Ce TD a pour but d'écrire le code permettant de lire des 'options' depuis un fichier texte comme celui ci-dessous.

```
FlagOption#mode de debogage=false
IntOption#profondeur de recherche=3
PathOption#chemin vers imgs=/usr/share/monpgm/rsr:rsr:../rsr
DirectionOption#boussole=Nord
```

config.ini

On supposera toujours que les lignes sont de la forme

`<type de l'option>#<nom de l'option>=<valeur de l'option>`

où le type ne contient pas de '#' ni de '=', où le nom ne contient pas de '=' et où la valeur est une chaîne a priori quelconque.

1 Manipulation de String

% s	attend `String`	%d	attend `int` (décimal)	%f	attend `double`
% c	attend `char`	%b	attend `bool`		
%n	affiche une fin de ligne	%%	affiche %		

TABLE 1 – Antisèche pour `format`

On considère la classe abstraite suivante.

```
public abstract class Option {
  public final String nom;
  public Option(String nom) { this.nom=nom; }
  abstract public String toConfigurationLine();
}
```

Exercice 1. Écrire une classe `FlagOption` qui étend `Option`. C'est une option à deux possibilités, elle est soit activée soit ne l'est pas. Lui ajouter un constructeur qui prend une `String` à couper elle-même. Par exemple, en utilisant le fichier `config.ini` en début du TD, elle recevrait `"mode␣de␣debogage=false"`
La méthode `toConfigurationLine()` renvoie la chaîne qui représente `this`, comme elle serait écrite dans `config.ini`.

Exercice 2. Écrire similairement la classe `IntOption` (il faut utiliser `Integer.parseInt(String)`).

Exercice 3. Écrire la classe `PathOption` qui admet comme attribut une liste de `File`. Dans l'exemple donné ci-dessus, la liste aurait la valeur `[usr/share/monpgm/rsr, rsr, ../rsr]`.

2 Reflexivité

Exercice 4. Écrire similairement la classe `DirectionOption`. Comment améliorer l'implémentation de `toConfigurationLine()` en utilisant la réflexivité ?

Exercice 5. Ajouter dans `Option` une méthode statique `depuisUneLine` qui prend une `String` correspondant à une ligne du fichier de configuration puis crée et renvoie l'`Option` appropriée. Il faut utiliser la réflexivité.

Exercice 6. Écrire une classe `GestionnaireDOptions` contenant une méthode `void treatFile(file f)` qui lit le fichier donné en argument puis, pour chaque ligne, crée l'objet du bon type et le stocke dans une liste, un attribut de `GestionnaireDOptions`.

3 Exceptions

Exercice 7. Créer trois exceptions (ne pas changer les autres classes pour l'instant) :

- `NePeutPasCreerOption` qui sera levée quand la ligne du fichier de configuration ne permet pas d'instancier correctement une classe (dans l'exercice 6) ;
- `OptionInexistante` qui sera levée quand on demande au Gestionnaire une option qui n'existe pas (exercice 8) ;
- `OptionMauvaisType` qui sera levée quand une option est demandée existe mais le type de retour ne correspond pas (exercice 8).

Lesquels devrait être étendre `RuntimeException` plutôt qu'`Exception` ?

Exercice 8. Ajouter dans `GestionnaireDOptions` trois getters qui lance l'exception `OptionInexistante` si l'option demandée n'existe pas et l'exception `OptionMauvaisType` si l'option n'est pas du bon type.

- `boolean getOptionFlag(String str)` ; marche uniquement pour `"mode␣de␣debogage"`.
- `int getOptionInt(String str)` ; marche uniquement pour `"profondeur␣de␣recherche"`.
- `List<File> getOptionList(String str)` ; marche uniquement pour `"chemin␣vers␣imgs"`.

Exercice 9. Traiter raisonnablement les exceptions que peuvent envoyer les différents appels réflexifs et la lecture de fichier (voir ci-dessous), notamment en utilisant `NePeutPasCreerOption`.

```
BufferedReader.readLine() throws IOException;

FileReader.FileReader(File f) throws FileNotFoundException;

public Constructor<T> Class.getConstructor(Class<?>... parameterTypes)
    throws
        NoSuchMethodException, //le constructeur n'existe pas
        SecurityException; //RuntimeException, probleme d'acces (ou autre)

Constructor.newInstance(Object.. args) throws
        IllegalAccessException,  //Probleme d'acces (public, etc)
        IllegalArgumentException, //RuntimeException, probleme d'arguments
        InstantiationException //Constructeur de classe abstraite
        InvocationTargetException, //Exception levee dans le constructeur
        ExceptionInInitializerError; //Error
```