Manuel de Travaux Pratiques

Le langage JAVA

Table des matières

Itération 1 : Premiers pas

Contexte : En suivant les instructions du formateur, vous allez écrire un premier programme en Java©. Ce code sera ensuite compilé et exécuté.

Instructions

Etape 1 : Recopier un exemple de code

Pour cette première faisons simple, avec un simple éditeur de texte recopiez l'exemple de code suivant dans le fichier Main.java :

```
/**
 * Mon premier exemple
 * @author beInSoft
 */
public class Main {

	//Point d'entrée
	public static void main(String[] args) {
	 sayHello();
	}

	/*
	 * Méthode qui dit bonjour
	 */
	private static void sayHello(){
	 System.out.println("Hello world");
	}
}
```

Etape 2 : Compilation

Vous allez compiler le fichier Main.java créé lors de la précédente étape.
Pour pouvoir faire cette opération en ligne de commande, il vous faut un environnement de développement correctement configuré sur votre poste. La commande java –version permet de le savoir rapidement.

Si Java n'est pas installé sur votre poste, il vous faut télécharger le SDK (Software Development Kit) sur Internet, décompresser le fichier sur votre disque dur, et mettre à jour votre variable d'environnement PATH. Le PATH vous permet d'utiliser les utilitaires du SDK (java, javac, jar, …) en ligne de commande depuis n'importe quel répertoire.

Voici les principales commandes à connaître :

> java –version	Savoir si Java est installé, et dans quelle version
> javac FICHIER.java	Compiler un fichier

> java CLASSE	Exécuter une classe
> jar	Créer une archive

Après compilation vous devez obtenir le fichier Main.class.

Etape 3 : Exécution

Toujours en ligne de commande, exécutez le programme que vous avez précédemment compilé.

Quelques réflexions :

Travaille t-on en ligne de commande ?
Un programme Java est-il toujours constitué d'un seul fichier ?
Comment organise t-on le code, alors ?
Comment la mémoire est-il gérée, effacée…

Optionnel : En suivant les instructions du formateur, passez le prénom de la personne à qui vous souhaitez dire bonjour en paramètre de votre programme.

Itération 2 : La Syntaxe de base

Avant d'aborder les notions essentielles de l'objet et de Java, nous allons commencer par voir les similitudes avec d'autres langages déjà rencontrés. Rien de tel que la syntaxe pour commencer à nous familiariser avec ce langage.

Vous allez repartir du fichier Main.java créée précédemment.

Etape 1 : Parcourir un tableau

Déclarez le tableau **listeDeLecture** dans lequel vous allez définir des titres de chansons et/ou d'albums de musique.

Puis parcourir ce tableau afin d'afficher chacun des titres sur la console.

Etape 2 : Quelques fonctions, appelées méthodes

Déplacez le code permettant la lecture des titres dans une méthode de classe appelées **lectureListe()**, et appelez cette méthode pour qu'elle s'exécute.

Ajoutez la méthode **lectureAleatoireListe()** qui effectue une lecture du tableau de manière aléatoire.

Pour information l'instruction *Math.random()* permet de faire un tirage aléatoire entre 0 et 1.

Pour aller plus loin

Que se passe t-il si le tableau est déclaré dans la méthode main() ?
Est-il possible de le déclarer en dehors ?
Comment les arguments sont-ils passés en paramètre des méthodes ?

Itération 3 : Première classe et premiers objets

Nous venons de voir comment définir une classe en Java, vos allez définir votre première classe et vos premiers objets.

Etape 1 : Un tableau de « Titre »

Définissez la classe Titre est remplacez le tableau de chaine de caractères de l'itération précédente par un tableau d'objets.

Le résultat doit être le même que celui de l'itération précédente.

Un titre est défini par une durée en seconde, le nom du titre et son genre musical. Le genre musical peut prendre une des valeurs suivante : Rock, Jazz ou Pop.

Etape 2 : Encapsulation

Une technique courante consiste à déclarer « privé » les attributs et de passer par des accesseurs pour lire ou modifier (get/set) les attributs correspondants.

Déclarez « private » tous les attributs de la classe Titre, et observez l'incidence sur votre code.

Ajoutez les accesseurs pour renseigner les informations relatives à vos titres. Que pensez-vous de votre code d'initialisation ?

Quel est l'intérêt de l'encapsulation alors ?

Etape 3 : Constructeur et surcharge

Ajoutez un constructeur permettant d'initialiser le nom de nom de vos titres.

Rappel : lorsque plusieurs méthodes (ou constructeurs) ont le même nom dans une classe, on dit que cette méthode (ce constructeur) est surchargée. C'est la JVM qui appellera la bonne méthode, en fonction des paramètres (nombre et type) passés.

Itération 4 : Un peu de conception

Pour cette étape, nous allons vous fournir le cahier de charge suivant :

Au démarrage, l'application BeInPlayer comporte un certain nombre de titres.

Lorsque l'application démarre tous les titres sont correctement initialisés, et portés à la connaissance de l'utilisateur. L'application propose alors plusieurs fois à l'utilisateur de compléter sa liste de lecture de manière aléatoire avec un titre. Si l'utilisateur refuse, la liste de lecture est immédiatement jouée par le lecteur en suivant l'ordre dans lequel les titres ont été ajoutés.

Etape 1 : Analyse

A l'aide du formateur, vous allez pouvoir en réaliser l'analyse de ce cahier des charges. Quels sont les diagrammes UML que vous pourriez utiliser ?

Etape 2 : Conception

A l'aide du formateur, vous allez maintenant pouvoir en réaliser la conception. Là encore quels sont les diagrammes UML les plus utiles ?

Etape 3 : Implémentation

A partir du diagramme de classes obtenu en fin de conception et des autres diagrammes transposez ces éléments en code Java.

Par quelle classe devriez-vous commencer ?

Etape 4 : Tests

Enfin, n'oubliez pas de tester le résultat.

Itération 5 : Les packages

Notre application s'est enrichie de quelques classes, il nous faut commencer à les organiser. Nous allons donc faire appel aux packages !

Etape 1 : Quel organisation

Compte tenu des classes en présence, quelle est selon vous la meilleure organisation ? Vous pouvez éventuellement réaliser un diagramme de package pour illustrer votre réflexion.

Une fois votre nouvelle organisation validée, réalisez ces packages pour ranger vos classes. Attention aux modifications dans vos classes…

Etape 2 : Encapsulation

Réfléchissez maintenant à quelle visibilité vous allez vouloir donner à vos classes.

Itération 6 : Héritage

Nous allons faire évoluer nos modèles UML avec de nouvelles contraintes :

Etape 1 : Les clips vidéo

Une nouvelle contrainte apparaît, nous souhaiterions pouvoir gérer les clip vidéo. En effet certain Titre dispose d'un clip vidéo et il est important dans ce cas de pouvoir renseigner le nom du réalisateur du clip.

Etape 2 : Les albums

Autre contrainte, nous souhaiterions pouvoir disposer de d'album, tout en sachant qu'un album dispose d'un auteur et d'un nom, et qu'il référence un certain nombre de Titre. Par ailleurs, les albums pourront également être ajouté à notre liste de lecture. Par contre lorsqu'un album est lu, tous ses titres sont lus en suivant l'ordre des pistes.

Itération 7 : Les collections

Etape 1 : Tableau vs Collection

Remplacez tous les tableaux utilisés jusqu'à cette itération par des collections.

Etape 2 : Recherche par Auteur

Chaque Titre dispose désormais d'un Auteur et on souhaite désormais pouvoir effectuer des recherches par Auteur.
Réorganisez votre code afin d'intégrer cette nouvelle contrainte.

Itération 8 : La classe Object

Maintenant que vous savez que toutes les classes héritent de la classe Object, vous pouvez le préciser sur ces dernières.

Etape 1 : Unicité dans les Set

Lors de l'étape précédente, il n'était pas évident d'avoir l'unicité des auteurs, comme clés dans la Map.
L'unicité dans les « Set » set fait au sens des méthodes *equals* et *hashCode*. Redéfinissez ces deux méthodes sur la classe Auteur, et testez de nouveau le résultat. Par défaut cette détection de l'unicité se fait sur les adresses mémoires. Modifiez votre code afin de vous assurer de l'unicité des artistes par rapport à leur nom.

Etape 2 : Redéfinition de la méthode toString

Profitez-en pour redéfinir la méthode *toString* sur la classe Titre afin de faciliter vos différents affichages.

Itération 9 : Les interfaces

Etape 1 : Initialisation du catalogue

Imaginons qu'on délègue à une autre équipe le développement de toute la couche d'accès aux données de notre application. Et nous savons que nous allons avoir besoin d'accéder à cette couche pour initialiser le catalogue.

Evaluez vos besoins en terme de données et définissez le contrat attendu au travers d'une interface.

Etape 2 : Le bouchonnage (Mock)

Pour ne pas avoir à attendre que le couche d'accès aux données soit entièrement développée pour tester le reste de l'application, et notamment l'initialisation du catalogue, vous allez pouvoir implémenter cette interface au travers d'une objet de type « Mock »

Itération 10 : les bases de données

Avant de plonger dans le code, il nous faut installer une base de données. Nous allons vous fournir une base de données hsqlDB qui a l'avantage d'être simple à installer :

- Le fichier runServer.bat vous permet de démarrer le serveur de base de données
- Le fichier runManager.bat vous permettra de tester vos requêtes.

Etape 1 : Du Mock à La Base

Créez une classe d'implémentation de l'interface que vous avez réalisée lors de l'itération 9. Cette fois ci, les données seront récupérées en base de données.