ISET SFAX AU 2017/2018 SEMESTRE I		
DEVOIR FINAL		
Classes : DSI2 RSI2 SEM2	*Matière : Prog. Objet*	*Nb pages : 4*
Enseignants : Halouani Naima, Megdich Imen, Moalla Hounaida, Souissi Hafedh		
Documents Non Autorisés	*Durée : 1 h 30 m*	***Barème : 5 + 15***

Questions de cours

1- Quelle est la différence entre une classe abstraite et une interface ?

2- Que permet l'opérateur **instanceof** ?

3- Que signifie le mot clef **static** associé à un attribut ?

   a. Que la valeur de cet attribut est constante
   b. Que cet attribut n'est visible que dans la classe où il est défini
   c. Que cet attribut sera toujours passé par valeur
   d. Que cet attribut a une référence unique pour toutes les instances de la classe

4- Au sujet du mot clef **abstract**, quelle est la ou les assertions fausses ?

   a. Une classe abstraite ne peut pas être instanciée
   b. Une méthode abstraite n'a pas d'implémentation
   c. Une classe abstraite n'a pas forcément de classe fille
   d. Une classe abstraite doit contenir au moins une méthode abstraite

5- Soit les classes suivantes :

```
public class A{
  public void m1(){
    System.out.println("Je suis la
                  méthode m1 de A") ;
  }
  public void m2(int i){
    System.out.println("Je suis la
                  méthode m2 de A") ;
  }
}
```

```
public class B extends A{
  public void m1(){
     System.out.println("Je suis la
                  méthode m1 de B") ;
  }
  public void m2(){
    System.out.println("Je suis la
                  méthode m2 de B") ;
  }
}
```

Examiner chacune des instructions suivantes et indiquer si elle génère une erreur de syntaxe à laquelle vous y proposez une correction et une trace d'exécution.

```
A a = new B();
B b = (B) a;
```

Instruction	Correction si erreur de Syntaxe	Trace d'exécution
`a.m1();`		
`a.m2();`		
`a.m2(5);`		
`b.m2(2);`		

Problème

Un **Store** (ou place de marché) est une vitrine électronique qui permet aux développeurs d'applications mobiles de publier leurs applications et qui permet aux utilisateurs de consulter, de rechercher et d'installer les applications mobiles. On désire écrire une application "**GestionStore**" qui permet de gérer les applications d'un store. "**GestionStore**" contient :

- "**Statistique**" : représente une interface qui contient trois méthodes.
- "**AvisUtilisateur**" : représente un avis d'un utilisateur, il peut être un **Commentaire** ou une **Evaluation**.
- "**Commentaire**" : représente un commentaire d'une application et il est défini par un texte.
- "**Evaluation**" : représente une évaluation d'une application et elle est définie par une note.
- "**AppMobile**" : représente une application mobile. Elle implémente l'interface **Statistique** et elle est définie par un nom, un thème (Sport, Jeu,…), un développeur, un système d'exploitation (Android, iOS,…), un prix et un nombre d'installation (nbIns). **AppMobile** possède aussi une constante MAX_AVIS (égale à 100), un nombre d'avis et un tableau qui contient les différents **AvisUtilisateur** sur cette application.
- "**Store**" : représente un store. Il implémente l'interface "Statistique" et est défini par un nom (Play Store, App Store …), une url (https://play.google.com/store, https://www.apple.com/fr/ios/app-store/, …), et un système d'exploitation (Android, iOS,…). **Store** possède aussi un **Vector** qui contient les **AppMobile**.
- "**TestStore**" : représente une classe de test des autres classes.

NB :

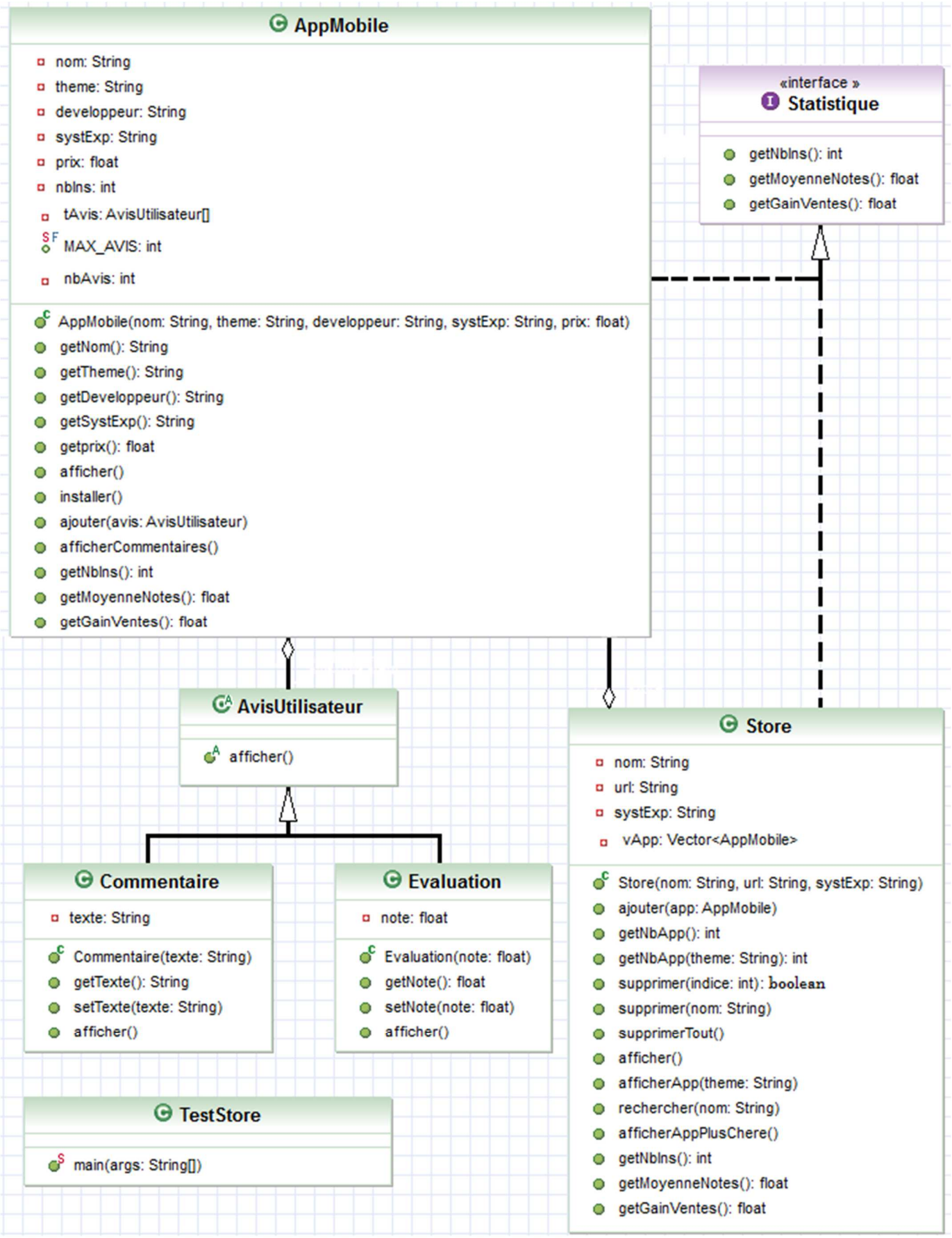
- Les getters et les setters des classes Evaluation, Commentaire et AppMobile sont supposés prédéfinis et peuvent être utilisés directement selon les besoins.
- Il est recommandé de respecter les noms de classes, méthodes et attributs fournis dans l'énoncé

La description de l'interface et des méthodes est données par le tableau suivant :

Classe ou interface	Méthodes	Descriptions
Statistique	`getNbIns()`	Méthodes de l'interface
	`getMoyenneNotes()`	
	`getGainVentes()`	
AvisUtilisateur	`afficher()`	Méthode abstraite
Commentaire	`Commentaire(...)`	Constructeur
	`afficher()`	Affiche le texte du commentaire
Evaluation	`Evaluation(...)`	Constructeur
	`afficher()`	Affiche la note de l'évaluation
AppMobile	`AppMobile(...)`	Constructeur
	`afficher()`	Affiche tous les attributs de la classe
	`installer()`	Affiche le message "Installation..." et incrémente le nombre d'installations
	`ajouter(AvisUtilisateur)`	Ajoute un avis utilisateur au tableau si c'est possible
	`afficherCommentaires()`	Si le tableau contient des commentaires, ils seront tous affichés, sinon la méthode affiche le message "Aucun commentaire."
	`getNbIns()`	Retourne le nombre d'installations de l'application
	`getMoyenneNotes()`	Si le tableau contient des évaluations, la méthode

		calcule et retourne la moyenne des notes des évaluations, sinon elle retourne -1
	`getGainVentes()`	Retourne le gain des ventes par la formule : « nombre d'installations * Prix »
Store	`Store(...)`	Constructeur
	`ajouter(AppMobile)`	Ajoute une application mobile au Vector si le store et l'application possèdent le même système d'exploitation
	`getNbApp()`	Retourne le nombre d'applications mobiles du store
	`getNbApp(String)`	Retourne le nombre d'applications mobiles du thème passé en paramètre
	`supprimer(int)`	Si l'indice passé en paramètre est valide, alors elle supprime l'application mobile correspondante. La méthode retourne un boolean.
	`supprimer(String)`	Supprime toutes les applications qui ont le nom passé en paramètre
	`supprimerTout()`	Supprime toutes les applications
	`afficher()`	Affiche tous les attributs du store
	`afficherApp(String)`	Recherche et affiche : - toutes les applications ayant comme thème le paramètre si elles existent, - le message "Aucune application pour ce thème." si aucune application ne correspond au thème spécifié.
	`rechercher(String)`	Recherche et affiche toutes les applications qui ont le nom passé en paramètre en ignorant la casse
	`afficherAppPlusChere()`	Si le store contient des applications, la méthode cherche et affiche la première application la plus chère (son prix et le maximum des prix), sinon elle affiche "Le store ne contient aucune application."
	`getNbIns()`	Retourne la somme des nombres d'installation de toutes les applications
	`getMoyenneNotes()`	Si le store contient au moins une application notée (son getMoyenneNotes() ne retourne pas -1) alors elle retourne la moyenne des notes des applications notées, sinon elle retourne -1
	`getGainVentes()`	Retourne la somme des gains des ventes des applications
TestStore	`main(String[])`	Permet de : - Déclarer et d'instancier un Store, - Afficher le Store - Déclarer et instancier deux applications - Ajouter les applications au Store - Afficher le nombre d'applications du store - Installer la première application trente fois - Installer la deuxième application dix fois - Afficher le gain des ventes de l'application - Afficher les gains des ventes du store

Le diagramme de classe de "**GestionStore**" est le suivant :



Travail demandé :

Implémenter l'**interface** et **toutes** les **classes** de ce diagramme.