

Examen pratique de janvier 2024

TABLEAU KANBAN

L'application à réaliser doit permettre de créer un tableau répertoriant les différentes tâches à réaliser pour mener à bien un projet. En développement informatique, un tableau de ce type est appelé un **tableau *kanban***.

A FAIRE	EN COURS	TERMINÉE
Documenter les fonctions	Coder les fonctionnalités indispensables	Créer le projet dans Eclipse
Coder le reste des fonctionnalités	Réaliser les tests unitaires	Lire l'énoncé de l'examen
Créer et déposer l'archive ZIP du projet		

Fig. 1 – Tableau kanban pour l'examen de PRB.

Comme montré sur la figure 1, le tableau sera constitué de plusieurs colonnes, permettant ainsi d'organiser les tâches par **état d'avancement**. Dans l'exemple, ces états sont au nombre de trois : « *A faire* », « *En cours* » et « *Terminée* ».

Il doit être également possible de **faire progresser une tâche** d'une colonne à l'autre afin d'indiquer le changement de son état d'avancement.

Par exemple, une fois la tâche « *Coder les fonctionnalités indispensables* » terminée, le tableau devient :

A FAIRE	EN COURS	TERMINÉE
Documenter les fonctions	Réaliser les tests unitaires	Créer le projet dans Eclipse
Coder le reste des fonctionnalités		Lire l'énoncé de l'examen
Créer l'archive ZIP du projet et la déposer en ligne		Coder les fonctionnalités indispensables

Fig. 2 – Tableau kanban modifié après la finalisation d'une tâche supplémentaire.

DESCRIPTION DE L'APPLICATION

Au lancement de l'application, l'utilisateur doit encoder les **intitulés des colonnes** du tableau *kanban*, autrement dit les différents états d'avancement que peut prendre une tâche. Ces derniers ne doivent pas nécessairement correspondre à ceux de l'exemple donné précédemment.

Les intitulés doivent être encodés dans l'ordre souhaité, c'est-à-dire de l'**état initial** (par exemple, « *A faire* ») à l'**état final** (par exemple, « *Terminée* »).

Pour mettre fin à l'encodage, l'utilisateur doit simplement presser la touche « *Entrée* » (ce qui correspond à l'encodage d'une chaîne vide).

Un **menu** est ensuite affiché, proposant 3 options à l'utilisateur : « *Ajouter une tâche* », « *Faire progresser une tâche* » et « *Quitter* ».

La 1^{ère} option, « **Ajouter une tâche** », permet à l'utilisateur d'encoder une nouvelle tâche. Celle-ci apparaît dès lors dans la 1^{ère} colonne du tableau *kanban*.

La 2^e option, « **Faire progresser une tâche** », permet à l'utilisateur de spécifier la tâche dont l'état d'avancement doit évoluer vers le suivant. A moins que cette dernière n'ait déjà atteint l'état final, elle apparaît alors dans la colonne suivante du tableau *kanban*.

Référez-vous aux exemples d'exécution donnés en fin d'énoncé afin de mieux comprendre le résultat attendu.

AVANT-PROPOS

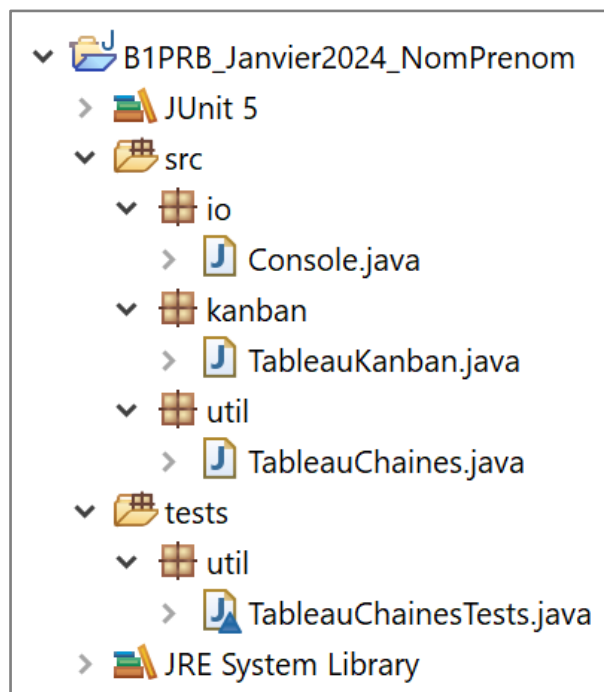
À deux exceptions près, les types **String** et **PrintStream**, vous ne pouvez pas utiliser le paradigme orienté objet pour coder cette application. N'oubliez pas de qualifier toutes vos fonctions de **static** !

PRÉPARATION

1. Dans Eclipse, créez un projet Java nommé **B1PRB_Janvier2024_NomPrenom**, où **NomPrenom** doit être remplacé par vos nom et prénom.

N'oubliez pas de vérifier que l'encodage utilisé pour les fichiers est l'**UTF-8**. Pour ce faire, il faut suivre le chemin : *Window > Preferences > General > Workspace > Text file encoding*.

2. Reproduisez exactement l'organisation suivante :



La classe **Console** est à télécharger au début de la page *HELMo Learn* du cours. Les autres classes doivent être créées par vos soins. La fonction **main** doit être déclarée dans la classe **TableauKanban**.

Le projet déposé sur l'espace de cours doit être complet et exécutable !

FONCTIONNALITÉS INDISPENSABLES


Les fonctionnalités suivantes doivent être opérationnelles, tout en veillant à respecter les consignes données dans leur description :

- La fonction *ajouterElement* de la classe **TableauChaines** doit être correctement réalisée, testée et utilisée.
- La fonction *afficherTableauKanban* de la classe **TableauKanban** doit être correctement réalisée et utilisée.
- Dans la fonction *main* de la classe **TableauKanban**, un tableau de chaînes de caractères à une dimension doit être correctement créé (voir la section « Les structures de données »). Les **intitulés des colonnes** saisis doivent y être ajoutés à l'aide de la fonction **TableauChaines.ajouterElement**.
- Dans la fonction *main* de la classe **TableauKanban**, un tableau de chaînes de caractères à deux dimensions doit être correctement créé et initialisé afin d'y répertorier les **tâches** par état d'avancement (voir la section « Les structures de données »).
- La fonction *main* doit permettre l'acquisition des intitulés des colonnes, l'ajout d'au moins une tâche à la première colonne du tableau kanban (à nouveau, à l'aide de la fonction **TableauChaines.ajouterElement**) et l'affichage de ce dernier (voir l'exemple d'exécution « MINIMUM REQUIS » en fin d'énoncé).

Dans la version minimale, il n'est pas nécessaire de vérifier la validité des données saisies par l'utilisateur.

Si l'une de ces fonctionnalités n'est pas observée, cela entraîne automatiquement l'échec pour l'examen. Attention, la réussite de ces fonctionnalités seules ne garantit pas l'obtention d'une note supérieure ou égale à 10 / 20 ! Ne vous contentez pas de ces fonctionnalités et veillez à soigner les tests unitaires et la javadoc des fonctions.

APPROCHE RECOMMANDÉE

Dans un premier temps, **ne réalisez que les fonctions indispensables**. Ces dernières sont indiquées dans la section « *Fonctionnalités indispensables* » et sont marquées d'un point d'exclamation  au niveau de leurs descriptions.

Réalisez ensuite la fonction **main** sur base des directives données dans la section « *Fonctionnalités indispensables* » et de l'exemple d'exécution **MINIMUM REQUIS**.

Il est recommandé de ne réaliser la suite du programme que lorsque ces deux premières étapes sont terminées.

LES STRUCTURES DE DONNÉES

ENREGISTRER LES INTITULÉS DES COLONNES

Utilisez un **tableau de chaînes de caractères à une dimension** pour enregistrer les intitulés des colonnes, dans l'ordre de la saisie et en lettres majuscules (pour ce faire, utilisez la fonction *toUpperCase* de la classe **String**).

Initialement, ce tableau doit être vide :



Dans le cas de l'exemple donné à la figure 1, une fois les 3 intitulés ajoutés successivement à l'aide de la fonction **TableauChaines.ajouterElement**, le tableau obtenu est le suivant :

"A FAIRE"	"EN COURS"	"TERMINEE"
0	1	2

ENREGISTRER LES TÂCHES

Utilisez un **tableau de chaînes de caractères à deux dimensions** pour enregistrer les tâches selon leurs états d'avancement. Chaque ligne du tableau correspond à l'une des colonnes du tableau *kanban*, répertoriant ainsi toutes les tâches de l'état d'avancement associé.

Initialement, ce tableau ne comporte que des lignes vides :

0	
1	
2	

Dans le cas de l'exemple donné à la figure 1, le tableau obtenu est le suivant :

0	"Documenter les fonctions"	"Coder le reste des fonctionnalités"	"Créer et déposer l'archive ZIP du projet"
1	"Coder les fonctionnalités indispensables"	"Réaliser les tests unitaires"	
2	"Créer le projet dans Eclipse"	"Lire l'énoncé de l'examen"	

Comme indiqué précédemment, toutes les tâches qui apparaissent dans une même ligne ont le même état d'avancement. Dans ce cas par exemple, les tâches de la première ligne (la ligne d'**indice 0 du tableau à deux dimensions**), c'est-à-dire « Documenter les fonctions », « Coder le reste des fonctionnalités » et « Créer et déposer l'archive ZIP du projet », sont toutes « A faire » (qui correspond à l'état spécifié à l'**indice 0 du tableau à une dimension**).

*Dans ce programme, veillez à ne pas confondre les lignes et les colonnes ! En effet, une **colonne du tableau kanban** est représenté sous la forme d'une **ligne du tableau à deux dimensions** ci-dessus.*

Pour faire progresser une tâche à l'état d'avancement suivant, il vous faudra la retirer de sa ligne actuelle à l'aide de la fonction `TableauChaines.retirerElement`, puis l'ajouter à la ligne suivante à l'aide de la méthode `TableauChaines.ajouterElement`.

LES CLASSES `TableauChaines` ET `TableauChainesTests`

Toutes les fonctions de la classe `TableauChaines` doivent être documentées avec des commentaires javadoc et testées à l'aide de JUnit 5.

Dans la classe `TableauChaines`, déclarez :



→ Une fonction nommée `ajouterElement` qui retourne une copie des éléments d'un tableau de chaînes de caractères à une dimension augmentée d'un nouvel élément :

```
public static String[] ajouterElement(String[] t,
                                     String element)
```

Le paramètre `t` est le tableau dont les éléments doivent être copiés.

Le paramètre `element` est l'élément à ajouter aux éléments du tableau `t`.

Exemple : si `t` est le tableau `["A", "B", "C"]` et `element` la chaîne `"D"`, alors le tableau retourné est `["A", "B", "C", "D"]`.

[SUGGESTION] Aidez-vous de la fonction `copyOf` de la classe `Arrays`.

[PRÉCONDITION] La référence réceptionnée par le paramètre `t` est supposée valide (ne vaut pas `null`).

→ Une fonction nommée `retirerElement` qui retourne une copie des éléments d'un tableau de chaînes de caractères à une dimension diminuée de l'un d'entre eux :

```
public static String[] retirerElement(String[] t,
                                     int position)
```

Le paramètre `t` est le tableau dont les éléments doivent être copiés.

Le paramètre `position` indique la position de l'élément du tableau `t` à exclure de la copie.

Exemple : si `t` est le tableau `["A", "B", "C", "D"]` et `position` l'entier `1`, alors le tableau retourné est `["A", "C", "D"]`.

[SUGGESTION] Vous pouvez utiliser la fonction `arraycopy` de la classe `System`, mais la réalisation d'une boucle est sans doute la solution la plus simple.

[PRÉCONDITIONS] Les paramètres réceptionnés par la fonction sont supposés valides (référence différente de `null` et position existante).

→ Une fonction nommée `positionDe` qui détermine la position d'une chaîne de caractères au sein d'un tableau de chaînes de caractères à une dimension, et ce, sans tenir compte de la casse (par exemple, la chaîne `"alice"` est considérée comme équivalente à la chaîne `"Alice"`) :

```
public static int positionDe(String[] t, String prefixe)
```

Le paramètre `t` est le tableau contenant les éléments parmi lesquels la recherche doit être effectuée.

Le paramètre `prefixe` est le début de la chaîne de caractères recherchée (par exemple, la chaîne `"ali"` permet de retrouver la chaîne `"Alice"`).

*La recherche dans le tableau doit s'effectuer **de gauche à droite**, à partir de l'indice 0 !*

***Exemple** : si `t` est le tableau `["Camille", "Bilal", "Aline", "Alice", "Diego"]` et `prefixe` la chaîne `"ali"`, alors la fonction retourne l'entier 2 (`"Aline"` étant la première chaîne trouvée qui débute par `"ali"`).*

[CAS PARTICULIER 1] Si aucune chaîne du tableau `t` ne commence par le préfixe spécifié, alors la fonction retourne -1.

[CAS PARTICULIER 2] La fonction doit pouvoir gérer la présence de références `null` au sein du tableau.

***Exemple** : si `t` est le tableau `["Camille", "Bilal", null, "Alice", "Diego"]` et `prefixe` la chaîne `"ali"`, alors la fonction retourne l'entier 3 (`"Alice"` étant la première chaîne trouvée qui débute par `"ali"`).*

[SUGGESTION] Aidez-vous des fonctions `startsWith`, `toLowerCase` et `toUpperCase` de la classe `String`.

[PRÉCONDITIONS] Les paramètres réceptionnés par la fonction sont supposés valides (références différentes de `null`).

→ Une fonction nommée ***positionDe*** qui détermine la position $[i, j]$ (i et j correspondant respectivement aux indices d'une ligne et d'une colonne) d'une chaîne de caractères au sein d'un tableau de chaînes de caractères à deux dimensions, et ce, sans tenir compte de la casse :

```
public static int[] positionDe(String[][] t,
                               String prefixe)
```

Le paramètre `t` est le tableau contenant les éléments parmi lesquels la recherche doit être effectuée.

Le paramètre `prefixe` est le début de la chaîne de caractères recherchée.

*La recherche dans le tableau doit s'effectuer **ligne par ligne**, à partir de la ligne d'indice 0, et ce à l'aide de la fonction `positionDe(String[], String prefixe)` déclarée précédemment !*

Exemple : si `t` est le tableau `[["Camille", "Bilal", "Eloi"], ["Aline"], ["Alice", "Diego"]]` et `prefixe` la chaîne `"ali"`, alors la fonction retourne le tableau `[1, 0]` ("`Aline`" étant la première chaîne trouvée qui débute par `"ali"`).

[CAS PARTICULIER 1] Si aucune chaîne du tableau `t` ne commence par le préfixe spécifié, alors la fonction retourne `null`.

[CAS PARTICULIER 2] La fonction doit pouvoir gérer la présence de références `null` en tant que références de ligne.

Exemple : si `t` est le tableau `[["Camille", "Bilal", "Eloi"], null, ["Alice", "Diego"]]` et `prefixe` la chaîne `"ali"`, alors la fonction retourne le tableau `[2, 0]` ("`Alice`" étant la première chaîne trouvée qui débute par `"ali"`).

[PRÉCONDITIONS] Les paramètres réceptionnés par la fonction sont supposés valides (références différentes de `null`).

LA CLASSE `TableauKanban`

À l'exception de la fonction `main`, toutes les fonctions de la classe `TableauKanban` doivent être documentées avec des commentaires javadoc. Cependant, aucune des fonctions de cette classe ne doit être testée à l'aide de JUnit 5.

Dans la classe `TableauKanban`, déclarez :



→ Une fonction nommée `afficherTableauKanban` qui affiche un tableau de type *kanban* :

```
public static void afficherTableauKanban(
    String[] intitules,
    String[][] colonnes)
```

Le paramètre `intitules` est un tableau contenant les intitulés des colonnes du tableau *kanban*.

Le paramètre `colonnes` est un tableau contenant les tâches organisées par état d'avancement.

Exemple : si `intitules` est le tableau `["Etat 1", "Etat 2", "Etat 3"]` et `colonnes` le tableau `[["Tâche 2", "Tâche 3"], [], ["Tâche 1"]]`, alors l'exécution de la fonction produit l'affichage suivant en console :

```
Etat 1 :
- Tâche 2
- Tâche 3

Etat 2 :
(vide)

Etat 3 :
- Tâche 1
```

[CAS PARTICULIER] Si une ligne du tableau à deux dimensions est vide, autrement dit si une colonne du tableau *kanban* ne contient aucune tâche, alors la fonction doit afficher le texte `"(vide)"`.

[PRÉCONDITIONS] Les paramètres réceptionnés par la fonction sont supposés valides (références du tableau et de ses lignes différentes de `null`).

→ Une fonction nommée `lireChaineNonVide` qui effectue l'acquisition sécurisée d'une chaîne de caractères :

```
public static String lireChaineNonVide(String question)
```

Le paramètre `question` est la chaîne de caractères à afficher en guise de question pour spécifier la valeur que doit saisir l'utilisateur.

*L'acquisition **doit être répétée** tant que la saisie ne correspond pas à une chaîne constituée d'au moins un caractère non blanc. Pour rappel un caractère blanc est par exemple : un espace, une tabulation...*

Exemple : si `question` est la chaîne de caractères `"Tâche à ajouter ? "`, alors l'exécution de la fonction pourrait produire l'affichage suivant en console :

```
Tâche à ajouter ?  _ _ ↵
Tâche à ajouter ?  ↵
Tâche à ajouter ?  Tâche 1 ↵
```

L'acquisition est ici répétée trois fois car la première saisie ne contient que des caractères blancs (plus précisément, deux espaces) et la seconde consiste en une chaîne vide.

Dans ce cas, la fonction retourne la chaîne `"Tâche 1"`.

[SUGGESTION] Aidez-vous, au choix, des fonctions `trim`, `isEmpty` et `isBlank` de la classe `String`.

[PRÉCONDITION] La chaîne `question` est supposée valide (ne vaut pas `null`).

LES EXEMPLES D'EXÉCUTION

MINIMUM REQUIS

Etape #1 ('Entrée' pour terminer) ? **A faire**
Etape #2 ('Entrée' pour terminer) ? **En Cours**
Etape #3 ('Entrée' pour terminer) ? **Terminée**
Etape #4 ('Entrée' pour terminer) ?

Tâche à ajouter ? **Lire l'énoncé de l'examen**

A FAIRE :

- Lire l'énoncé de l'examen

EN COURS :

(vide)

TERMINÉE :

(vide)

VERSION IDÉALE

Etape #1 ('Entrée' pour terminer) ? **A faire**
Etape #2 ('Entrée' pour terminer) ? **En Cours**
Etape #3 ('Entrée' pour terminer) ? **Terminée**
Etape #4 ('Entrée' pour terminer) ?

MENU

1. Ajouter une tâche
2. Faire progresser une tâche
3. Quitter

Choix ? **1**

Tâche à ajouter ? **Lire l'énoncé de l'examen**

A FAIRE :

- Lire l'énoncé de l'examen

EN COURS :

(vide)

TERMINÉE :

(vide)

MENU

1. Ajouter une tâche
2. Faire progresser une tâche
3. Quitter

Choix ? **1**

Tâche à ajouter ? **Coder les fonctions indispensables**

A FAIRE :

- Lire l'énoncé de l'examen
- Coder les fonctions indispensables

EN COURS :

(vide)

TERMINÉE :

(vide)

MENU

1. Ajouter une tâche
2. Faire progresser une tâche
3. Quitter

Choix ? **1**

Tâche à ajouter ? **Réaliser les tests unitaires**

A FAIRE :

- Lire l'énoncé de l'examen
- Coder les fonctions indispensables
- Réaliser les tests unitaires

EN COURS :

(vide)

TERMINÉE :

(vide)

MENU

1. Ajouter une tâche
2. Faire progresser une tâche
3. Quitter

Choix ? **1**

Tâche à ajouter ? **Coder la fonction main**

A FAIRE :

- Lire l'énoncé de l'examen
- Coder les fonctions indispensables
- Réaliser les tests unitaires
- Coder la fonction main

EN COURS :

(vide)

TERMINÉE :

(vide)

MENU

1. Ajouter une tâche
2. Faire progresser une tâche
3. Quitter

Choix ? 2

Tâche à faire progresser ? lire

A FAIRE :

- Coder les fonctions indispensables
- Réaliser les tests unitaires
- Coder la fonction main

EN COURS :

- Lire l'énoncé de l'examen

TERMINÉE :

(vide)

MENU

1. Ajouter une tâche
2. Faire progresser une tâche
3. Quitter

Choix ? 2

Tâche à faire progresser ? lire

A FAIRE :

- Coder les fonctions indispensables
- Réaliser les tests unitaires
- Coder la fonction main

EN COURS :
(vide)

TERMINÉE :
- Lire l'énoncé de l'examen

MENU

1. Ajouter une tâche
2. Faire progresser une tâche
3. Quitter

Choix ? 2

Tâche à faire progresser ? lire

Cette tâche ne peut plus progresser !

MENU

1. Ajouter une tâche
2. Faire progresser une tâche
3. Quitter

Choix ? 2

Tâche à faire progresser ? codr

Cette tâche n'existe pas !

MENU

1. Ajouter une tâche
2. Faire progresser une tâche
3. Quitter

Choix ? 2

Tâche à faire progresser ? coder

A FAIRE :

- Réaliser les tests unitaires
- Coder la fonction main

EN COURS :
- Coder les fonctions indispensables

TERMINÉE :
- Lire l'énoncé de l'examen

MENU

```
1. Ajouter une tâche
2. Faire progresser une tâche
3. Quitter
Choix ? 4
```

```
Choix incorrect !
```

```
MENU
1. Ajouter une tâche
2. Faire progresser une tâche
3. Quitter
Choix ? 3
```

```
Fin du programme.
```