Cours :	**Java Base**	Rôle :	Manipuler
Sujet :	**Mise en pratique des notions POO**	Formateur :	Yannick Boogaerts

Création d'un Menu générique et configurable.

Sprint 1 : Créer et exécuter une action résultat d'un choix dans un menu.

Diagramme de classe

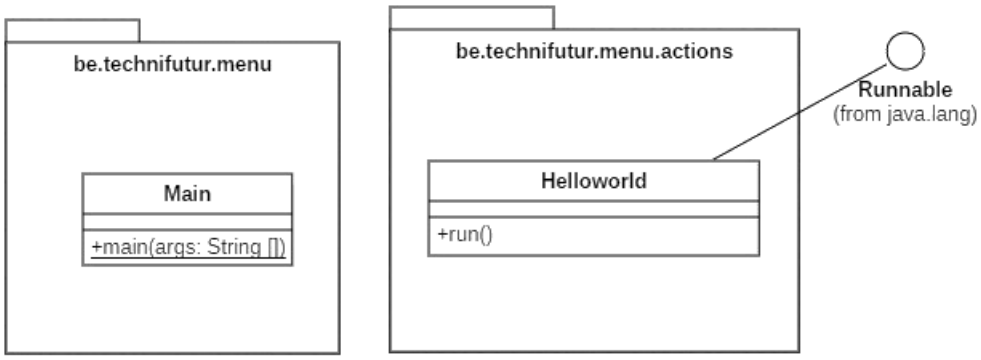
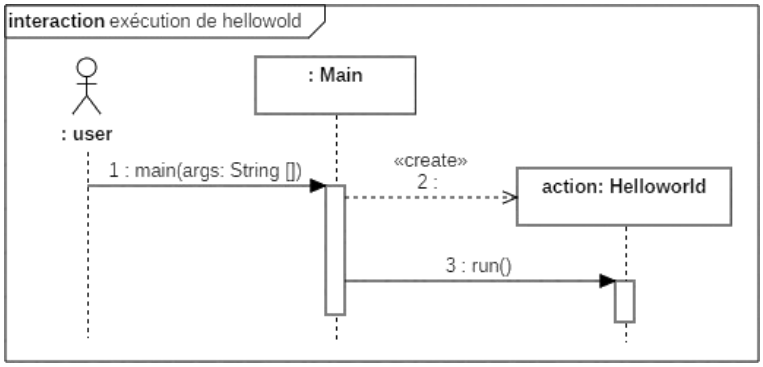


Diagramme de séquence



Description des tâches

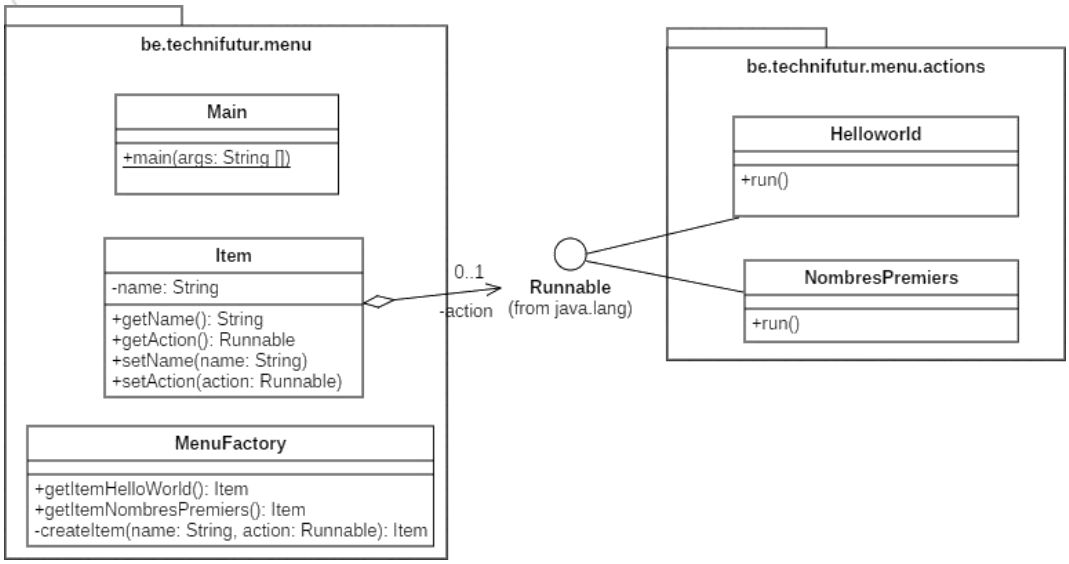
1. Création d'un projet java menu
   1.1. Mettre le projet sous contrôle de version avec « git ».
2. Création de 2 classes
   2.1. « be.technifutur.menu.Main »
      2.1.1. Ajout de la méthode « public static void main(String[] args) »
   2.2. « be.technifutur.menu.actions.Helloworld »
      2.2.1. Définir que la classe implémente l'interface « java.lang.Runnable »
      2.2.2. Ajout de la méthode « @Override public void run() »
3. Implémentation des méthodes
   3.1. Méthode : Helloworld.run()
      afficher « Hello wold » en console.
   3.2. Méthode : Main.main()
      3.2.1. Créer une instance de Helloworld et socker sa référence dans une variable « action » de type Runnable.
      3.2.2. Appeler la méthode run() de action.
4. Exécuter le programme

Pour aller plus loin

Créer un autre classe « action » sur le modèle de Helloworld qui exécute un des exercices de logique.

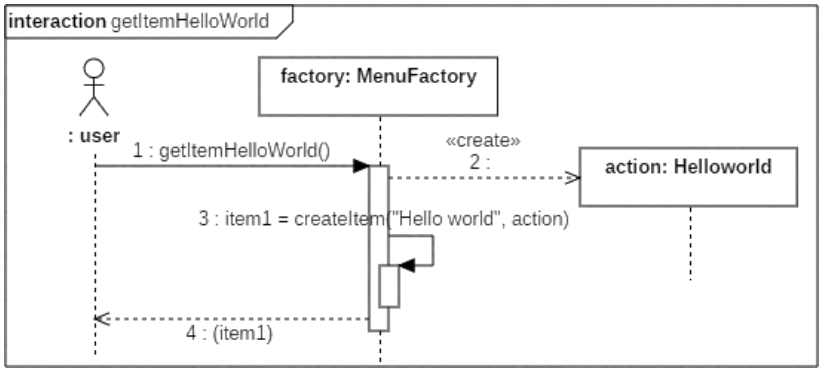
Sprint 2 : Créer, initialiser et utiliser des items de menu

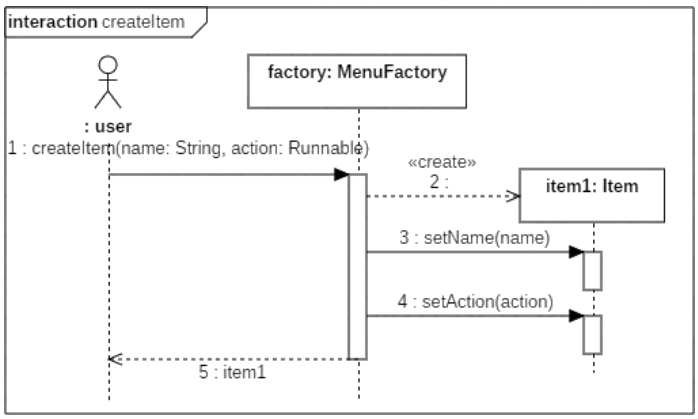
Diagramme de classe



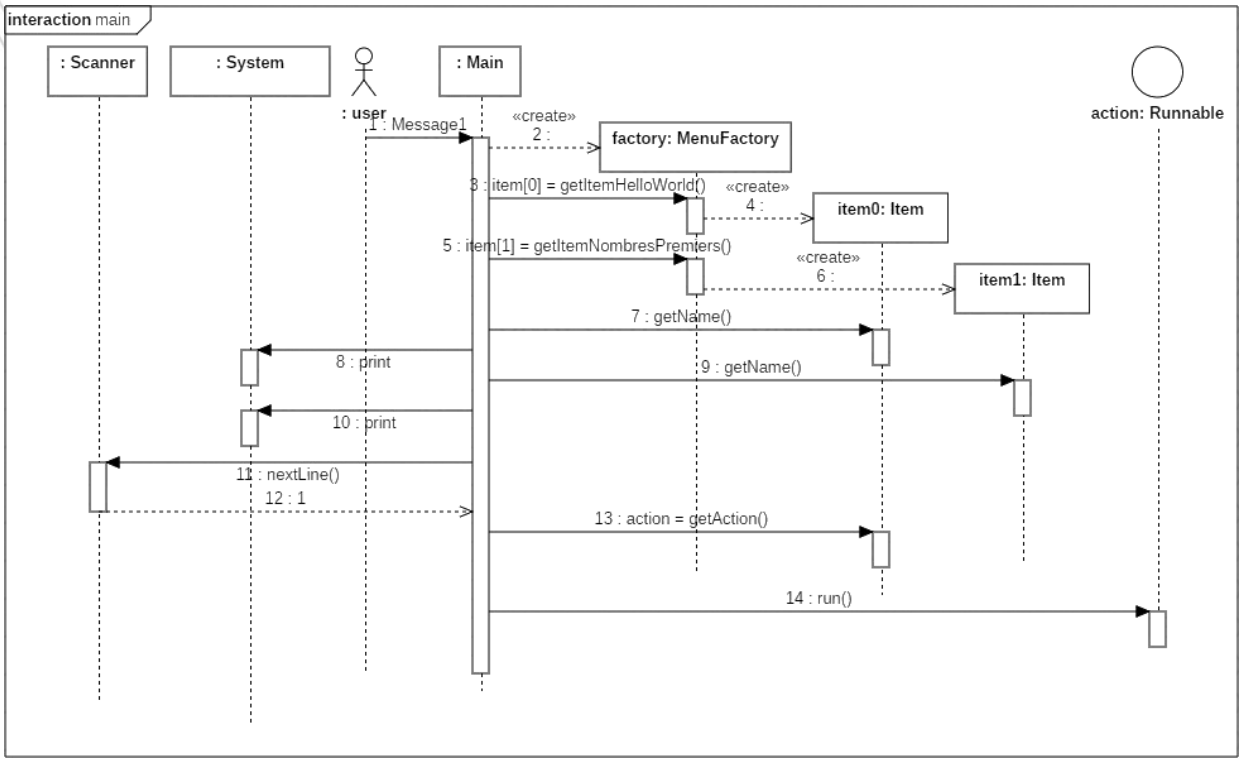
Diagrammes de séquences

Méthodes de la classe MenuFactory





Méthode de la classe Main



Description des tâches

1. Création de la classe « Item »
   1.1. Ajout de 2 attributs privés
      1.1.1. name : String
      1.1.2. action : Runnable
   1.2. Ajout des accesseurs getter et setter sur les 2 attributs.
2. Création de la classe « MenuFactory »
   2.1. Ajout de la méthode getItemHelloWorld qui crée, initialise et retourne un Item
      2.1.1. Créer un nouvel Item
      2.1.2. Lui donnée le nom « Hello world » grâce au setter
      2.1.3. Créer nouveau HelloWorld et l'attribuer à l'item grâce au setter
   2.2. Ajout de la méthode getItemNombresPremier qui crée, initialise et retourne un Item
      2.2.1. Créer un nouvel Item
      2.2.2. Lui donnée le nom « Nombres premiers » grâce au setter
      2.2.3. Créer nouveau NombresPremiers et l'attribuer à l'item grâce au setter
   2.3. Refactoriser les 2 méthodes créées en regroupant le code commun
      2.3.1. Ajouter une méthode privée createItem(name :String, action :Runnable) :Item
      2.3.2. Déplacer le code commun des 2 méthodes.
      2.3.3. Appeler la nouvelle méthode dans les 2 autres.
3. Réécriture du code de la méthode main
   3.1. Création et initialisation des objets
      3.1.1. Créer un MenuFactory
      3.1.2. Grâce aux méthodes de la factory récupérer les Item et les sauver dans un tableau
   3.2. Afficher la clé et nom des items

```
(1) Hello Word
(2) Nombres premiers
```

   3.3. Demander à l'utilisateur de faire un choix
   3.4. Exécuter l'action choisie
      3.4.1. Récupérer l'Item choisi dans le tableau
      3.4.2. Récupérer l'action dans l'Item
      3.4.3. Lancer la méthode run() de l'action

Pour aller plus loin

Créer une nouvelle action et l'ajouter au menu sur le même modèle que les 2 autres

Sprint 3 : Créer, initialiser et utiliser un menu en MVC (Model, Vue, Contrôleur)

Diagramme de classe

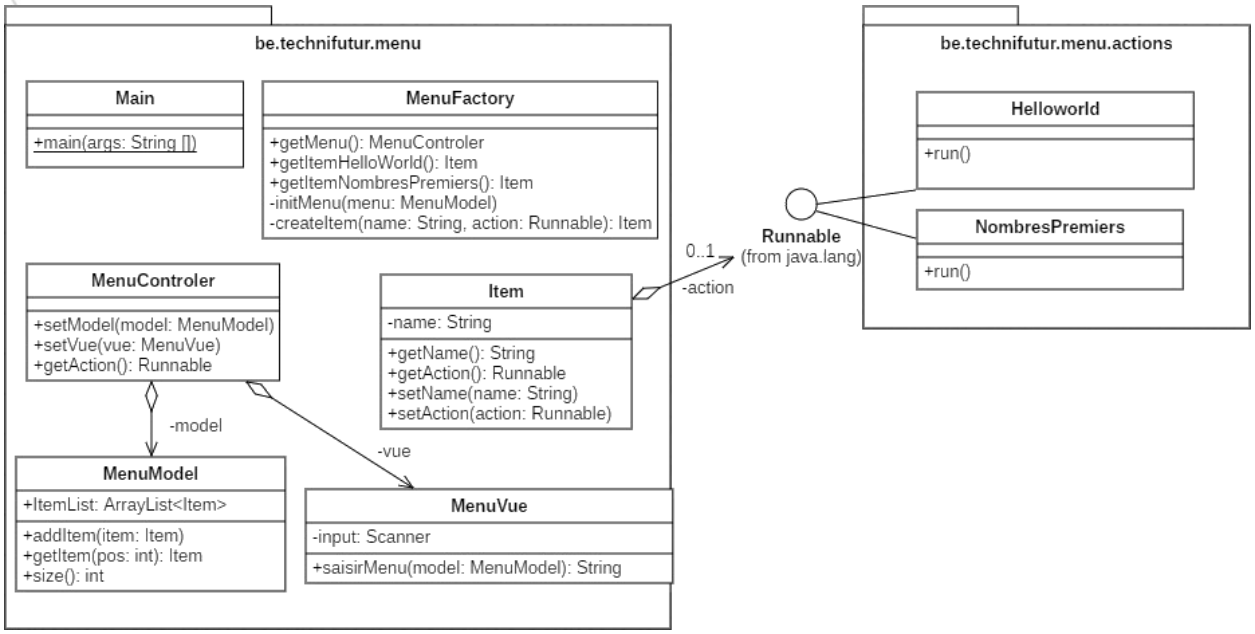
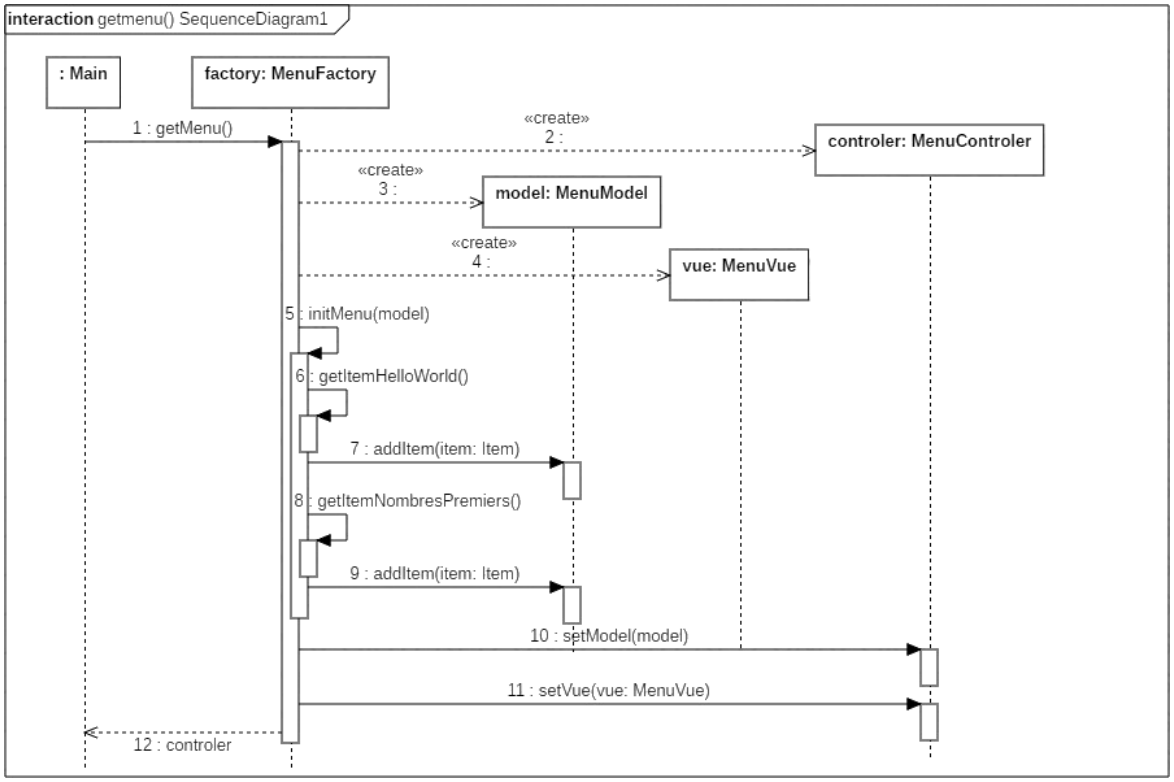


Diagramme de séquence de la méthode MenuFactory.getMenu()



Description des tâches

1. Créer la classe « MenuModel »
   1.1. Ajouter 1 attribut privé itemList : ArrayList<Item> et l'initialiser avec une nouvelle instance.
   1.2. Ajouter 3 méthodes publiques
      1.2.1. + addItem(item : Item)
      le code ajoute la valeur du paramètre item à la fin de la liste itemList.
      1.2.2. + getItem(pos : int) : Item
      le code retourne la référence de l'Item en position pos dans itemList
      si pos ne correspond à aucun éléments de itemList la méthode retourne null.
      1.2.3. + size() : int
      le code retourne le nombre d'Item dans itemList.

2. Créer la classe « MenuVue »
   2.1. Ajouter 1 attribut privé input : Scanner
      2.1.1. Initialiser l'attribut avec un nouveau Scanner pointant sur « System.in »
   2.2. Ajouter 1 méthode publique « + saisirMenu(menu : MenuModel) : String
      le code affiche les items du menu, demande le choix de l'utilisateur, saisi le choix grâce à input et retourne ce choix.

```
( 1) Hello Word
( 2) Nombres premiers
choix :
```

3. Créer la classe « MenuControler »
   3.1. Ajouter 2 attributs privés
      3.1.1. – model : MenuModel
      3.1.2. – vue : MenuVue
   3.2. Ajouter 2 accesseurs publique en écriture (setter). 1 par attribut
   3.3. Ajouter une méthode publique + getAction() :Runnable.
      3.3.1. Récupération du choix de l'utilisateur grâce à la vue
      3.3.2. Transformation du choix en position
      3.3.3. Si la position est valide
         3.3.3.1. Récupération de l'item puis de l'action à partir du model
      3.3.4. Retourner l'action ou null
4. Ajouter 2 méthodes à la classe MenuFactory
   4.1. + getMenu() : MenuControler
      4.1.1. Creation des objets
         4.1.1.1. Créer un MenuControler
         4.1.1.2. Créer un MenuModel
         4.1.1.3. Créer un MenuVue
      4.1.2. Initialisation du model grâce à la méthode initMenu
      4.1.3. Initialisation du controleur grâce à ces 2 setters
      4.1.4. Retourner le controleur
   4.2. – initMenu(menu :MenuModel)
      le code ajoute dans le menu, grâce à sa méthode addItem(…), les items en appelant les autre méthodes de la factory

Pour aller plus loin