TP-1 : Premiers pas en Smalltalk

Noury Bouraqadi

1 Présentation

L'objectif de ce TP est de vous familiariser avec l'environnement de programmation de Smalltalk tout en révisant les concepts d'objet et de classe. Cette familiarisation se fera à l'aide du Smalltalk libre Pharo.

2 Lampe

Dans la catégorie **Trafic**, définissez la classe **Lampe** qui dispose de deux variables d'instances : **estAllumee** et **couleur**. La variable d'instance **estAllumee** référence un booléen qui est **true** quand la lampe est allumée et **false** quand la lampe est éteinte. Quant à la variable d'instance **couleur**, elle référence une instance de la classe **Color** (disponible dans la bibliothèque de base de Pharo). Des instances prédéfinies de la classe **Color** peuvent être obtenues à l'aide des expressions suivantes :

- **Color red**
- **Color green**
- **Color blue**
- **Color yellow**
- **Color gray**

Dotez la classe **Lampe** de la catégorie **etat** et définissez y les méthodes suivantes :

- **estAllumee** retourne le booléen contenu dans la variable d'instance de même nom.
- **estAllumee: unBooleen** affecte à la variable d'instances **estAllumee** le booléen référencé par l'argument unBooleen.
- **allumer** allume la lampe, ce qui revient à modifier la valeur de la variable d'instance**s estAllumee.** Utilisez pour ce faire la méthode **estAllumee:.**
- **eteindre** éteint la lampe. Opère de manière symétrique à la méthode **allumer.**

Dotez la classe **Lampe** de la catégorie **apparence** et définissez y les deux méthodes suivantes :

- **couleur: uneCouleur** modifie la couleur de la lampe pour utiliser la couleur correspondant au paramètre **uneCouleur.**
- **couleur** retourne la couleur courante de la lampe (i.e. le contenu du champ de même nom) si la lampe est allumée ou la couleur grise si la lampe est éteinte.

Nous souhaitons que toute nouvelle lampe soit par défaut jaune et éteinte. Pour ce faire, dotez la classe **Lampe** de la catégorie **initialisation** et définissez y les méthodes suivantes :

- **couleurParDefaut** retourne la couleur jaune.
- **estAllumeeParDefaut** retourne faux (**false**) pour indiquer que la lampe doit être initialement eteinte.

- **initialize** pour initialiser toute nouvelle lampe avec les valeur par défaut définies dans les deux méthodes précédentes.

Dans un **Workspace**, testez le code que vous venez de développer. Pour ce faire, créez des instances de la classe **Lampe** et envoyez leur des messages. Vérifier le bon fonctionnement à l'aide d'un inspecteur (menu « **inspect it** »).

3 Sauvegarde et Import/Export de code

Sauvegardez votre image, puis quittez la. Relancez la et vérifiez que votre environnement est inchangé et que votre code a été sauvegardé.

Sélectionnez la catégorie **Trafic** de votre TP et exportez la à l'aide du menu **fileOut**. Ouvrez le fichier **.st** obtenu dans le même dossier que l'image à l'aide d'un éditeur de texte (Notepad par exemple). Vérifiez que la totalité de votre code y est sauvegardée. Ouvrez une image Pharo vierge (qui ne contient pas votre TP) et à l'aide de la souris, glissez le fichier **.st** sauvegardé dans Pharo. Faites l'opération 3 fois successivement (3 glisser / déposer dans des *images vierges*) et choisir : « **changelist Browser »**, « **code-file Browser »** et « **fileIn entire file »**. Analysez le fonctionnement de chacun de ces menu et comment arriver à charger le code (entre deux glisser/déposer quitter l'image sans sauvegarder).

- **changelist Browser** : Affiche les différentes parties du code (définitions de classes ou de méthodes) sous forme d'une liste. On peut en sélectionner quelques lignes et les charger dans l'image à l'aide du bouton « **file in selections** »
- **code-file Browser** : Affiche le code dans un browser de visualisation (impossible de modifier/sauvegarder). On peut sélectionner une catégorie, une classe ou une méthode et la charger en sélectionnant le menu « **file in** » obtenu avec un clic droit sous windows.
- **fileIn entire file** : charge directement le fichier.

4 Affichage textuel de la lampe

Dans la même catégorie que la classe **Lampe**, définissez la classe **AfficheurTextuelDeLampe**. Comme son nom l'indique, une instance de cette classe affiche de manière textuelle les informations relatives à une lampe. Cet affichage est réalisé sur la console accessible à l'aide de la variable globale **Transcript**. Notez que pour voir l'affichage, la fenêtre de la console (**Transcript**) doit être ouverte.

La classe **AfficheurTextuelDeLampe** doit être dotée d'une seule variable d'instances: **affichageAutorise**. Elle correspond à un booleen qui est à **true** quand les informations relatives à une lampe sont affichées.

Définir dans la catégorie **accesseurs** de la classe **AfficheurTextuelDeLampe** les méthodes **affichageAutorise** et **affichageAutorise:** qui permettent d'obtenir et de remplacer la valeur de la variable d'instance **affichageAutorise**.

Dans la catégorie **initialisation** de la classe **AfficheurTextuelDeLampe** ajouter les méthodes :

- **affichageAutoriseParDefaut** qui retourne true.
- **initialize** qui initialise la variable d'instance **affichageAutorise** avec la valeur par défaut. Utilisez pour cela les méthodes existantes.

Dotez la classe **AfficheurTextuelDeLampe** de la catégorie **affichage** qui contient les méthodes suivantes :

- **demarrerAffichage** affecte **true** à la variable d'instances **affichageAutorise** (utiliser les méthodes de la catégorie **accesseurs**).
- **arreterAffichage** affecte **false** à la variable d'instances **affichageAutorise** (utiliser les méthodes de la catégorie **accesseurs**).
- **miseAJourAvec: uneLampe**. Cette méthode effectue l'affichage de l'état (éteinte ou allumée) et la couleur de la lampe passée en paramètre. L'affichage consiste à écrire le text adéquat sur la console (accessible via la variable globale **Transcript**). Notez que :
    - l'affichage n'est effectué que si **affichageAutorise** a pour valeur **true** (utiliser les méthodes de la catégorie **accesseurs**).
    - le nom d'une couleur (*yellow, red, blue, green, ...*) s'obtient en envoyant à une instance de la classe **Color** le message **name**.

Notez que la console (**Transcript**) sait répondre aux messages :

- **cr** provoque un retour à la ligne (carrige return).
- **tab** affiche une tablulation
- **show: uneChaineDeCaracteres** affiche la chaîne de caractères passée en argument.

Modifiez la classe **Lampe** en la dotant d'une variable d'instance **afficheur**. Cette variable d'instance est destinée à référencer un objet affichage de lampe. Dans la catégorie **affichage** ajoutez les méthodes suivantes :

- **afficheur** retourne l'afficheur utilisé.
- **afficheur: unAfficheur** mémorise l'afficheur correspondant à l'argument **unAfficheur**.
- **mettreAJourAfficheur** opère la mise à jour de l'afficheur si la variable d'instances afficheur est initialisée, c'est à dire si la méthode afficheur retourne une valeur différente de nil. Notez que les messages **ifNil:** et **ifNotNil:** permettent d'exécuter de manière conditionnelle un bloc de code si le receveur est respectivement **nil** ou non. Exemples :
    - **nil ifNil: [Transcript cr ; show : 'Hello']** affiche **Hello** sur le **Transcript**.
    - **123 ifNil: [Transcript cr ; show : 'Hello']** n'affiche rien.
    - **nil ifNotNil: [Transcript cr ; show : 'GoodBye']** n'affiche rien.
    - **123 ifNotNil: [Transcript cr ; show : 'GoodBye']** affiche **GoodBye** sur le **Transcript**.
- **demarrerAffichage** ne fait rien s'il n'y a pas d'afficheur (variable d'instance **afficheur** à **nil**). Sinon, envoi le message **demarrerAffichage** à l'afficheur et le met à jour.
- **arreterAffichage** ne fait rien s'il n'y a pas d'afficheur (variable d'instance **afficheur** à **nil**). Sinon, envoi le message **arreterAffichage** à l'afficheur.

Pour finir cet exercice, identifiez les méthodes de la classe **Lampe** à modifier pour mettre à jour l'afficheur (envoi du message **mettreAJourAfficheur**) de manière à afficher les changements de l'état et de la couleur de la lampe.

Enfin, assurez-vous que le **Transcript** est ouvert puis testez votre code en évaluant des expressions dans un **Workspace**. Dans le tableau qui suit vous est fournie une série d'expressions évaluées dans le **Workspace** et l'affichage correspondant dans le **Transcript**.

Code évalué dans le workspace	*Affichage sur le Transcript*
lampe := Lampe new. afficheur := AfficheurTextuelDeLampe new. lampe afficheur: afficheur.	Lampe eteinte
lampe allumer.	Lampe allumée de couleur yellow
lampe eteindre.	Lampe eteinte
lampe couleur: Color red.	Lampe eteinte
lampe allumer.	Lampe allumée de couleur red
lampe couleur: Color blue.	Lampe allumée de couleur blue
lampe arreterAffichage.	
lampe couleur: Color green.	
lampe eteindre.	
lampe demarrerAffichage.	Lampe eteinte
lampe allumer.	Lampe allumée de couleur green

5 Polymorphisme

Chargez (file-in) la classe **AfficheurGraphiqueDeLampe** dans le fichier **.st** fourni (cf. exercice 3 pour l'import de fichiers). Utilisez une instance de cette classe à la place de l'afficheur textuel. Remarquez que les différentes classes ont été réutilisées sans modification. C'est une conséquence directe du polymorphisme. En effet, les instances des 2 classes **AfficheurGraphiqueDeLampe et AfficheurTextuelDeLampe** savent répondre aux messages envoyés par une lampe à son afficheur.

6 Feu tricolore

Toujours dans la catégorie **Trafic**, définissez la classe **FeuTricolore**. Les instances de cette classe doivent avoir trois lampes : la première rouge, la seconde orange et la troisième verte. Le feu tricolore, mémorise la lampe allumée à un instant donné. Lorsqu'il reçoit le message **allumerLampeSuivante**, il éteint la lampe courante, détermine puis allume la lampe suivante.

A l'initialisation, le feu tricolore associe à chaque lampe un afficheur graphique. Chaque afficheur peut être placé sur l'écran à l'aide du message **position:** qui prend comme paramètre un point dans l'écran. Exemple :

unAfficheurGraphique position: 100@200 place l'afficheur au point d'abscisse 100 et de coordonnée 200

Enfin, le feu tricolore doit disposer de deux méthodes **demarrerAffichage** et **arreterAffichage** qui font respectivement apparaître et disparaître les afficheurs des lampes de l'écran.