DIJON AUXERRE NEVERS
UNIVERSITÉ DE BOURGOGNE

Lors de ce TP, vous allez poursuivre le développement d'une application, en ajoutant différentes fonctionnalités, tout en maintenant les fonctionnalités valides.

TP n°6

Poursuite d'un développement

Alexandre Guidet

Compétence 1 : « Réaliser »	Compétence 5 : « Conduire »
☒ Implémenter des conceptions simples ☐ Élaborer des conceptions simples ☒ Faire des essais et évaluer leurs résultats au regard des spécifications ☐ Développer des interfaces utilisateurs	☐ Appréhender les besoins du client et de l'utilisateur ☒ Mettre en place les outils de gestion de projet ☐ Identifier les acteurs et les différentes phases d'un cycle de développement

R2.03

Qualité de développement

Une application, `ChercheMots`, a été développée pour aider à trouver des mots dans un dictionnaire, par exemple pour assister dans les jeux de mots, pour aider à résoudre des énigmes…

L'application est totalement fonctionnelle mais ne présente que deux fonctionnalités :

- Donner la définition d'un mot
- Donner les anagrammes d'un mot

Comme pour le TP n°4, le travail se déroule en **équipe** de **TP**.

INITIALISATION

Pour chaque fonctionnalité, il faut impérativement respecter la conception réalisée par l'architecte/concepteur (fournie dans le fichier conception.vpp, au format Visual Paradigm, ou visible au format HTML dans le dossier commun). Les parties « métier » des fonctionnalités doivent être correctement testées par des tests unitaires, toutes les opérations et classes non privées doivent être documentées correctement. Il ne faut pas « pousser » sur le dépôt GitHub un projet qui ne compile pas (faites bien attention aux fusions).

> ⓘ *Chaque fonctionnalité doit utiliser une **branche distincte** de main.*

Les différentes fonctionnalités sont décrites ci-après. Chaque fonctionnalité possède un coefficient de difficulté et/ou de durée. **Connectez**-vous dans Visual Studio au dépôt GitHub du projet. Commencez par **étudier** les fonctionnalités existantes (lire la documentation, la conception, essayer le programme). Remarquez que, normalement, tous les tests unitaires passent. Choisissez une branche (donc une fonctionnalité). **Placez-vous DANS cette branche dans Visual Studio !**

Commencez le **codage** de votre fonctionnalité. N'oubliez pas de vous **synchroniser** avec les autres membres de votre équipe (**travaillez bien dans votre branche et non dans le master**).

Dès qu'une équipe a implémenté une fonctionnalité (codage fini, les tests passent tous, le fonctionnement est total), elle doit le **signaler** au chef de projet (l'enseignant de TP) qui validera (ou non) la fonctionnalité et proposera à l'équipe une nouvelle fonctionnalité à implémenter.

> ⓘ *Quand une fonctionnalité est finalisée, l'enseignant va **fusionner** la branche dans la branche principale (main). La prochaine branche sera créée par rapport à main (après la fusion).*

N'oubliez pas d'indiquer dans la documentation **l'auteur** de la fonction en utilisant la balise XML :

```
/// <author>prénom nom</author>
```

LISTE DES FONCTIONNALITES SOUHAITEES

LISTER LES MOTS COMMENÇANT PAR UN PREFIXE (COEF 2)

Le but de cette fonctionnalité est de permettre à l'utilisateur de saisir un préfixe (quelques lettres), et le programme lui indiquera tous les mots du dictionnaire qui commencent par ces lettres.

Cette fonctionnalité correspond au cas d'utilisation « chercher les mots à partir d'un préfixe » du diagramme de cas d'utilisation fourni par la conception.

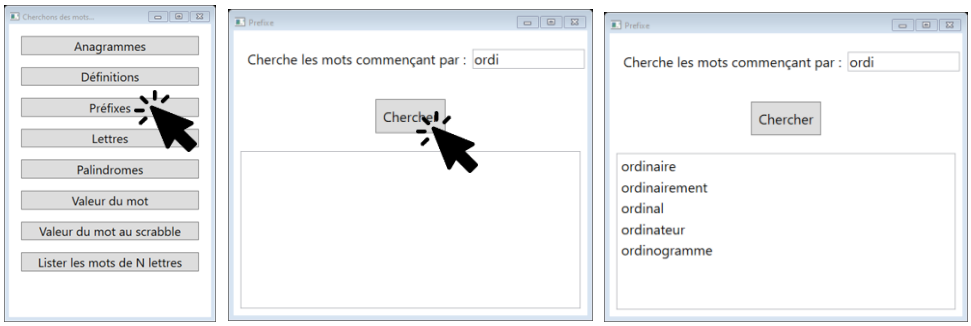
La conception fournie comprend la maquette de l'écran souhaité (`PrefixeWindow` maquette). Le diagramme de classe « ihm » et le diagramme de classes « métier » comprennent les classes nécessaires pour réaliser ce travail.

Le diagramme de séquence correspondant décrit les opérations à implémenter.

Cette fonctionnalité revient donc à :

- Créer une fenêtre `PrefixeWindow` en respectant la maquette fournie
- Créer une opération `Prefixes` dans la classe métier `Dictionnaire` qui cherche les mots correspondants
- Créer un test unitaire pour l'opération métier
- Coder dans `PrefixeWindow` la séquence d'opérations fournie
- Modifier `MainWindow` pour permettre de lancer la séquence totale

Exemple de fonctionnement attendu :



TROUVER LES MOTS QUI CONTIENNENT LES LETTRES DEMANDEES (COEF 3)

Le but est de lister, dans le dictionnaire, les mots qui contiennent les lettres demandées, en laissant la possibilité de se limiter ou non à ces lettres.

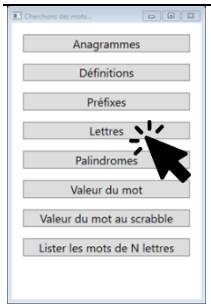
Cette fonctionnalité correspond au cas d'utilisation « trouver à partir des lettres ».

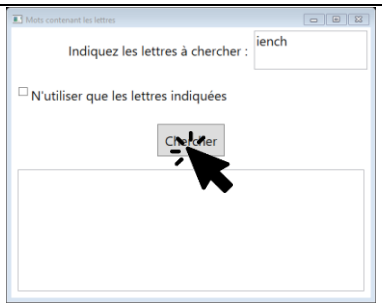
La conception fournit une maquette (`LettresWindow` maquette) pour la fenêtre à créer. Le diagramme de séquence correspondant décrit les opérations à implémenter.

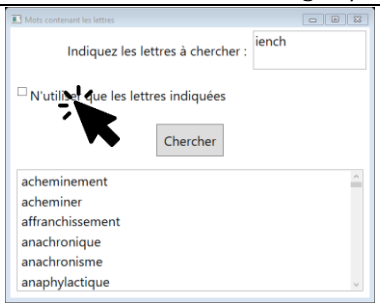
Cette fonctionnalité revient donc à :

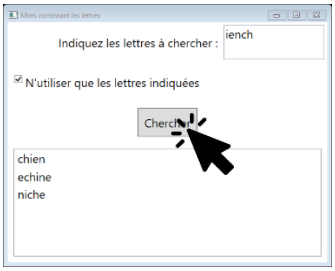
- Créer une fenêtre `LettresWindow` qui implémente la maquette
- Ajouter dans `MainWindow` le nécessaire pour lancer la séquence
- Créer dans la classe Dictionnaire une opération `MotsAvecLettres` qui réalise le travail
- Créer un test unitaire pour vérifier l'opération `MotsAvecLettres`
- Coder le déroulement de la séquence dans la classe `LettresWindow`

Exemple de fonctionnement attendu :









TROUVER LES PALINDROMES (COEF 2)

Le but de cette fonctionnalité est d'afficher les différents **palindromes** du dictionnaire (les mots qui se lisent dans les deux sens).

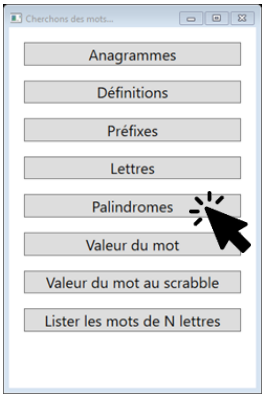
Cette fonctionnalité correspond au cas d'utilisation « trouver les palindromes ».

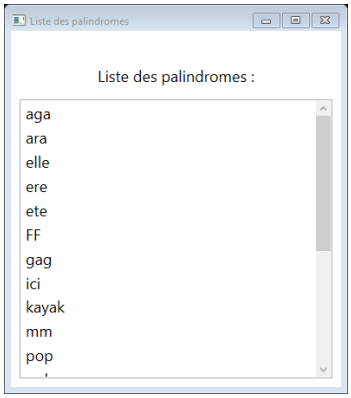
Un diagramme de séquence décrit la totalité de la fonctionnalité.

Le travail à faire est donc :

- Créer une fenêtre `PalindromesWindow` qui implémente la maquette
- Ajouter dans la fenêtre principale le nécessaire pour lancer la fonctionnalité
- Ajouter dans la classe `Dictionnaire` l'opération de classe `EstPalindrome` qui indique si un mot est un palindrome
- Ajouter dans la classe `Dictionnaire` l'opération `Palindromes` qui donne la liste des palindromes du dictionnaire
- Créer les tests unitaires pour les opérations `EstPalindrome` et `Palindromes`
- Intégrer dans `PalindromesWindow` le code nécessaire pour dérouler la séquence

Exemple de fonctionnement attendu :





CALCULER LA VALEUR D'UN MOT (COEF 1)

Dans de nombreux jeux (notamment d'énigmes) il faut convertir un mot en nombre. En général, la conversion est faite avec A=1, B=2, etc. puis en faisant la somme des lettres du mot.

Cette fonctionnalité correspond au cas d'utilisation « calculer la valeur du mot ».

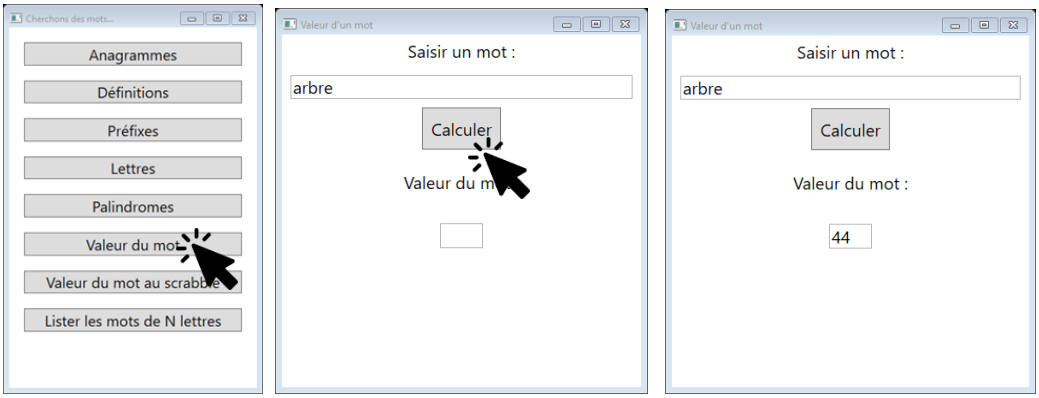
La maquette est définie dans la conception (`ValeurWindow` maquette).

La conception contient un diagramme de séquence qui décrit la fonctionnalité.

Cette fonctionnalité revient donc à :

- Créer une fenêtre `ValeurWindow` qui implémente la maquette
- Ajouter dans la classe `Dictionnaire` une opération de classe `Valeur` qui calcule la valeur
- Ajouter un test unitaire pour l'opération `Valeur`
- Intégrer l'utilisation de la classe `Dictionnaire` dans la fenêtre `ValeurWindow`
- Modifier la fenêtre principale pour lancer la séquence

Exemple de fonctionnement attendu :



CALCULER LA VALEUR D'UN MOT AU SCRABBLE (COEF 1)

Ce type de logiciel peut facilement être utilisé dans les jeux de lettres type Scrabble : il est donc pertinent de rajouter une fonctionnalité pour un tel calcul.

Cette fonctionnalité correspond au cas d'utilisation « calculer une valeur scrabble ».

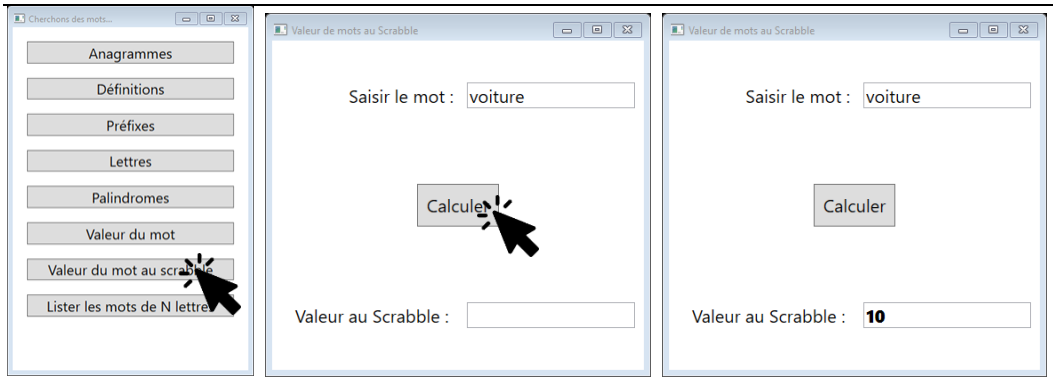
La conception précise la maquette (`ScrabbleWindow maquette`) ainsi qu'un diagramme de séquence détaillant la fonctionnalité.

La fonctionnalité revient à :

- Créer une fenêtre `ScrabbleWindow` qui implémente la maquette
- Rajouter une opération de classe `ValeurScrabble` dans la classe `Dictionnaire` qui calcule cette valeur
- Réaliser un test unitaire pour l'opération `ValeurScrabble`
- Modifier la fenêtre principale pour lancer la séquence
- Lier le code IHM au code métier (suivant la séquence) dans `ScrabbleWindow`

> ⓘ *La valeur des lettres au Scrabble, en français, est la suivante : Lettres du Scrabble — Wikipédia (wikipedia.org)*

Exemple de fonctionnement attendu :



LISTER LES MOTS SUIVANT LEUR NOMBRE DE LETTRES (COEF 3)

Il peut être intéressant de se connaitre les mots de 4 lettres, par exemple. Cette fonctionnalité permet d'extraire du dictionnaire les mots contenant un certain nombre de lettres.

La fonctionnalité correspond au cas d'utilisation « Lister mots n lettres » et est décrite dans le diagramme de séquence correspondant.

Une maquette (`MotsNWindow maquette`) décrit l'aspect souhaité pour la fenêtre.

Réaliser la fonctionnalité revient à :

- Créer une fenêtre `MotsNWindow` qui implémente la maquette
- Créer une classe `NbLettresIncorrect` qui est une exception de la couche métier
- Créer dans la classe `Dictionnaire` une fonction `Lister` qui liste les mots de n lettres (cette fonction doit lever `NbLettresIncorrect` si n est inférieur à 2).
- Créer un test unitaire pour la fonction `Lister`
- Intégrer l'appel du code métier dans le code IHM de `MotsNWindow`
- Modifier la fenêtre principale pour lancer la séquence

Exemple de résultat attendu :