Rapport sur l'outil Pharo

En découvrant Pharo, j’ai été intrigué par son approche purement orientée objet et son environnement interactif. Pharo, basé sur Smalltalk, n'est pas juste un langage de programmation mais aussi un IDE (Environnement de Développement Intégré) très intégré. Ce rapport détaille mon exploration de certaines fonctionnalités clés de Pharo, notamment les collections, les conditionnels, la création de classes, et son style de codage unique.

1. En savoir plus sur les collections dans Pharo et leurs itérateurs

Qu'est-ce qu'une collection ?
Une collection en Pharo est simplement une structure qui permet de regrouper des éléments. Elle sert à manipuler facilement des ensembles de données sans se soucier de leur gestion interne.

Types de collections dans Pharo
En explorant Pharo, j'ai découvert plusieurs types de collections :
- Array : Un tableau fixe.
- OrderedCollection : Un tableau extensible qui garde l’ordre d’insertion.
- Set : Une collection d’éléments uniques, non ordonnée.
- Dictionary : Permet de stocker des paires clé-valeur.
- Bag : Similaire à Set, mais accepte les doublons.

Itérer sur des collections
La bibliothèque standard de Pharo propose plusieurs façons d’itérer sur ces collections :
- `do:` pour parcourir chaque élément,
- `collect:` pour appliquer une transformation sur chaque élément et renvoyer une nouvelle collection,
- `select:` pour filtrer les éléments selon une condition,
- `reject:` pour l’inverse de `select:`,
- `detect:` pour trouver le premier élément qui satisfait une condition.

Voici un exemple que j’ai essayé pour itérer avec `do:` :

myCollection := OrderedCollection with: 1 with: 2 with: 3.
myCollection do: [:each | Transcript show: each; cr].

Comment j'ai trouvé ces informations ?
Je les ai obtenues principalement en explorant la documentation intégrée de Pharo et en consultant des tutoriels en ligne. J'ai aussi beaucoup expérimenté dans le playground, ce qui m'a permis de mieux comprendre le fonctionnement des collections.

2. En savoir plus sur les conditionnels dans Pharo

Écriture des conditionnels
Dans Pharo, les conditions sont gérées différemment des autres langages. Au lieu d’utiliser une syntaxe classique comme `if...else`, on envoie des messages aux objets booléens. Par exemple, pour exécuter un bloc de code si une condition est vraie, on utilise `ifTrue:`.

Voici un exemple :

```
a := 10.
b := 5.
a > b ifTrue: [Transcript show: 'a est plus grand que b'; cr].
```

Différences avec d'autres langages
Contrairement à Python ou Java, où les conditions sont définies par des mots-clés comme `if` ou `else`, Pharo reste fidèle à son paradigme orienté objet, en traitant les conditions comme des messages envoyés aux objets. Au début, c'est un peu déstabilisant, mais après quelques essais, j'ai trouvé ça plutôt élégant.

Avantages et inconvénients
- Avantages : Cette approche renforce la cohérence du paradigme objet de Pharo. Tout est vraiment un objet, même les structures de contrôle.
- Inconvénients : C'est un peu moins intuitif pour quelqu'un qui vient de langages traditionnels comme moi, surtout au début.

Comment j'ai trouvé ces informations ?
J'ai suivi des exemples de code dans la documentation officielle et j'ai testé plusieurs combinaisons dans le playground.

3. Apprenez à créer des classes et des méthodes

Création de classes et méthodes
La création de classes dans Pharo est très simple et se fait directement via l'interface. C'est un peu déroutant au début car il faut passer par l'interface graphique plutôt que de tout écrire dans un fichier texte, comme dans les autres langages. Voici un exemple basique que j'ai réalisé :

```
Object subclass: Personne [
    Personne >> direBonjour [
        ^ 'Bonjour, je suis une personne.'
    ]
]
```

Ensuite, j'ai instancié la classe et utilisé la méthode `direBonjour` :

```
personne := Personne new.
Transcript show: personne direBonjour.
```

Problèmes rencontrés
Le principal problème que j'ai rencontré est l'adaptation à l'IDE. Pharo est très différent des éditeurs de texte traditionnels que j'utilise d'habitude. Il m'a fallu du temps pour comprendre comment créer et organiser des classes et des méthodes.

Comment j'ai trouvé ces informations ?
J'ai suivi plusieurs tutoriels vidéo et j'ai parcouru la documentation officielle de Pharo. L'environnement fournit aussi de nombreux outils d'aide pour apprendre à naviguer et à écrire du code.

4. Découvrez le style de codage de base de Pharo

Règles courantes
Pharo adopte un style de codage assez minimaliste. Les méthodes doivent être courtes et lisibles, et on évite la complexité inutile. J'ai appris qu'il vaut mieux écrire plusieurs petites méthodes plutôt qu'une grosse méthode qui fait tout.

Voici quelques règles que j'ai trouvées :
- Utiliser des noms de méthodes explicites.
- Écrire des méthodes courtes qui ne font qu'une seule chose.
- Éviter la duplication de code.

Outils de vérification
J'ai découvert un outil nommé SmallLint qui permet de vérifier si le code suit bien les conventions de style. Il détecte les violations et propose des suggestions pour améliorer le code.

Exemple de code qui enfreint les règles
J'ai écrit cet exemple qui enfreint certaines bonnes pratiques en étant trop long et peu clair :

```
Personne >> informationsComplet: nom age [
   Transcript show: 'Nom : ', nom; show: ' Age : ', age.
   Transcript show: 'Message additionnel inutile.'.
   ^ 'Fin.'
]
```

5. Suppléments : Cascades et fermetures de blocs

Cascades
En Pharo, les cascades permettent d'envoyer plusieurs messages au même objet de manière plus concise. J'ai trouvé cette fonctionnalité très pratique pour éviter de répéter le même objet plusieurs fois.

Exemple :
```
personne := Personne new.
personne direBonjour; direAuRevoir.
```

Fermetures de blocs
Les fermetures (ou blocs) sont des objets dans Pharo et peuvent être utilisées pour capturer un morceau de code avec son contexte. J'ai trouvé cet exemple dans la documentation et l'ai testé dans le playground :
```
myBlock := [ :x | x * 2 ].
Transcript show: (myBlock value: 10).
```

Interactions avec la communauté
J'ai posé quelques questions sur le serveur Discord de Pharo, notamment sur les fermetures et l'utilisation des cascades. J'ai reçu des réponses utiles, et cela m'a permis d'approfondir ma compréhension de ces concepts.

Conclusion

En tant que nouveau venu dans Pharo, j'ai trouvé cet environnement très différent des outils que j’ai utilisés auparavant, mais aussi très riche et puissant. Les collections, conditionnels et classes sont abordés d'une manière orientée objet qui demande un peu de temps pour s’y adapter, mais qui devient ensuite naturelle. L'IDE demande aussi un temps d’adaptation, mais une fois à l’aise avec, il permet une grande interactivité dans l’écriture de code.