# **Interactive Java Book**

31.12.2015

#### Blondel Kevin / Pico Christopher

## Responsabilité des classes

<u>Server</u>: Lance le serveur ainsi que les instance des classes necessaire pour le bon fonctionnement du programme.

<u>Exercise</u>: Definit un exercice ainsi que ses methodes.

Exercises: Listes des exercices avec des methode de liste.

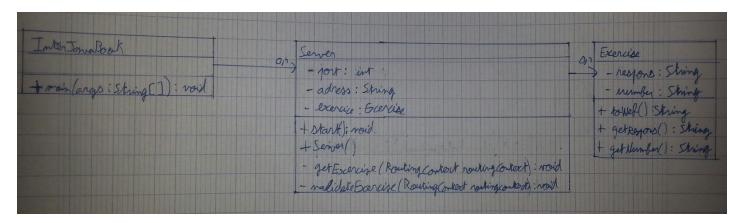
<u>InterJavaBook</u>: main du programme.

<u>Validation</u>: Permet de accepter/verifier/valider une reponse donnée par l'utilisateur.

Watcher: Surveille si un exercice dans le dossier exercice est modifié.

<u>Test Exo</u>: Permet de lancer les fonctions de test des exercices, l'accès à cette classe se fait uniquement par reflection.

### **Diagrammes UML**



Dans un premier temps nous avons pensé à utiliser un exercice comme un numero et une réponse sous forme de string mais les exercices ont très rarement une réponse sous forme de string, nous avons donc changer pour en faire une liste de méthode.

ensuite on a implementer le watcher pour les modifications d'exercice pendant l'utilisation du serveur et Exercises qui sera l'utilisation complete de tout les Exercices.

enfin Validation qui permettra de valider un code java sous forme de String et renvoyer si la réponse est bonne ou non.

- condition: Condition - 30: 55 kall - injurt: Mung - reflect : int - part: int - part: int - part: int - waldon: Walkbernick - walkbernick - walkbernick - walkbernick - pool: Exercises - get Exercises - get Exercises - get Exercises - realidate Exercise ( Routing Context nowing Context) void - folden: String - walkbernick ( Routing Context nowing Context) void - folden: String - walkbernick ( Routing Context nowing Context): void + Services ( folden: String - walkbernick ( folden: String - get Another Translog - respons: Cist < Mother - respons: Cist < Mother - respons: Clat < Moth	
- "Fraid (injut: String, methoditext: I  The Time Book of Server 1; Wather - waster: Wather waster: Wather -	
Type Time Book 27 Somen 1; Walther - part: int - part: int - walther: Walther red (part: Shing) - watcher: String - watcher: Walther + watcher(part: Shing) - watcher: Walther + restrodify (): Boollem + restrodify (): Boollem - exercises: Exercises + stant(): you'd - get Exercises Routing Contact noutling Contact) wou'd - exercises: List < Exercises - wall at the Exercise (Routing Contact noutling Contact) wou'd + Exercises (folder: Shing - update Eile (Routing Contact noutling Contact): wou'd + Exercises (folder: Shing + Sover (String folder) - get till troscuo (): wo + get to Wef Exercises (1) *  **Tember: Shing folder 1) **Tember: Shing - folder: Shing - remains (interpretable)	
- part: int - walker: Walkern - adress: String - adress: String - adress: String - walker: Walker - watcher: Walker - carcises: List (Exercises - get Exercises Routing Context routing Context) would - carcises: List (Exercises Routing Context routing Context) would - folder: String - update File Routing Context routing Context) would + Exercises (folder: String - get tell trusters (string folder) - get tell trusters (string folder) - get tell trusters (string folder) - get to Wef Trombay (string folder: String - folder: folder: String - folder: fo	yt < Methods): String
- part: int - walker: Walkern - adress: String - adress: String - adress: String - walker: Walker - watcher: Walker - carcises: List (Exercises - get Exercises Routing Context routing Context) would - carcises: List (Exercises Routing Context routing Context) would - folder: String - update File Routing Context routing Context) would + Exercises (folder: String - get tell trusters (string folder) - get tell trusters (string folder) - get tell trusters (string folder) - get to Wef Trombay (string folder: String - folder: folder: String - folder: fo	
- advers: String + intitle (jath: String) - watcles: Watcher + restrodify (): footless - ecs: Exercises + stant(): void - get Exercise (Routing Context nouting Context invoid - exercises: List < Exercises - ralidate Exercise (Routing Context nouting Context) void - folder: String - update File (Routing Context nouting Context): void + Exercises (folder: String + Server (String folder) + Get Hostuse Trembey ( - muncles: String - holder: String - holder: String - restress: (int < Mother)	
- watchen: Watchen  - exis: Exercises  + stant(): proid  - get Exercise Routing Context nouting Context invoid  - exercises: List < Exercises  - ralidate Exercise (Routing Context nouting Context): void  - folder: String  - uplate File (Routing Context nouting Context): void  + Exercises (folder: Str  + Server (String folder)  - get Another Trombay (  - munder: String  - munder: String  - folder: String  - folder: String  - folder: String	
+ stant(): poid.  - gett Ecercise (Routing Context nouting Context): void - exercises: List (Exercise)  - ralidate Exercise (Routing Context nouting Context): void - folder: String - update Eile (Routing Context nouting Context): void + Exercises (folder: String + Server (String folder) - gettell tracuse (): void + get Answer Frankley (): void + get Answer Frankley (): void + get Tower Frankley (): void (): voi	
- get Ecercise Routing Context nouting Context is rout - ocercises: List < Exercises  - ralidate Exercise (Routing Context nouting Context): Novid - folder: String - light Context nouting Context : Novid + Exercises (folder: String + Server (String folder) - get Another Frontieg (- get Another Frontieg (- runled: String - runled: String - folder: String - runled: String - folder: String - folder: String - runled: String - folder: String - runled: runled: String - runled: String - runled: String - runled: String	
- get Ecercise (Routing Context nouting Context) is vaid - cercises: List (Exercise nating Context) is vaid - folder: String - update File (Routing Context nouting Context): vaid + Exercises (folder: String + Server (String folder) - get Answer Frankley (): void + get Answer Frankley (): void + get Answer Frankley (): void + get To Wef Frankley (): void ()	
- ralidate Exercise (Routing Contact nouting Contact): Novid - folder: String - welder: String folder: String folder: String folder: String folder: String folder: String folder: String - get Avoise Frontieg (- munder: String - folder: String - folder: String - folder: String - passence (int & hathad)	de)
+ Server (String folder) - gettell trisewar (): voo + get trisewar tromkey () - bro () = Get Nise (): voor tromkey (): voor	
+ Server (String folder) - gettell trisewar (): vo + get trowker From key ()  - Boo of Goe Mise 11* + get To Wef From key (  - number: String - folder: String -	4
- Exercise 11* + get To Wef Fronting ( -munsel : String - folder: String - paragraph ( list & Matthed )	1
- bxo (1 Coercise 1)* + get To Wef Tronkey (- munder: String) - folder: String - nandom: (list & Matthed)	ry: String): list Method
- number: String - folder: String -	ry: String String
- folder: String	
- nanomi (ist & hathad)	
the state of the s	
to the following which will be to the first the state of	ca l
~ tower(): String	
v Jetkerpons (): List < Method >	
v gershmero(); String	

#### Voici le diagramme final du projet :

lci server est la classe principale du projet, il recoit les informations depuis localhost pour pouvoir analyser et repondre en conséquence.

Il utilise la classe watcher pour les modification de fichier, Exercises pour la liste des exercice, et validation pour valider la String passer par l'utilisateur dans le localhost.

Exercises utilise Exercice qui est créer grace a la classe Test\_Exo qui contient toute les méthode de test pour l'exercice demander, par exemple pour l'exo5 il y a une ou plusieurs method test\_5(...) qui renverra obligatoirement un boolean et qui testera la validiter de ce qu'a rentrer l'utilisateur.

Create\_Exercise remplace le constructeur de Exercise car il n'a pas les même champs, il utilise donc la class passé en argument pour créer sa liste de méthode. la classe mise en argument est obligatoirement Test\_Exo, je l'ai donc noté dans le diagramme pour mieux comprendre comment ca fonctionne.