

Contrôle DE TP SIO:

## **PARTIE 2:**

A2:

*"Création des groupes SIO"*

*AllGroups add: 'SIO'.*

*"Pour les groupes 1 à 4, binômes 1 à 6"*

*1 to: 4 do: [ :g |*

*1 to: 6 do: [ :each | (AllUsers addNewUserWithId: ('sio', g printString, each printString) password:*

*'PolytechSIO') addGroup: 'SIO'; addPrivilege: #UserPassword; insertDictionary: GBWebDemo at: 1 ]].*

*"Suppression"*

*1 to: 4 do: [ :g |*

*1 to: 7 do: [ :each | AllUsers remove: (AllUsers userWithId: ('sio', g printString, each printString) )]*

*]*

### **Que contient cette page?**

Elle contient un code smalltalk Base qui veut ajouter un groupe a la base de donnée.

### **Comment utiliser ce qu'elle contient ?**

Pour l'utiliser il faut sélectionner le code à exécuter et cliquer sur l'icône « Do it ».

Pour utiliser ce quelle contient il faut se connecter à la base ce qui permettra d'insérer les groupes

### **Quel serait le résultat ?**

ajout des groupes 1 à 4 et des binômes 1 à 6 sous le login Sio[1-4][1-6] et le password PolytechSIO dans le groupe SIO et avec les privilèges changement de mot de passe et accès au dictionnaire GBWebDemo

A3:

### **Exécutez les instructions de la page « Preparation des TPs » du Worspace étudié précédemment. Que se passe-t-il ?**

On a une exception, ce qui est normal

### **Pourquoi ?**

Il faut être forcément connecté à la base.

## **PARTIE 3:**

### **Exécutez les instructions de la page « Preparation des TPs » du Worspace étudié précédemment. Que se passe-t-il ? Pourquoi ?**

on ne peut toujours pas exécuter le code car nous n'avons pas les privilèges (déclenchement d'une exception).

## **PARTIE 4:**

### **A**

**Déterminez quelle méthode renvoie la liste des voitures (instances de GBWCar) stockées dans la base.**

Pour déterminer la méthode :

Cat: GBWebDemo,->Classe:GBWCar->Prot:class variable accessing,

méthode:allCars() à ne pas confondre avec *allCars*: )

Cette méthode renvoie une instance de allclass, la variable all class dans GBWCar

**Où cette méthode récupère-t-elle cette liste ?**

Elle récupère cette liste dans la variable de classe AllCars

**Réalisez les exercices proposés dans la page « Exercices proxy, replicate et forwarder » du Workspace. Vous aurez besoin de consulter les Connecteurs définis pour votre session. Vous utiliserez pour ceci le « Connector Browser »<sup>1</sup>. Vous utiliserez aussi des fenêtres « inspecteur d'objet » :**

1-allCarsProxy := GBSM execute: 'GBWCar allCars'

*print it*

a GbxL4Delegate sess: 2 flags: 0 {16r2A6E1}

2-allCarsProxy

*inspect it*



résultat: un GbxDelegate représentant un objet stocké dans la base GemStone/S(onglet commençant par GS et de couleur mauve)

### **Interprétation:**

Les deux lignes de commande s'exécute ( par print it ou par inspect it) a travers un Delegate.

**Dans les variables du Workspace.**

allCarsProxy :

3-allCarsProxy select: [ :each | each price < 10000 ]

---

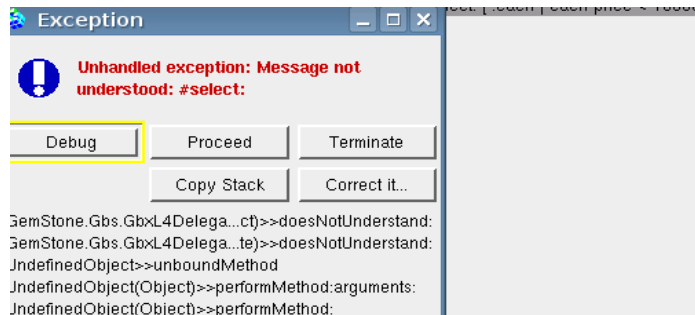
<sup>1</sup> Icône représentant une prise dans la fenêtre GemStone. Ou commande « Browse Connectors » du menu GemStone de la fenêtre Visualworks.

print it

### Interprétation :

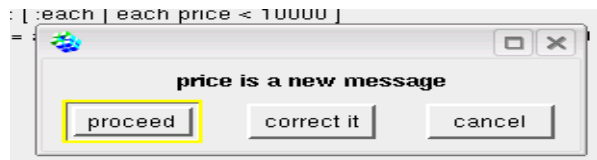
Le Delegate ne permet pas d'envoyer un message a un objet de base.

C'est un intermédiaire opaque pour l'envoi de message a l'objet. => generation d'exception

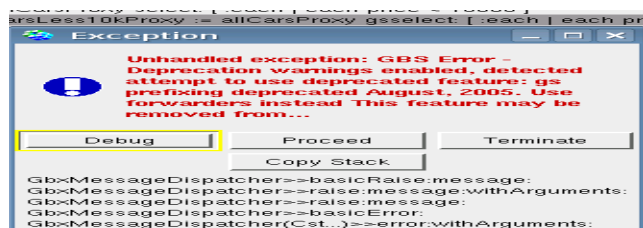


4-carsLess10kProxy := allCarsProxy gsselect: [ :each | each price < 10000 ]

"print it"



puis après 2 "Proceed" :



On fait un troisième **procced**

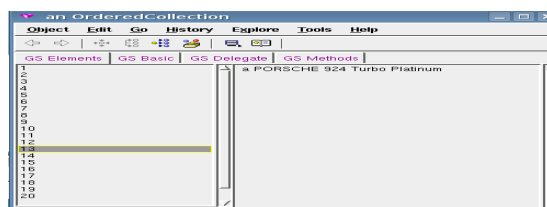
et ça marche. Il initialise notre variable :

a GbxL4Delegate sess: 2 flags: 0 {16r324C5}

5-carsLess10kProxy := allCarsProxy gsselect: [ :each | each price < 10000 ]

a GbxL4Delegate sess: 2 flags: 0 {16r8F805}

6-carsLess10kProxy



Décrivez le résultat d'une exécution par "inspect it" de la ligne ci-dessus :

### Interprétation :

il renvoi une collection de voiture

```
7-carsLess10kProxyParRemotePerform := allCarsProxy remotePerform: #select: with: [ :each | each price < 10000 ]
```

### **"print it"**

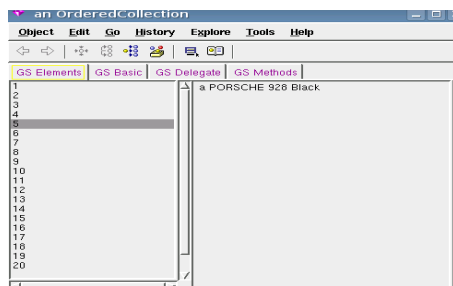


après proceed

*a GbxL4Delegate sess: 2 flags: 0 {16r8F98D}*

```
8-carsLess10kProxyParRemotePerform
```

### **"inspect it"**



### Interprétation :

on a affaire a un objet GbxDelegate (onglet commençant par GS et de couleur mauve)

```
9-carsLess10kProxy = carsLess10kProxyParRemotePerform
```

Copiez le résultat de l'exécution par "print it" de la ligne ci-dessus :

false

### Interprétation :

contenu égal mais pas le même objet (remoteperform fait un nouvel objet chaque requête)

```
10-firstCarLess10kProxy := carsLess10kProxy gsat: 1
```

### **"print it"**

*a GbxL4Delegate sess: 2 flags: 0 {16r2A659}*

```
11-firstCarLess10kProxy2 := carsLess10kProxyParRemotePerform gsat: 1
```

### **"print it"**

*a GbxL4Delegate sess: 2 flags: 0 {16r2A659}*

```
12-firstCarLess10kProxy = firstCarLess10kProxy2
```

### **"print it"**

true

### Interprétation :

il s'agit bien du même objet

```
13-allCarsForwarder := (GBSM execute: 'GBWCar allCars') asForwarder
```

### **"print it" :**

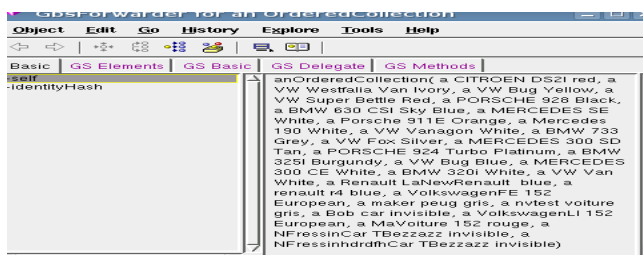
*anOrderedCollectionNothing more expected -> ( a CITROEN DS2I red, a VW Westfalia Van Ivory, a VW Bug Yellow, a VW Super Bettle Red, a PORSCHE 928 Black, a BMW 630 CSI Sky Blue, a MERCEDES SE White, a Porsche 911E Orange, a Mercedes 190 White, a VW Vanagon White, a BMW 733 Grey, a VW Fox Silver, a MERCEDES 300 SD Tan, a PORSCHE 924 Turbo Platinum, a BMW 325I Burgundy, a VW Bug Blue, a MERCEDES 300 CE White, a BMW 320i White, a VW Van White, a Renault LaNewRenault blue, a renault r4 blue, a VolkswagenFE 152 European, a maker peug gris, a nvtest voiture gris, a Bob car invisible, a VolkswagenLI 152 European, a MaVoiture 152 rouge, a NFressinCar TBezzazz invisible, a NFressinhrdrfhCar TBezzazz invisible)*

### **Commentaire :**

le forwarder donne la réponse comme si c'était l'objet

#### 14-allCarsForwarder

Décrivez le résultat d'une exécution par "inspect it" de la ligne ci-dessus :



Dans les variables du Workspace :

-----  
allCarsForwarder:

allCarsProxy:  
-----

### **Interprétation :**

on a enfin accès aux informations (grâce au passage par les forwarders)

#### 15-allCarsProxy = allCarsForwarder asGSObject

Copiez le résultat de l'exécution par "print it" de la ligne ci-dessus :

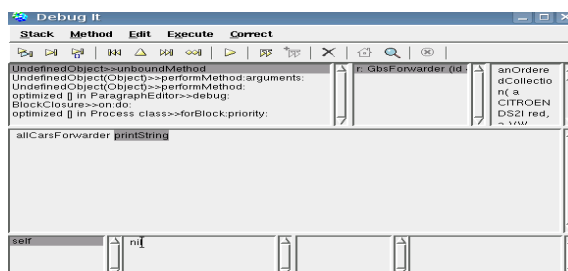
true

### **Interprétation :**

ils s'agit bien du même objet

#### 16-allCarsForwarder printString

Expliquez ce que montre une exécution pas à pas par "debug it" sur la ligne ci-dessus :



on bascule tout de suite dans l'environnement de GemStone  
on entre dans les différentes erreurs , on les exécute un a un .

Copiez ici la dernière ligne qui figure dans votre System Transcript (Fenetre VisualWorks):  
09-03-24 12:05:25:031 Logged in Session 2 (remote) for 'sio11' on '!@weppes!gemserver61'  
Donnez la liste des connecteurs de votre session :  
pas de connecteurs

```

17-carsLess10kReplicate := allCarsForwarder select: [ :each | each price < 10000 ]
"print it"
OrderedCollection (a GBWCar a GBWCar a GBWCar a GBWCar a GBWCar a GBWCar a
GBWCar a GBWCar a GBWCar a GBWCar a GBWCar a GBWCar a GBWCar a GBWCar a
GBWCar a GBWCar a GBWCar a GBWCar a GBWCar a GBWCar a GBWCar a GBWCar)

```

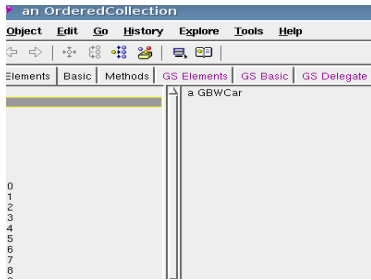
Copiez ici les deux dernières lignes qui figurent dans le System Transcript :

09-03-24 13:23:56:187 Creating Smalltalk class GBWCar.  
09-03-24 13:23:56:265 Creating Smalltalk class GBWData.

Donnez la liste des connecteurs de votre session :  
*GBWCar et GBWData*

18-carsLess10kReplicate

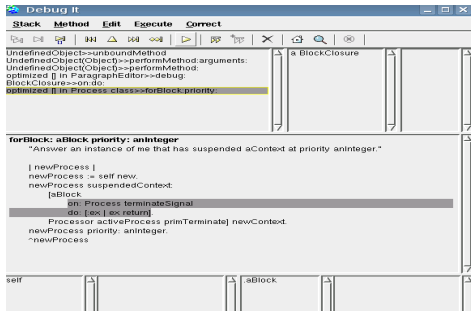
Décrivez le résultat d'une exécution par "inspect it" de la ligne ci-dessus :



Interprétation : on inspecte un objet replicate stocké dans la base Gemstone/S

*18-carsLess10kReplicate* printString.

Expliquez ce que montre une exécution pas à pas par "debug it" sur la ligne ci-dessus :



on ne bascule pas dans l'environnement GS car on a un replicat

19-carsLess10kForwarder := allCarsForwarder fwselect: [ :each | each price < 10000 ]

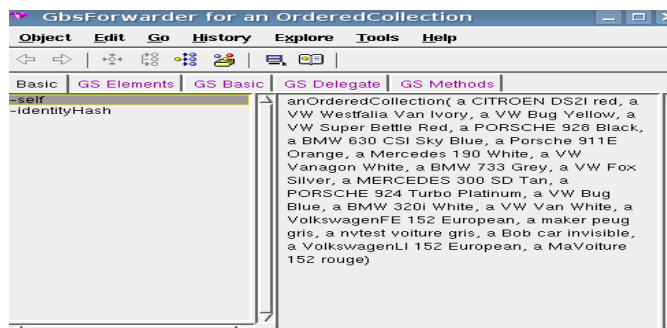
Copiez le résultat de l'exécution par "print it" de la ligne ci-dessus :

*anOrderedCollection( a CITROEN DS2I red, a VW Westfalia Van Ivory, a VW Bug Yellow, a VW Super Bettle Red, a PORSCHE 928 Black, a BMW 630 CSI Sky Blue, a Porsche 911E Orange, a Mercedes 190 White, a VW Vanagon White, a BMW 733 Grey, a VW Fox Silver, a MERCEDES 300 SD Tan, a PORSCHE 924 Turbo Platinum, a VW Bug Blue, a BMW 320i White, a VW Van White, a VolkswagenFE 152 European, a maker peug gris, a nvtest voiture gris, a Bob car invisible, a VolkswagenLI 152 European, a MaVoiture 152 rouge)*

20-carsLess10kForwarder

Décrivez le résultat d'une exécution par "inspect it" de la ligne ci-dessus :

onglet mauve commençant par GS et un onglet noir "basic"



Interprétation : GbsForwarder

21-allCarsReplicate := GBSM evaluate: 'GBWCar allCars'

Copiez le résultat de l'exécution par "print it" de la ligne ci-dessus :

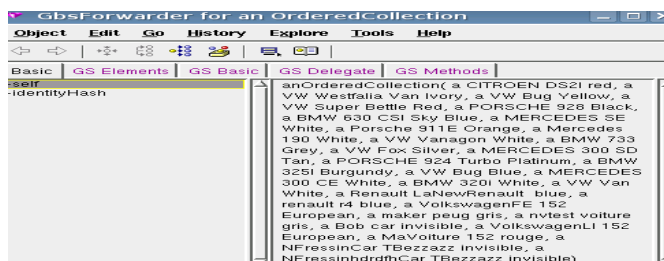
*anOrderedCollection( a CITROEN DS2I red, a VW Westfalia Van Ivory, a VW Bug Yellow, a VW Super Bettle Red, a PORSCHE 928 Black, a BMW 630 CSI Sky Blue, a MERCEDES SE White, a Porsche 911E Orange, a Mercedes 190 White, a VW Vanagon White, a BMW 733 Grey, a VW Fox Silver, a MERCEDES 300 SD Tan, a PORSCHE 924 Turbo Platinum, a BMW 325I Burgundy, a VW Bug Blue, a MERCEDES 300 CE White, a BMW 320i White, a VW Van White, a Renault LaNewRenault blue, a renault r4 blue, a VolkswagenFE 152 European, a maker peug gris, a nvtest voiture gris, a Bob car invisible, a VolkswagenLI 152 European, a MaVoiture 152 rouge, a NFressinCar TBezzazz invisible, a NFressinhdrdthCar TBezzazz invisible)*

Dans les variables du Workspace :

-----  
allCarsReplicate :  
-----

22-allCarsReplicate

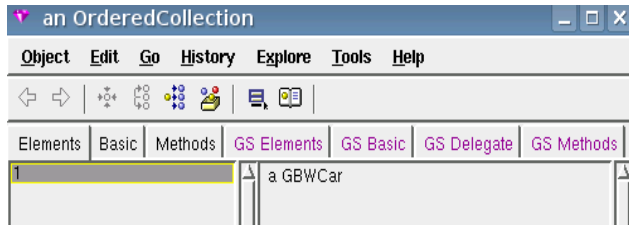
Décrivez le résultat d'une exécution par "inspect it" de la ligne ci-dessus : onglet mauve commençant par GS et un onglet noir "basic"



## **Interprétation :** GbsForwarder

22-redCarsReplicate := GBSM evaluate: 'GBWCar allCars select: [:each | each color = "Red" ]'

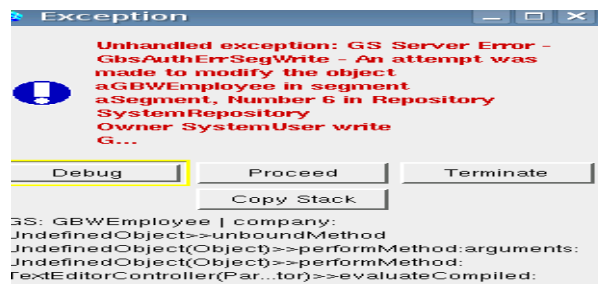
Décrivez le résultat d'une exécution par "inspect it" de la ligne ci-dessus : fenetre contenant tous les onglets noirs et mauves => on inspecte un objet replicate stocke dans la base Gemstone/S



### **B) Manipulation des classes et objets en GemStone Smalltalk**

- ❖ Au moyen du « GemStone Classes Browser »<sup>2</sup>, explorez le dictionnaire GBWebDemo et déterminez :
  - ❖ quelle méthode renvoie la liste des employés (instances de GBWEmployee) stockés dans la base  
*allGBWEmployees*
  - et où cette méthode récupère cette liste  
*dans la variable de classe allGBWEmployees*
  - ❖ quelle méthode permet de récupérer le nom d'un employé ;  
*name (méthode d'instance)*
  - ❖ quelle méthode permet de modifier la compagnie d'un employé.  
*company: newValue (méthode d'instance)*
- ❖ Ecrivez et exécutez dans le workspace un code GemStone Smalltalk qui modifie le nom de la compagnie de l'employé nommé Denny Bollay.

```
allEmployeesProxy := (GBSM execute: 'GBWEmployee allGBWEmployees' )  
asForwarder  
employeeDennyBollay := allEmployeesProxy fwdetect: [:each | each name='Denny  
Bollay']  
employeeDennyBollay company: 'abdoulaye lolo'
```





Que se passe-t-il ?  
exception

Pourquoi ?  
Problème de droit d écriture

Que faut-il faire (ou faire faire) pour changer cette situation pour votre compte uniquement ?  
changer les droits pour notre compte (donc par l'admin) -> nous ajouter au groupe Publishers

❖ **Avant de continuer, demandez aux encadrants d'effectuer l'action qui changera cette situation pour votre compte.**

**Faire valider la modification dans la base de donnée par un COMMIT**

❖ Ecrivez un script GemStone Smalltalk qui crée un employé vous représentant dans la liste des employés stockée dans GemStone.

|diaby|

diaby := GBWEmployee name:'diaby' company:'citybank' city:'london' state:'uk' zip:'59000'  
phone:'0679693612'.

GBWEmployee addEmployee:diaby

employeeDiaby:= allEmployeesProxy fwdetect: [:each | each name='diaby']

Quelles difficultés vous et/ou vos collègues risquez-vous de rencontrer si vous souhaitez que les employés vous représentant soient ajoutés durablement dans la liste des employés ?

Cette modification est liée à notre session (nous sommes donc dans une transaction et il faudrait donc faire un commit et que tout le monde accepte d'annuler ses propres modifications)

C-Création d'une application Smalltalk accédant aux objets GemStone Smalltalk de l'application GBWebDemo (45 min)

-Créez dans votre image Smalltalk et connectez une copie synchronisée (un « replicate ») de la liste des GBWCar existante dans GemStone.

a- Pour ceci, répondez d'abord aux questions suivantes : quels types d'objets et de variables devez-vous relier ?

La liste des cars est représentée par une OrderedCollection stockée dans la variable de classe GBWCar allCars

b- Quel type de connecteur pouvez-vous utiliser ?

Les connecteurs sont GBWCar et GBWData (subClass ou sur classe). On doit donc relier un objet Ordered Collection (liste des Cars) à une variable de classe (GBWCar allCars) par un connecteur de variables de classe.

-Une fois le connecteur créé, inspectez les objets qu'il connecte et interprétez le résultat.

-Reproduisez le protocole de la classe GemStone Smalltalk GBWCar dans la classe Smalltalk GBWCar (cf).

**Dans un « GemStone Classes Browser », sélectionner la classe GBWCar, ouvrir le menu contextuel (maintenir enfoncé le bouton droit de la souris) et sélectionner l'item « compile in st »**

On doit donc sélectionner dans le Gemstone Class Browser (Gemstone->Browse->AllClasses) la classe GBWebdemo->GBWCar et par un clic droit on exécute la commande compile in ST.

On peut vérifier dans le System Browser de Smalltalk que les méthodes ont été reproduites.

- **Observez la répercussion de la modification de la couleur de la première voiture de la collection dans GemStone (resp. Smalltalk) sur l'objet correspondant dans Smalltalk (resp. GemStone)**

Dans la fenêtre VisualWorks ->Gemstone->Browses Connector->Sessions Connectors Sio11 ->Class-> UpdateST. On a la création de Creating Smalltalk class GBWCar

On est en Smalltalk updateST. Tous les changements effectués sur Gemstone seront répercutés sur Smalltalk mais à l'inverse, on n'aura pas de mise à jour de Gemstone en cas de modification sur Smalltalk.

- **interprétez.**

La synchronisation est automatique dans le sens Gemstone vers Smalltalk mais il faut choisir marquer les objets modifiés dans le cas inverse, avant la modification, par la commande:

GBWCar makeGSTransparent

## ANNEXES:

### PARTIE 4-A)

allCarsProxy := GBSM execute: 'GBWCar allCars' a GbxL4Delegate sess: 2 flags: 0 {16r2A6E1}  
allCarsProxy

allCarsProxy select: [ :each | each price < 10000 ]

carsLess10kProxy := allCarsProxy gsselect: [ :each | each price < 10000 ] a GbxL4Delegate sess: 2 flags: 0 {16r324C5}

carsLess10kProxy

carsLess10kProxyParRemotePerform := allCarsProxy remotePerform: #select: with: [ :each | each price < 10000 ] a GbxL4Delegate sess: 2 flags: 0 {16r2C585}

carsLess10kProxyParRemotePerform

carsLess10kProxyParRemotePerform := allCarsProxy remotePerform: #select: with: [ :each | each price < 10000 ] a GbxL4Delegate sess: 2 flags: 0 {16r3FA49}

carsLess10kProxyParRemotePerform

carsLess10kProxy = carsLess10kProxyParRemotePerform false

carsLess10kProxy = carsLess10kProxyParRemotePerform false

firstCarLess10kProxy := carsLess10kProxy gsat: 1 a GbxL4Delegate sess: 2 flags: 0 {16r2A659}

firstCarLess10kProxy2 := carsLess10kProxyParRemotePerform gsat: 1 a GbxL4Delegate sess: 2 flags: 0 {16r2A659}

firstCarLess10kProxy = firstCarLess10kProxy2 true

allCarsForwarder := (GBSM execute: 'GBWCar allCars') asForwarder anOrderedCollection( a CITROEN DS2I red, a VW Westfalia Van Ivory, a VW Bug Yellow, a VW Super Bettle Red, a PORSCHE 928 Black, a BMW 630 CSI Sky Blue, a MERCEDES SE White, a Porsche 911E Orange, a Mercedes 190 White, a VW Vanagon White, a BMW 733 Grey, a VW Fox Silver, a MERCEDES 300 SD Tan, a PORSCHE 924 Turbo Platinum, a BMW 325I Burgundy, a VW Bug Blue, a MERCEDES 300 CE White, a BMW 320i White, a VW Van White, a Renault LaNewRenault blue, a renault r4 blue, a VolkswagenFE 152 European, a maker peug gris, a nvtest voiture gris, a Bob car invisible, a VolkswagenLI 152 European, a MaVoiture 152 rouge, a NFressinCar TBezzazz invisible, a NFressinhdrdfhCar TBezzazz invisible)

allCarsProxy

allCarsProxy select: [ :each | each price < 10000 ]

carsLess10kProxy := allCarsProxy gsselect: [ :each | each price < 10000 ] a GbxL4Delegate sess: 2 flags: 0 {16r3EE35}

carsLess10kProxy a GbxL4Delegate sess: 2 flags: 0 {16r3EE35}

carsLess10kProxyParRemotePerform := allCarsProxy remotePerform: #select: with: [ :each | each price < 10000 ] a GbxL4Delegate sess: 2 flags: 0 {16r3EF21}

carsLess10kProxyParRemotePerform

carsLess10kProxy = carsLess10kProxyParRemotePerform false

carsLess10kReplicate := allCarsForwarder select: [ :each | each price < 10000 ] OrderedCollection (a GBWCar a GBWCar)

carsLess10kReplicate

carsLess10kReplicate printString

carsLess10kForwarder := allCarsForwarder fwselect: [ :each | each price < 10000 ]

anOrderedCollection( a CITROEN DS2I red, a VW Westfalia Van Ivory, a VW Bug Yellow, a VW Super Bettle Red, a PORSCHE 928 Black, a BMW 630 CSI Sky Blue, a Porsche 911E Orange, a Mercedes 190 White, a VW Vanagon White, a BMW 733 Grey, a VW Fox Silver, a MERCEDES 300 SD Tan, a PORSCHE 924 Turbo Platinum, a VW Bug Blue, a BMW 320i White, a VW Van White, a VolkswagenFE 152 European, a maker peug gris, a nvtest voiture gris, a Bob car invisible, a VolkswagenLI 152 European, a MaVoiture 152 rouge)

carsLess10kForwarder

allCarsReplicate := GBSM evaluate: 'GBWCar allCars'

anOrderedCollection( a CITROEN DS2I red, a VW Westfalia Van Ivory, a VW Bug Yellow, a VW Super Bettle Red, a PORSCHE 928 Black, a BMW 630 CSI Sky Blue, a MERCEDES SE White, a Porsche 911E Orange, a Mercedes 190 White, a VW Vanagon White, a BMW 733 Grey, a VW Fox Silver, a MERCEDES 300 SD Tan, a PORSCHE 924 Turbo Platinum, a BMW 325I Burgundy, a VW Bug Blue, a MERCEDES 300 CE White, a BMW 320i White, a VW Van White, a Renault LaNewRenault blue, a renault r4 blue, a VolkswagenFE 152 European, a maker peug gris, a nvtest voiture gris, a Bob car invisible, a VolkswagenLI 152 European, a MaVoiture 152 rouge, a NFressinCar TBezzazz invisible, a NFressinhdrdfhCar TBezzazz invisible)

allCarsReplicate

redCarsReplicate := GBSM evaluate: 'GBWCar allCars select: [ :each | each color = "Red" ]'

#### **PARTIE 4-B)**

allEmployeesProxy := GBSM execute: 'GBWEmployee allGBWEmployees'

a GbxL4Delegate sess: 2 flags: 0 {16r3EF45}

employeeDennyBollay := allEmployeesProxy gsdetect: [:each | each name='Denny Bollay']

a GbxL4Delegate sess: 2 flags: 0 {16r3EF41}

cgCompName := employeeDennyBollay gscompany: 'mrbob'

AllUsers userWithId: ('sio17', Argument expected ->addGroup: 'Publishers')

|diaby|

diaby := GBWEmployee name:'diaby' company:'citybank' city:'london' state:'uk' zip:'59000'

phone:'0679693612'.

GBWEmployee addEmployee:diaby

employeeDiaby Nothing more expected ->:= allEmployeesProxy gsdetect: [:each | each name='diaby']