonnes.
self remplirTablePersonnes.

```

modifier

```

" on recupere le numero de la ligne sur laquelle on a cliqué "
| v ligne indice date a m |
v := self sightingsTableRdv selectionIndex.
ligne := v y.
(ligne = 0) iffFalse: [
    " on va modifier le Rdv selctionné "
    utilisateur modifierRdv: rdvSelect and: champJour value and: champMois
value and: champAnnee value and: champHeure value and: champMinute value and:
champLieu value and: champIntitule value and: champDuree value.
    " on met a jour l'affichage "
    indice := 1 + ((ligne-1)*4).
    a := Annee new.
    date := champJour value printString.
    m := a numeroMois: champMois value asString.
    date := date, '/', m printString.
    date := date := date, '/', champAnnee value printString.
    contenuTableListeRdv at: indice put: date.
    indice := indice+1.
    contenuTableListeRdv at: indice put: champLieu value.
    indice := indice+1.
    contenuTableListeRdv at: indice put: champIntitule value.
    indice := indice+1.
    contenuTableListeRdv at: indice put: champDuree value.
]

```

supprimer

```

" supprime une personne participant au rdv de la liste "
| v l p |
v := self sightingsTablePersonnes selectionIndex.
l := v y.
p := rdvSelect listePersonnes at:l.
rdvSelect listePersonnes remove:p.
nbPersonnes := nbPersonnes -1.
self creerTablePersonnes.
self remplirTablePersonnes.

```

supprimeRdv

```

“ supprime un rdv “
| v l rdv |
v := self sightingsTableRdv selectionIndex.
l := v y.
rdv := utilisateur listeRdv at: l.
utilisateur listeRdv supprimerRdv: rdv.
nbRdv := nbRdv -1.
self creerTableRdv.
self remplirTableRdv.

```

valider

```

" on enregistre les rdv dans le fichier bos "
stream := fichier writeStream.
bos := BinaryObjectStorage onNew: stream.
[bos nextPut: utilisateur listeRdv    ] valueNowOrOnUnwindDo: [bos close].
stream close.
" on ferme la fenetre courrante "
self closeRequest.
" on ouvre la fenetre des menus "
parent initialize:utilisateur.

```

MesInterfaces

initialize: p

```

| l min dur name |
parent := p.
" on recupere utilisateur qui vient de se connecter "
utilisateur := p personne.
" on recupere la liste des ses rdv "
chemin := 'Rdv\'.
chemin := chemin,utilisateur nom,'_'.
chemin := chemin,utilisateur prenom.
chemin := chemin,'.bos'.
fichier := chemin asFilename.
(fichier exists) ifTrue:[
    self recupListeRdv.
]
ifFalse: [
    utilisateur listeRdv: List new.
].
"self recupListeContact."
" on initialise la Table calendrier "
self creerTableCalendrier.
self remplirTableCalendrier.
self initSightingsTable.
self metLabelTableCalendrier.
" on initialise la table des personnes "
nbPersonnes := 0.
self initTablePersonne.

```

```

self metLabelTablePersonne.
listPersonnes := List new.
" on lie les champ champRdvMois et champRdvAnnee aux champs champMois et
champAnnee "
champRdvMois := champMois.
champRdvAnnee := champAnnee.
" on initialise les listes des heures et des minutes "
"-----"
champHeure := 8 asValue.
l := List new.
l add:8;
add:9;
add: 10;
add:11;
add:13;
add: 14;
add:15;
add:16;
add:17;
add:18;
add:19.
choixHeures := l asValue.
champMinute := 0 asValue.
min := List new.
min add:0;
add:15;
add:30;
add:45.
choixMinutes := min asValue.
champDuree := 1 asValue.
dur := List new.
dur add:1;
add:2;
add:3;
add:4.
choixDuree := dur asValue.
"-----"
"on ouvre la fenetre apres avoir tous initialisé "
self open.
name := utilisateur nom.
champNom value:name.

```

creerTableCalendrier

```

contenuTableCalendrier := TwoDList columns: 7 rows: 6.
self champRdv table: contenuTableCalendrier.

```

initSightingsTable

```

sightingsTableCalendrier := SelectionInTable with: contenuTableCalendrier.
champRdv := TableInterface new selectionInTable: sightingsTableCalendrier.

```

initTablePersonne

```
contenuTablePersonnes := TwoDList columns: 2 rows: nbPersonnes.  
self tablePersonnes table: contenuTablePersonnes.
```

metLabelTableCalendrier

```
champRdv columnLabelsArray: #('M' 'Tu' 'W' 'Th' 'F' 'Sa' 'Su');  
rowLabelsArray: #(1 2 3 4 5 6).
```

metLabelTablePersonne

```
tablePersonnes columnLabelsArray: #('Nom' 'Prenom').
```

recupListeContact

```
| ch f st b |  
ch := 'Contacts\  
ch := ch,utilisateur nom,'_'.  
ch := ch,utilisateur prenom.  
ch := ch,'.bos'.  
f := ch asFilename.  
(f exists) ifTrue: [   
    st := fichier readStream.  
    b := BinaryObjectStorage onOld:st.  
    utilisateur listeContacts: b next.  
    b close.  
    st close.  
]
```

recupListeRdv

```
stream := fichier readStream.  
bos := BinaryObjectStorage onOld:stream.  
utilisateur listeRdv: bos next.  
bos close.  
stream close.
```

remplirTableCalendrier

```
"on recupere le mois de la date courante"  
| m annee d numJour nbJour indice j i |  
m := Date today monthName.  
" on l'affecte au champ"  
self champMois value:m.  
" on cree un mois"  
mois_courant := Mois new:m.  
" on recupere l'année courante "  
annee := Date today year.  
" on l'affecte au champ"  
self champAnnee value: annee.  
" on cree une année"  
annee_courante := Annee new:annee.  
d := Date newDay:1 month: mois_courant mois year:annee_courante annee.  
numJour := d weekdayIndex.
```

```

nbJour := d daysInMonth.
" on initialise la list contenant la liste des jours du mois "
indice := 1.
" on met des blancs jusqu'au premier jour du mois "
[indice < numJour] whileTrue: [
    contenuTableCalendrier at: indice put: ".
    indice := indice+1
].
j:= 1.
i := 1.
" on remplit le tableau "
[i < nbJour] whileTrue: [
    contenuTableCalendrier at: indice put:j.
    j:=j+1.
    i:=i+1.
    indice := indice+1
].
" on met des blancs jusqu'a la fin du tableau "
[indice <= 42] whileTrue: [
    contenuTableCalendrier at: indice put: ".
    indice := indice+1
].

```

Acions

ajouterPersonne

```

| indiceTablePersonnes nouvellePersonne p pasTrouve i |
((champNomPersonne value =) & (champPrenomPersonne value = "")) ifFalse:[
    " on verifie que la personnes et dans ca liste de contact "
    pasTrouve := true.
    i := 1.
    [(i <= utilisateur mesContacts size) & (pasTrouve)] whileTrue:[
        p := utilisateur mesContacts at: i.
        ((p nom = champNomPersonne value) & ( p prenom = champPrenomPersonne
value)) ifFalse:[
            pasTrouve := false.
        ].
        i := i+1.
    ].
    " si la personne est presente dans la liste "
    (pasTrouve) ifTrue:[
        nbPersonnes := nbPersonnes + 1.
        self initTablePersonne.
        nom := champNomPersonne value.
        prenom := champPrenomPersonne value.
        nouvellePersonne := Personne new:nom and: prenom.
        listPersonnes add: nouvellePersonne.
        indiceTablePersonnes := 1.
        n := 1.
    ].

```

```

        [(n <= nbPersonnes)] whileTrue: [
            p := listPersonnes at: n.
            contenuTablePersonnes at:indiceTablePersonnes put: p nom.
            indiceTablePersonnes := indiceTablePersonnes+1.
            contenuTablePersonnes at:indiceTablePersonnes put: p prenom.
            indiceTablePersonnes := indiceTablePersonnes+1.
            n := n+1.
        ].
    Dialog warn:'vous avez ajouté ',nom,' ',prenom.
].

    champNomPersonne value:".
    champPrenomPersonne value:".
]
ifTrue:[
    Dialog warn: ' vous n"avez pas remplis tous vous champs '.
].

```

annuler

```

(Dialog confirm: 'Voulez vous quitter ?') ifTrue: [
    self closeRequest.
    parent initialize:utilisateur.
]

```

dbClick

```

| v |
v := self sightingsTableCalendrier selection.
(v = "") ifFalse: [ self champJour value: v.].

```

deconnexion

```

(Dialog confirm:'Etes vous sur de vouloir vous deconnecter?') ifTrue: [
    self closeRequest.
    Enregistrement open.
].

```

suivant

```

|a ms m1 indexMois nb indice j i d premierJour |
m1 := champMois value.
ms := annee_courante moisSuivant: m1.
self champMois value: ms.
ms = 'January' ifTrue:[
    a :=champAnnee value + 1.
    self champAnnee value: a.
    annee_courante := Annee new:a.
].
" on recupere le nombre de jours du mois suivant"
mois_courant mois:ms.
" on recupere le numero du mois "
indexMois := annee_courante listeMois indexOf:ms.
" nb est le nb de jour dans le mois "

```

```

nb := annee_courante nbJoursMois at: indexMois.
mois_courant nbJours: nb.
" on cree la date "
d := Date newDay:1 month: mois_courant mois year:annee_courante annee.
" on recupere le numero du jour "
premierJour := d weekdayIndex.
" on va mettre a jour la table "
indice := 1.
(premierJour < 7) ifTrue: [
    [indice <= premierJour] whileTrue: [contenuTableCalendrier at: indice put: ". indice :=
indice+1].
    j:= 1.
    i := 1.
    [i <= mois_courant nbJours] whileTrue: [contenuTableCalendrier at: indice put:j.
j:=j+1.i:=i+1. indice := indice+1].
    [indice <= 42] whileTrue: [contenuTableCalendrier at: indice put: ". indice :=
indice+1].
]
    ifFalse: [
        j:= 1.
        i := 1.
        [i <= mois_courant nbJours] whileTrue: [contenuTableCalendrier at: indice put:j.
j:=j+1.i:=i+1. indice := indice+1].
        [indice <= 42] whileTrue: [contenuTableCalendrier at: indice put: ". indice :=
indice+1].
    ].

```

terminer

```

" on met tous les Rdv dans le fichier bos "
stream := fichier writeStream.
bos := BinaryObjectStorage onNew: stream.
[bos nextPut: utilisateur listeRdv ] valueNowOrOnUnwindDo: [bos close].
stream close.
self closeRequest.
parent initialize:utilisateur.

```

valider

```

" on ajoute le rdv dans la liste de rdv si on peut "
utilisateur nouveauRdv: champJour value and: champRdvMois value and:
champRdvAnnee value and: champHeure value and: champMinute value and: champLieu
value and: champIntitule value and: champDuree value and: listPersonnes.
champLieu value:".
champIntitule value:".

```

precedent

```

|a ms m1 indexMois nb indice j i d premierJour |
m1 := champMois value.
ms := annee_courante moisPrecedent: m1.
self champMois value: ms.
ms = 'December' ifTrue:[

```

```

    a :=champAnnee value -1.
    self champAnnee value: a.
    annee_courante := Annee new:a.
].
" on recupere le nombre de jours du mois suivant"
mois_courant mois:ms.
indexMois := annee_courante listeMois indexOf:ms.
nb := annee_courante nbJoursMois at: indexMois.
mois_courant nbJours: nb.
" on cree la date "
d := Date newDay:1 month: mois_courant mois year:annee_courante annee.
" on recupere le numero du jour "
premierJour := d weekdayIndex.
" on va mettre a jour la table "
indice := 1.
[indice < premierJour] whileTrue: [contenuTableCalendrier at: indice put: ". indice :=
indice+1].
j:= 1.
i := 1.
[i <= mois_courant nbJours] whileTrue: [contenuTableCalendrier at: indice put:j.
j:=j+1.i:=i+1. indice := indice+1].
[indice <= 42] whileTrue: [contenuTableCalendrier at: indice put: ". indice := indice+1].

```

Nouvel_utilisateur

initialize: a

```

fichier := 'utilisateur.bos' asFilename.
" on recupere l'admin qui vient de se connecter "
admin := a.
listFichier := List new.
nbUtilisateur := admin listeUtilisateurs size.
self creerTable.
self remplirTable.
self initSightings.
self metLabel.
self open.

```

creerTable

```

" on crée une TwoDlist qui contient le contenu de la table "
contenu := TwoDList columns:4 rows: nbUtilisateur.
self lesUtilisateurs table: contenu.

```

initSightings

```

sightingsTable := SelectionInTable with: contenu.
lesUtilisateurs := TableInterface new selectionInTable: sightingsTable.

```

metLabel

```

lesUtilisateurs columnLabelsArray: #('Nom' 'Prenom' 'Login' 'Pass');
rowLabelsWidth: 20.

```

remplirTable

```
| indice i utilisateurCourant |
indice := 1.
i := 1.
[(indice <= nbUtilisateur)] whileTrue: [
    utilisateurCourant := admin listeUtilisateurs at: indice.
    contenu at:i put: utilisateurCourant nom.
    i := i +1.
    contenu at:i put: utilisateurCourant prenom.
    i := i+1.
    contenu at: i put: utilisateurCourant login.
    i := i +1.
    contenu at:i put: utilisateurCourant password.
    i := i+1.
    indice := indice+1.
].
```

annuler

```
self closeRequest.
Menu_principal new: admin.
```

dbClick

```
| v ligne utilisateurSelect n p l pass |
" on recupere le numero de la ligne sur laquelle ona cliqué "
v := self sightingsTable selectionIndex.
l := v y.
ligne := (l-1)*4 +1.
" on recupere les donnée de l'utilisateur "
n := contenu at: ligne.
ligne := ligne+1.
p := contenu at: ligne.
ligne := ligne+1.
l := contenu at:ligne.
ligne := ligne+1.
pass := contenu at: ligne.
utilisateurSelect := Utilisateur new:n and: p and: l and: pass.
" on affiche les données de l'utilisateur sélectionné "
champNom value: utilisateurSelect nom.
champPrenom value: utilisateurSelect prenom.
champLogin value: utilisateurSelect login.
champPass value: utilisateurSelect password.
```

fin

```
" on enregistre dans le fichier bos "
| indice |
stream := fichier writeStream.
bos := BinaryObjectStorage onNew: stream.
bos nextPut: admin listeUtilisateurs.
bos close.
```

```

stream close.
" on supprime les fichiers des utilisateurs qu'on a supprimé "
indice := 1.
[(indice <= listFichier size)] whileTrue:[
    (listFichier at: indice) delete.
    indice := indice +1.
].
self closeRequest.
Menu_principal new:admin.

```

modifier

```

| v ligne utilisateurSelect nouveauUtilisateur indice |
    v := self sightingsTable selectionIndex.
ligne := v y.
(ligne = 0) ifFalse: [
    " on va modifier l'utilisateur sélectionné "
    utilisateurSelect := admin listeUtilisateurs at: ligne.
    nouveauUtilisateur := Utilisateur new: champNom value and: champPrenom value
and: champLogin value and: champPass value.
    admin modifierUtilisateur: utilisateurSelect and: nouveauUtilisateur.
    " on va mettre a jour l'affichage de la table "
    indice := 1 + ((ligne-1)*4).
    contenu at: indice put:(champNom value).
    indice := indice +1.
    contenu at: indice put: (champPrenom value).
    indice := indice +1.
    contenu at: indice put: (champLogin value).
    indice := indice +1.
    contenu at: indice put:(champPass value).
].

```

supprimer

```

"on verifie qu'une ligne est sélectionnée et on la supprime "
| v l u f |
v := self sightingsTable selectionIndex.
l := v y.
u := admin listeUtilisateurs at: l.
" on efface l'utilisateur "
admin effacerUtilisateur: u.
nbUtilisateur := nbUtilisateur - 1.
self creerTable.
self remplirTable.
(Dialog confirm:'Voulez vous supprimer ses rdv?') ifTrue:[
    f := 'RDV\' ,u nom, '.bos'.
    f := f asFilename.
    " si le fichier existe "
    (f exists) ifTrue:[
        listFichier add: f.
    ].
].

```

].

valider

```
" on verifie que tous les champs ne sont pas vides "  
  ((champNom value = "") | (champPrenom value = "") | (champLogin value = "") |  
(champPass value = "")) ifFalse: [  
  " on recupere les infos entrées dans les champ texte "  
  admin creerUtilisateur: champNom value and: champPrenom value and: champLogin  
value and: champPass value.  
  nbUtilisateur := nbUtilisateur +1.  
  "contenu := TwoDList columns:4 rows: nbUtilisateur."  
  "self lesUtilisateurs table: contenu."  
  self creerTable.  
  self remplirTable.  
  " on remet a vide les champs "  
  champNom value:".  
  champPrenom value:".  
  champLogin value:".  
  champPass value:".  
  ]  
ifTrue: [  
  Dialog warn:' un des champs est vide '  
  ].
```

NewRdvPersonne

Le code est identique a la classe MesInterfaces car cette classe permet l'admin de pouvoir ajouter un rdv a un utilisateur, la seule chose qui change est qu'il y a une liste deroulante pour choisir l'utilisateur

Modif_password

initialize:p

```
parent := p.  
utilisateur := p personne.  
self open.  
champLogin value: utilisateur login.  
champPass value:utilisateur password.
```

annuler

```
self closeRequest.  
parent initialize:utilisateur.
```

valider

```
| fichier st bos listeUtilisateurs indice u |  
"on va modifier le fichier bos contenant les utilisateurs "  
" on recupere la liste des utilisateurs "  
fichier := 'utilisateur.bos' asFilename.  
st := fichier readStream.  
bos := BinaryObjectStorage onOld:st.
```

```

listeUtilisateurs := bos next.
bos close.
st close.
indice := 1.
[(indice <= listeUtilisateurs size)] whileTrue:[
    u := listeUtilisateurs at: indice.
    " on cherche l'utilisateur qui est enregistré"
    (((u nom) = (utilisateur nom)) & ((u prenom) = (utilisateur prenom))) ifTrue:[
        u login: champLogin value.
        u password: champPass value.
    ].
    indice := indice +1.
].
" on remet la liste modifiée dans le fichier bos "
st := fichier writeStream.
bos := BinaryObjectStorage onNew:st.
bos nextPut: listeUtilisateurs.
bos close.
st close.
self closeRequest.
parent initialize: u.

```

Nouveau_contact

initialize:p

```

parent := p.
utilisateur := p personne.
utilisateur mesContacts: List new.
chemin := 'Contacts\'.
chemin := chemin,utilisateur nom,'_'.
chemin := chemin,utilisateur prenom.
chemin := chemin,'.bos'.
fichier := chemin asFilename.
" si le fichier existe on recupere la liste de contact "
(fichier exists) ifTrue:[
    self recupListe.
]
ifFalse:[
    nbContact := 0.
].
self creerTableContacts.
self remplirTableContacts.
self metLabelTableContacts.
self open.

```

creerTableContacts

```

contenuTableContact := TwoDList columns:2 rows: nbContact.
self listContact table: contenuTableContact.

```

metLabelTableContacts

listContact columnLabelsArray:#('Nom' 'Prenom').

recupListe

```
st := chemin readStream.  
b := BinaryObjectStorage onOld:st.  
utilisateur mesContacts:b next.  
b close.  
st close.  
nbContact := utilisateur mesContacts size.
```

remplirTableContacts

```
| indice i personne |  
indice := 1.  
i := 1.  
[(indice <= nbContact)] whileTrue:[  
    personne := utilisateur mesContacts at:indice.  
    contenuTableContact at: i put: personne nom.  
    i := i+1.  
    contenuTableContact at: i put: personne prenom.  
    i := i+1.  
    indice := indice +1.  
].
```

Actions

ajouter

```
| personne i pasTrouve p |  
((champNom value = "") | (champPrenom value = "")) ifTrue:[  
    Dialog warn: 'Il faut remplir tous les champs '  
]  
ifFalse:[  
    " on va verifier que la personne que l'on va ajouter n'est pas presente dans la liste "  
    i := 1.  
    pasTrouve := true.  
    [((i <= nbContact) & (pasTrouve))] whileTrue:[  
        p := utilisateur mesContacts at: i.  
        ((p nom = champNom value) & ( p prenom = champPrenom value)) ifTrue:[  
            pasTrouve := false.  
        ].  
        i := i +1.  
    ].  
    " si la personne n'est pas dans la liste on l'ajoute "  
    (pasTrouve) ifTrue:[  
        personne := Personne new: champNom value and: champPrenom value.  
        nbContact := nbContact +1.  
        utilisateur mesContacts add: personne.  
        self creerTableContacts.  
        self remplirTableContacts.  
    ]
```

```
ifFalse:[
    Dialog warn: 'Il y a deja une personne dans votre liste avec le meme nom'.
].
" on remet a vide les champs "
champNom value:".
champPrenom value:".
].
```

terminer

```
" on va enregistrer la liste dans le fichier bos"
| st bos |
st := fichier writeStream.
bos := BinaryObjectStorage onNew: st.
[bos nextPut: utilisateur mesContacts      ] valueNowOrOnUnwindDo: [bos close].
"bos nextPut:utilisateur mesContacts.
bos close."
st close.
self closeRequest.
parent initialize: utilisateur.
```