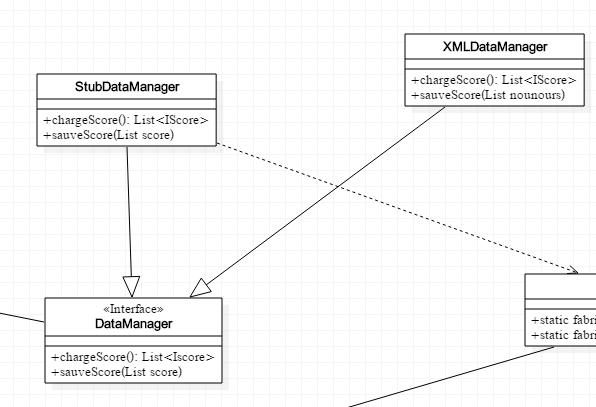
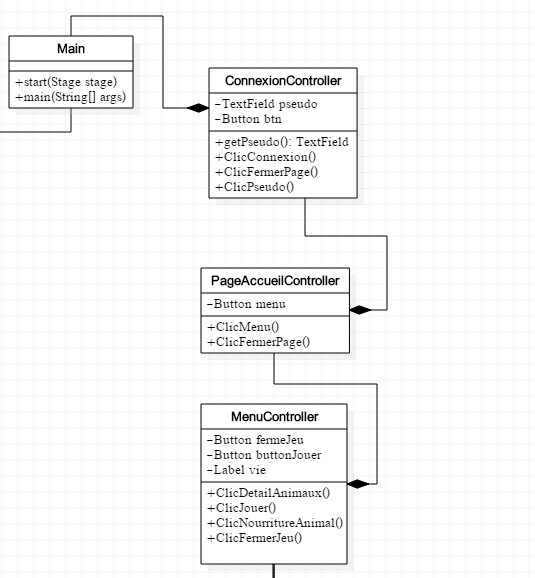
**DOCUMENTATION : 0/20**

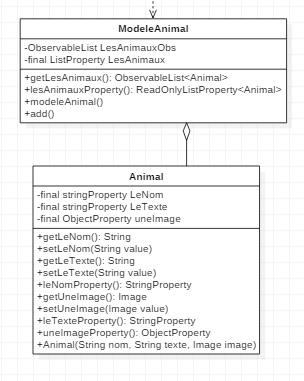
* Je sais concevoir un diagramme UML intégrant des notions de qualité et correspondant exactement à l’application que j’ai à développer. [sur 7 points]  
  **PREUVE :**  
  **Nous avons fait le diagramme Uml de notre application dans le fichier JavaFXDoc de notre git . Nous avons répertorié nos classes, qui se compose de leurs méthodes et de leurs attributs. Nous avons utilisé dans** **ce diagramme :**

**des héritages : une relation de spécialisation/généralisation.**

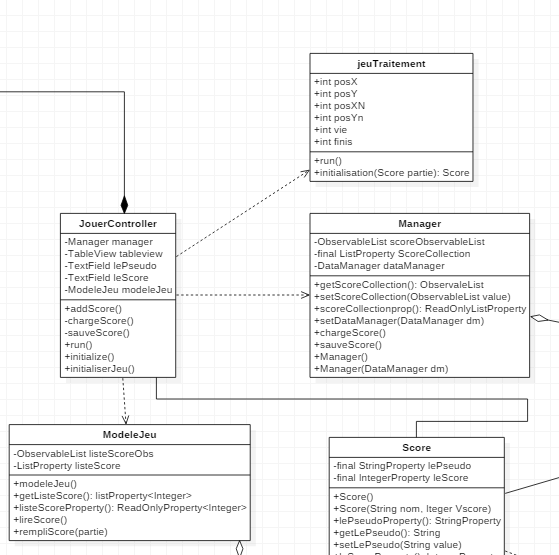
**La classe fille hérite de touts les attributs et méthodes.**

**des compositions :** 

**Si la classe mère disparaît celle qu’elle compose disparaît avec elle**

**des agrégations :**

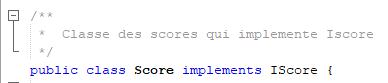
**Une classe se compose d’une autre classe**

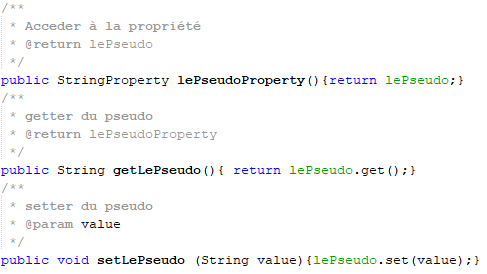
 **des associations :**

[**P**](https://fr.wikipedia.org/wiki/Association_(programmation))**ermet  est une connexion de lien logique entre deux classes**

**des dépendances : Une ou plusieurs méthodes reçoivent un objet d'un type d'une autre classe**

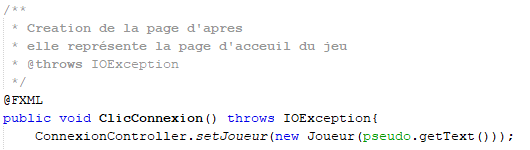
***=> 0/7***

* Je sais décrire un diagramme UML en mettant en valeur et en justifiant les éléments essentiels. [sur 3 points]  
  **PREUVE :**  
  ***Pas encore   
  => 0/3***
* Je sais documenter mon code et en générer la documentation. [sur 2 points]  
  **PREUVE :**  
  **Chacune de nos fonctions** **à une description rapide de son contexte et de** **sa fonctionnalité. Nous avons utilisé les notions de la javadoc pour documenter notre code,** **c'est-à-dire avec l’utilisation :**

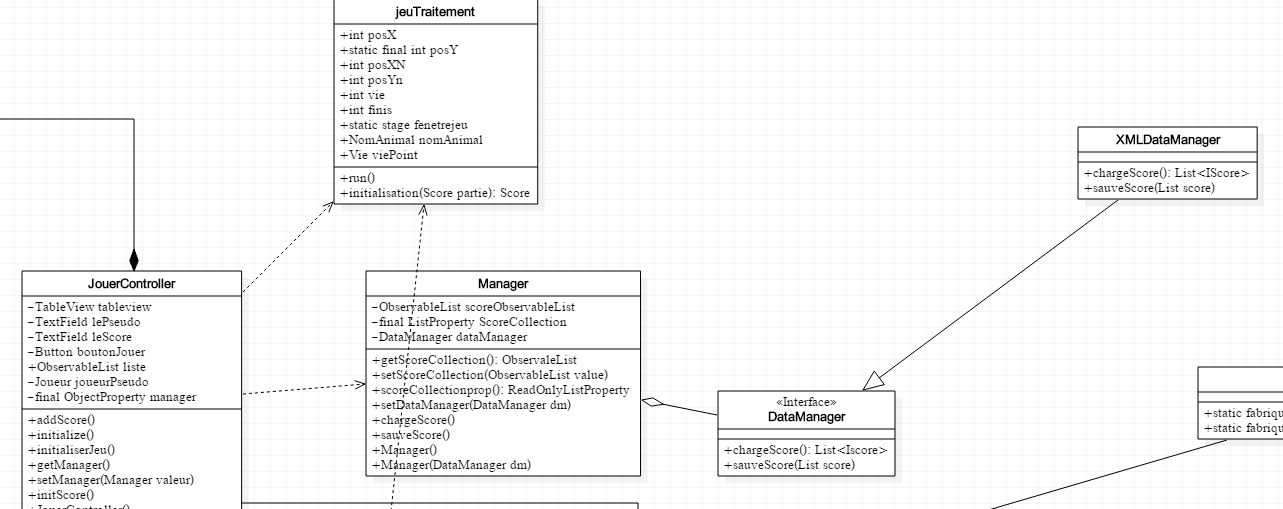


**Des return : elle permet de montrer se que return la fonction**

**Des param : elle permet de montrer se que la méthode reçoit en paramétré**

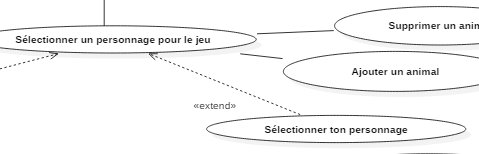
**Des throws :** **elle permet de montrer qu’il y a une utilisation des exceptions *=> 0/2***

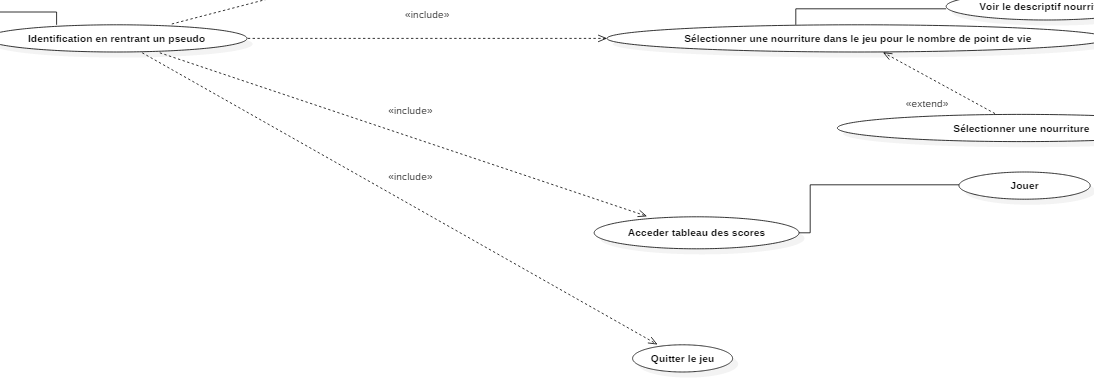
* Je sais justifier pourquoi j’ai utilisé tel ou tel patron de conception dans un projet. [sur 4 points]  
  **PREUVE :**  
  **Nous avons utilisé un patron stratégie : car il résout le problème du changement d’****algorithmes à la volée et il permet de choisir l’algorithme en fonction de l’instance passé en paramètre.**

****

***=> 0/4***

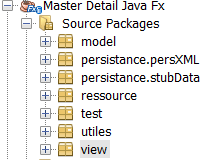
* Je sais décrire le contexte de mon application, pour que n’importe qui soit capable de comprendre à quoi elle sert. [sur 2 points]  
  **PREUVE :**  
  **Nous avons fait un fichier sur le contexte de notre application en décrivant ses fonctionnalités et son affichage. Ce fichier est dans le dossier** **javafxdoc. Le contexte sera intégré au** **PDF** **avec toute la documentation demandée.  *=> 0/2***
* Je sais faire un diagramme de cas d’utilisation pour mettre en avant les différentes fonctionnalités de mon application. [sur 2 points]  
  **PREUVE :**  
  **Nous avons fait un diagramme de cas d’utilisation qui se trouve dans le dossier** **javafxdoc. Il montre les différentes fonctionnalités de l’application et les étapes de l’application. Nous avons utilisé :**

**Des extend : Elle permet de montrer que le cas d’utilisation du début de la flèche est obligatoire a effectué pour aller au cas d’utilisation de l’indication de la flèche. Exemple : le cas d’utilisation <<sélectionner ton personnage>> doit être fait avant de pouvoir faire le cas d’utilisation <<sélectionner un personnage pour le jeu>>.**

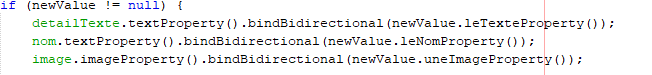
**Des Include : Elle permet de montrer que le cas d’utilisation d’où part la flèche sera obligatoirement effectué avant celui qui a l’indication de la flèche. Exemple : le cas d’utilisation du début de la flèche <<identification pseudo>> contiendra toujours le cas d’utilisation <<sélectionner un personnage>>.  *=> 0/2***

**Programmation : 0/40**

* Je maîtrise la notion d’immuabilité de la classe String. [sur 0.5 point]  
  **PREUVE :**  
  ***Pas encore  
  => 0/0.5***
* Je maîtrise les règles de nommage Java. [sur 1 point]  
  **PREUVE :**

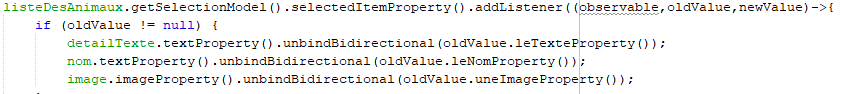
**Nous avons utilisé des packages selon leurs fonctionnalités.** **Nous avons le** **package-modèle : il permet de mettre nos modèles qui** **dépendent du** **contrôleur. Nous** **a le package persistance : il permet de mettre toutes les** **classes en rapport** **à la persistance** **c'est-à-dire le chargement et la sauvegarde du pseudo et du score du joueur. Nous avons le package** **ressource : il permet de mettre les images et les sons utilisés et** **les pages FXML et CSS. Nous avons le package** **test : il contient notre main de l’application. Nous avons le package** **utiles : il permet de mettre toute** **la classe utile** **c'est-à-dire celle qu’on utilise pour la création d’objet. Nous utilisons le package view : il permet de mettre** **tous nos contrôleurs de page.**

**Chaque fichier ne contient qu’une seule classe ou interface. Nous avons fait de la documentation pour chacune de nos méthodes. Nous avons mis des commentaires en utilisant les règles de la javadoc comme expliquer précédemment. Il ne doit pas y avoir d'espaces entre le nom d'une méthode et sa parenthèse ouvrante. L'accolade ouvrante qui définit le début du bloc de code doit être à la fin de la ligne de déclaration . L'accolade fermante doit être sur une ligne séparée dont le niveau d'indentation correspond à celui de la déclaration. La déclaration d'une méthode est précédée d'une ligne blanche. *=> 0/1***

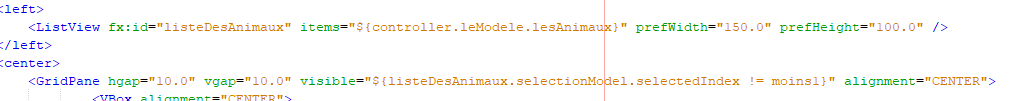
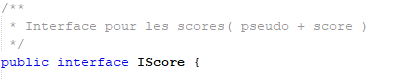
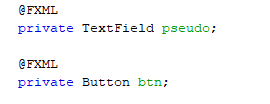
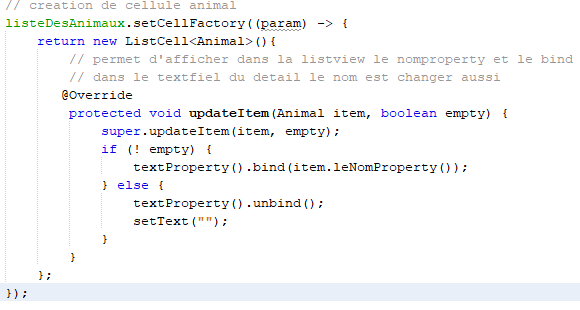
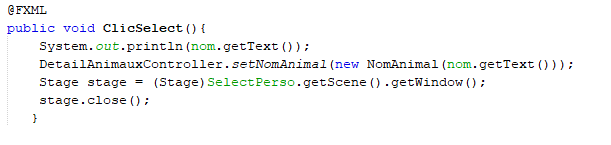
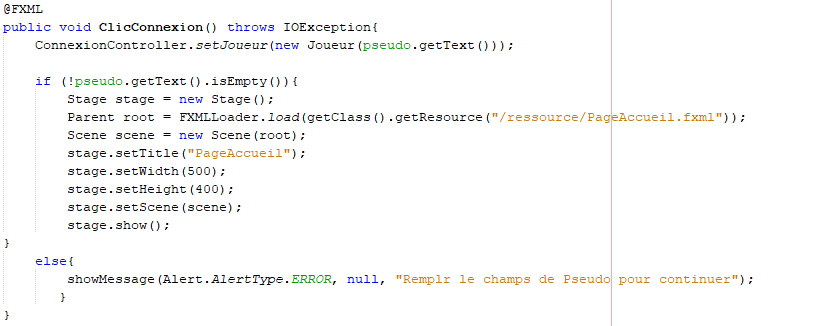
* Je sais binder bidirectionnellement deux propriétés JavaFX. [sur 1 point]  
  **PREUVE :**  
  **Liaisons dans les deux sens. Si la propriété A et la propriété B sont liées entre elles par une liaison bidirectionnelle, alors tout changement de valeur de la propriété A se répercute automatiquement sur la propriété B ; mais tout changement de valeur de la propriété B se répercute automatiquement sur la propriété A également. Exemple : quand nous modifions quelque chose dans la liste des animauxelle se répercute automatiquement dans le détail de la liste et inversement.**

**Liaisons dans un sens. Si la propriété A et la propriété B sont liées entre elles par une liaison bidirectionnelle, alors tout changement de valeur de la propriété A se répercute automatiquement sur la propriété B ; mais tout changement de valeur de la propriété B ne se répercute pas automatiquement sur la propriété A**

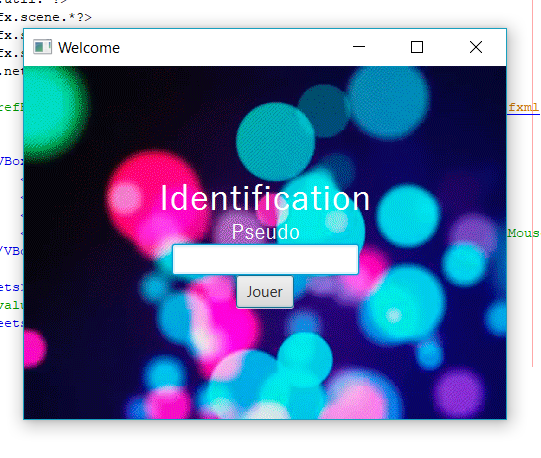
***=> 0/1***

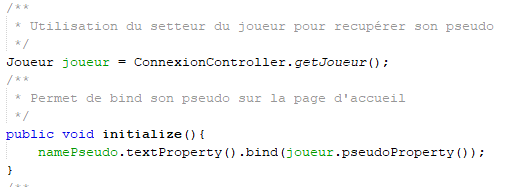
* Je sais binder unidirectionnellement deux propriétés JavaFX. [sur 1 point]

**PREUVE :**  
**Liaisons dans un sens. Si la propriété A et la propriété B sont liées entre elles par une liaison bidirectionnelle, alors tout changement de valeur de la propriété A se répercute automatiquement sur la propriété B ; mais tout changement de valeur de la propriété B ne se répercute pas automatiquement sur la propriété A *=> 0/1***

* Je sais coder une classe Java en respectant des contraintes de qualité de lecture de code. [sur 1 point]  
  **PREUVE :**  
  **Chaque fichier ne contient qu’une seule classe ou interface. Nous avons fait de la documentation pour chacune de nos méthodes. Nous avons mis des commentaires en utilisant les règles de la javadoc comme expliquer précédemment. Il ne doit pas y avoir d'espaces entre le nom d'une méthode et sa parenthèse ouvrante. L'accolade ouvrante qui définit le début du bloc de code doit être à la fin de la ligne de déclaration . L'accolade fermante doit être sur une ligne séparée dont le niveau d'indentation correspond à celui de la déclaration. La déclaration d'une méthode est précédée d'une ligne blanche. *=> 0/1***
* Je sais contraindre les éléments de ma vue, avec du binding FXML. [sur 1 point]  
  **PREUVE :**  
  **Nous avons bind des objets en FXML pour le master détail,****c'est-à-direentre la liste** **àgauche et les éléments de droite, les objets sont** **bindsgrâce** **àla balise items. *=> 0/1***
* Je sais définir et utiliser une classe abstraite. [sur 0.5 point]  
  **PREUVE :**  
  ***Pas encore  
  => 0/0.5***
* Je sais définir et utiliser une interface. [sur 0.5 point]  
  **PREUVE :**  
  **Nous utilisons une interface** **Iscore qui implémente Score  
  Elle permet d’avoir accès au getter et setteur du pseudo et du score  *=> 0/0.5***
* Je sais définir un attribut de classe. [sur 0.5 point]  
  **PREUVE :**  
  **Dans la plupart** **de nos classesnous avons des attributs. Nous les avons déclaré pour pouvoir les** **utiliserdans la suite du code. Il faut donner sa visibilité, son type puis son nom. *=> 0/0.5***
* Je sais définir une CellFactory fabriquant des cellules qui se mettent à jour au changement du modèle. [sur 2 points]  
  **PREUVE :**  
  **Elle est utilisée pour produire une cellule destinée à afficher chaque valeur visible de la liste. Nous l’utilisons pour l’affichage des valeurs de la liste view** **dans nos deux masters détails** **c'est-à-dire celui de la sélection de personnage et celui de la sélection de la nourriture.  *=> 0/2***
* Je sais définir une méthode de classe. [sur 0.5 point]  
  **PREUVE :**  
  **Dans quasi toutes nos classes nous utilisons des méthodes. *=> 0/0.5***
* Je sais définir une variable ou un attribut constant. [sur 0.5 point]  
  **PREUVE :**  
   ***=> 0/0.5***
* Je sais définir, attraper et laisser passer une exception. [sur 1 point]  
  **PREUVE :**  
  **Nous utilisons une exception car quand nous arrivons sur cette page, il faut vérifier si le pseudo a bien été rentré. c’est pourquoi nous levons une** **exceptionsi aucun pseudo n’est rentré. Donc nous l’avons défini comme sur l’image. Nous l’attrapons lorsque aucun pseudo n’est écrit et nous la laissons passer lorsqu’un pseudo est écrit. *=> 0/1***
* Je sais développer ma propre interface pour qu’elle soit utilisée en lambda-expression. [sur 0.5 point]  
  **PREUVE :**  
  ***Pas encore  
  => 0/0.5***
* Je sais développer une application graphique en JavaFX en utilisant FXML. [sur 1 point]  
  **PREUVE :**

**Nous avons fait** **toutes nos interfaces graphiques en FXML dans l’application sauf l’application de jeu qui passe directement sur du code java.** **L’image de code est du FXML et celui-ci permet d’afficher l’interface que vous voyez ici.**

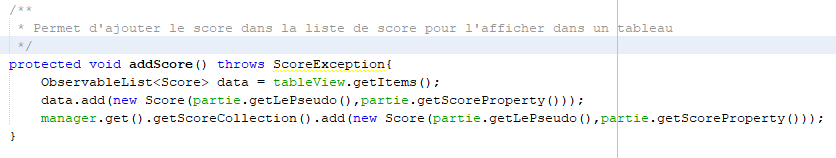
***=> 0/1***

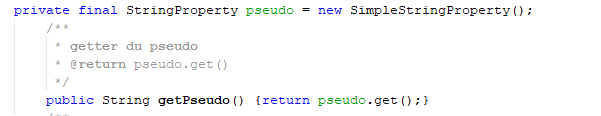
* Je sais éviter la duplication de code. [sur 1 point]  
  **PREUVE :**  
  ***Pas encore  
  => 0/1***
* Je sais hiérarchiser mes classes pour spécialiser leur comportement. [sur 2 points]  
  **PREUVE :**  
  **Nous avons fait des packages comme nous l’avons dit dans une explication précédente. C’est pourquoi****ce package montre la hiérarchisation car chaque package regroupe une hiérarchie de fonctionnalité. *=> 0/2***
* Je sais intercepter des évènements en provenance de la fenêtre JavaFX. [sur 2 points]  
  **PREUVE :**  
  **Nous avons** **interceptéun objet** **c'est-à-diredans la page d’identification. Le joueur doit** **rentrerson pseudo. Nous l’interceptons car nous** **avons une récupérationdans la classe pour le garder en mémoire. Nous utilisons donc un évènement en provenance de la fenêtre connexion. Pour pouvoir la** **réutiliserensuite sur la fenêtre suivante pour réécrire le pseudo qui a été récupéré. Exemple : sur l’image de code c’est l’affichage du pseudo qui a été écrit dans la fenêtre d’avant. *=> 0/2***
* Je sais maintenir une encapsulation adéquate dans mes classes. [sur 2 points]  
  **PREUVE :**  
  **Elle permet de définir des niveaux de visibilité des éléments de la**[**classe**](http://www.commentcamarche.net/contents/807-classe-et-instance-d-objet)**. Ces niveaux de visibilité définissent les droits d'accès aux données selon que l'on y accède par une méthode de la classe elle-même.**

**publique: les fonctions de toutes les classes peuvent accéder aux données ou aux méthodes.**

**privée: l'accès aux données est limité aux méthodes de la classe elle-même.**

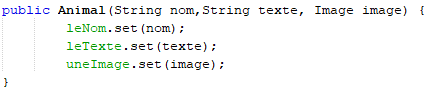
**protected : l'accès aux données est réservé aux fonctions des classes qui hérite**

****



***=> 0/2***

* Je sais maintenir, dans un projet, une responsabilité unique pour chacune de mes classes. [sur 2 points]  
  **PREUVE :**

**Nous utilisons dans notre projet des classe avec une seule responsabilité comme par exemple animal : qui a la fonctionnalité de créer un animal.**

***=> 0/2***

* Je sais gérer la persistance de mon modèle. [sur 2 points]  
  **PREUVE :**

**Nous avons essayer de sérialiser notre jeu mais nous sommes pas arriver au bout de cette persistance car nous avons eu des problèmse que nous savions pas résoudre. *=> 0/2***

* Je sais surveiller l’élément sélectionné dans un composant affichant un ensemble de données. [sur 2 points]  
  **PREUVE :**  
  **Grâce à notre master détail nous avons** **surveillé l’élément sélectionné dans un composant car quand le joueur clic sur un élément de la liste.** **L'élément averti qu’il a été sélectionné. c’est pourquoi il affiche ensuite le détail sur la partie droite.  *=> 0/2***
* Je sais transtyper. [sur 0.5 point]  
  **PREUVE :Par défaut les entiers sont typés en Integer mais un suffixe peut les spécifier long*.***

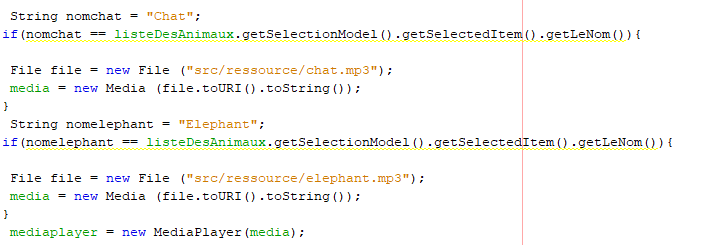
***=> 0/0.5***

* Je sais utiliser à mon avantage le polymorphisme. [sur 2 points]  
  **PREUVE :**

**Nous utilisons du polymorphisme dans le fait : quand nous déclarons une liste nous la déclaront en générale en ObservableListe mais son type est une Observable ArrayList**

 ***=> 0/2***

* Je sais utiliser certains composants simples que me propose JavaFX. [sur 0.5 point]  
  **PREUVE :**  
  **Nous utilisons dans notre master détail une image avec ObjectProperty<Image>. Elle permet de définir le type pour l’affichage de l’image. Nous utilisons aussi des médias pour mettre le son du bouton qui est dans le master détail des animaux.**

******

***=> 0/0.5***

* Je sais utiliser certains layout que me propose JavaFX. [sur 0.5 point]  
  **PREUVE :**  
  **Nous utilisons pour former** **nos fenêtres un border pane car il permet de modéliser la page** **en 5 parties c'est-à-dire droites, gauche, top, bottom, et** **centre. Cela nous paraissait le plus simple** **à** **utiliser pour afficher nos balises. Nous avons utilisé des** **Hbox et de** **Vbox pour faire un meilleur affichage entre nos différente partie. Nous avons aussi utilisé une gridpane pour mettre** **des dispositions de ligne et de colonne. Nous avons utilisé un** **tablview pour mettre** **un affichage de colonne et de ligne.**

******

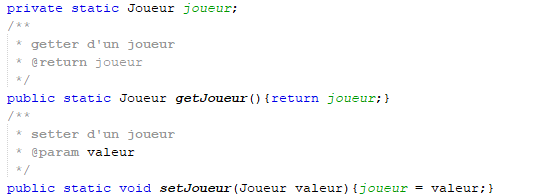
***=> 0/0.5***

* Je sais utiliser GIT pour travailler avec mon binôme sur le projet. [sur 2 points]  
  **PREUVE :**  
  **Nous utilisons git pour mettre notre travail. Nous l’utilisons pour mettre le code en commun et pour avoir** **tous les deux le code pour pouvoir le** **travailler individuellement.**

******

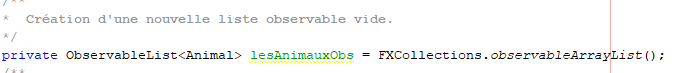
***=> 0/2***

* Je sais utiliser le type statique adéquat pour mes attributs ou variables. [sur 0.5 point]  
  **PREUVE :**  
  **Nous avons utilisé le type static car c’est un attribut qui est commun** **à tous les objets** **créés.**

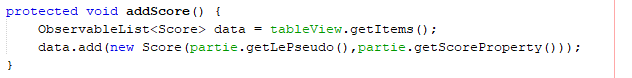
******

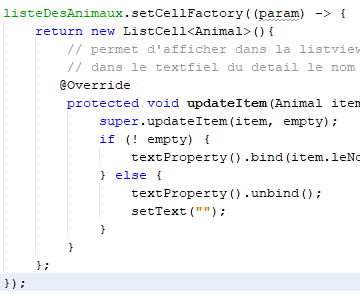
***=> 0/0.5***

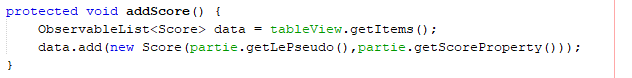
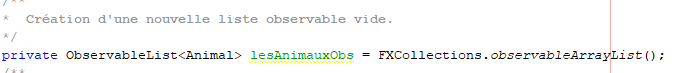
* Je sais utiliser les collections. [sur 1 point]  
  **PREUVE :**  
  **Dans notre master détail nous avons l’utilisation d’une collection d’animaux. En effet cela permet d’ajouter dans la liste** **tous les critères d’un animal. Nous utilisons une** **array List.**

 ***=> 0/1***

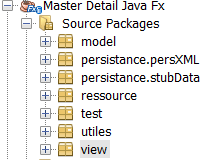
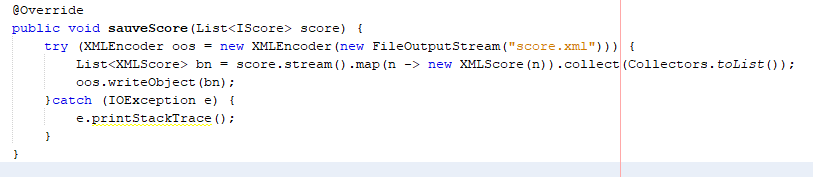
* Je sais utiliser les différents composants complexes (listes, combo…) que me propose JavaFX. [sur 1 point]  
  **PREUVE :**  
  **Nous utilisons une liste pour ajouter le pseudo et le score dans le** **tablView. Nous utilisons une** **observable Liste.**

 ***=> 0/1***

* Je sais utiliser les lambda-expression. [sur 1 point]  
  **PREUVE :**  
  **Nous utilisons des lambda expression pour amélioré le code pour la gestion d’évènement. *=> 0/1***
* Je sais utiliser les listes observables de JavaFX. [sur 1 point]  
  **PREUVE :**  
  **Nous avons utilisé comme** **dis auparavant** **les listes observables. Dans le premier cas nous utilisons une liste** **observableArrayliste pour l’ajout d’un animal dans une liste. Nous utilisons aussi d’une observable liste pour ajouter le pseudo et le score dans une** **tableView.**

******

***=> 0/1***

* Je sais utiliser les packages à bon escient dans un projet. [sur 1 point]  
  **PREUVE :**  
  **Nous avons utilisé des packages selon leurs fonctionnalités. Nous avons le** **package-modèle : il permet de mettre nos modèles qui** **dépendent du** **contrôleur. Nous** **a le package persistance : il permet de mettre toutes les** **classes en rapport** **à la persistance** **c'est-à-dire le chargement et la sauvegarde du pseudo et du score du joueur. Nous avons le package** **ressource : il permet de mettre les images et les sons utilisés et** **les pages FXML et CSS. Nous avons le package** **test : il contient notre main de l’application. Nous avons le package** **utiles : il permet de mettre toute** **la classe utile** **c'est-à-dire celle qu’on utilise pour la création d’objet. Nous utilisons le package view : il permet de mettre** **tous nos contrôleurs de page. *=> 0/1***
* Je sais utiliser les streams sur les collections Java8. [sur 1 point]  
  **PREUVE :**

**Le stream nous permet de parcourir une liste.**

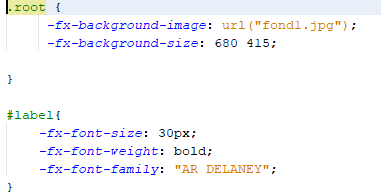
***=> 0/1***

* Je sais utiliser un convertisseur lors d’un bind entre deux propriétés JavaFX. [sur 1 point]  
  **PREUVE :**

**Pour afficher les points de vie dans le jeu il a fallu que nous passions d’un int en string pour pouvoir l’afficher sur un label. Mais en même temps nous puisons faire des calculs pour abaisser les points de vie lorsque nous perdons une vie dans le jeu.**

***=> 0/1***

* Je sais utiliser un fichier CSS pour styler mon application JavaFX. [sur 0.5 point]  
  **PREUVE :**  
  **Pour donner du style a nos fenêtre nous avons utilisé des fichier CSS.**

******

***=> 0/0.5***

* Je sais utiliser un formateur lors d’un bind entre deux propriétés JavaFX. [sur 1 point]  
  **PREUVE :**  
  ***Pas encore  
  => 0/1***