BPO
TP 11 - Introspection & JUnit

Première partie : Introspection

On dispose d'une panoplie de classes permettant de jouer à divers jeux simples à deux joueurs. Par exemple, la classe **hasard.Hasard** permet de jouer au jeu du nombre au hasard (l'ordinateur choisit un nombre et l'utilisateur doit le trouver) ; la classe **mastermind.Mastermind** permet de jouer au jeu du Mastermind (l'ordinateur choisit une configuration de pions de couleur et l'utilisateur doit la trouver), *etc.*

Toutes ces classes de jeu sont quelconques, mais disposent toutes de un ou plusieurs constructeurs admettant uniquement des paramètres de type **int**, ainsi qu'une méthode **public void unePartie()** qui permet de dérouler une partie.

Pour pouvoir jouer indifféremment à un des jeux disponibles, on construit une nouvelle application **Lanceur** qui permet d'entamer une partie d'un jeu, sous réserve d'indiquer la classe de jeu et les paramètres du constructeur du jeu.

```
package lanceur ;
import java.lang.reflect.* ;
/**
 * La classe Lanceur permet de démarrer un jeu à deux joueurs, défini par une classe possédant
 *        - un constructeur avec 0, .. k entiers en paramètres
 *        - une méthode void unePartie()
 */
public class Lanceur{
   public Lanceur(String[] args) throws ExceptionJeu  { … }
   public void jouer() throws ExceptionJeu { … }
   public static void main(String[] args ) {
              ....
               LanceurJeu j = new LanceurJeu(args) ;
               j.jouer() ;
              ...
  }
}
```

Voici quelques exemples d'exécution de cette classe ; dans les premiers exemples, il est impossible de lancer le jeu, pour diverses raisons clairement identifiées dans le message d'erreur affiché.

```
% java LanceurJeu Hasard
Classe de jeu non trouvée
Jeu terminé
```

```
% java LanceurJeu hasard.Hasard 123 556 43
Instanciation impossible
Jeu terminé
```

```
% java LanceurJeu jeu.Quiz 1
Méthode de jeu unePartie() inexistante
Jeu terminé
```

```
% java LanceurJeu hasard.Hasard 200
A vous : 166
C'est trop
A vous : 12
C'est trop peu
        ...
Jeu terminé
```

```
% java LanceurJeu mastermind.Mastermind 5 7
Case 0 : 1
Case 1 : 2
Case 2 : 1
Case 3 : 2
Case 4 : 3
   > Nombre de pions de la bonne couleur bien placés 0
   > Nombre de pions de la bonne couleur mal placés 2
        ....
Jeu terminé
```

Mise en place de l'environnement de travail

- Téléchargez l'archive **jeux.jar** fourni sur Arche
- Créez un nouveau projet Java dans IntelliJ.
- Dans **src**, créez un package **lanceur**.
- Dans le menu File/ProjectStructure, sélectionnez Librairies. Ajoutez l'archive que vous venez de télécharger. Les classes **hasard.Hasard** et **mastermind.Mastermind** sont dorénavant intégrées à votre projet.

Écrire les classes **lanceur.LanceurJeu** et **lanceur.ExceptionJeu**.

IMPORTANT : le texte de la classe **LanceurJeu** doit être indépendant des classes de jeu disponibles, c'est-à-dire qu'on ne doit pas y retrouver les noms des classes **Hasard** et **Mastermind**.

Les tests de nouveaux jeux doivent pouvoir se faire, sans recompiler la classe **LanceurJeu**.

Pour cela, il faut instancier le jeu en utilisant la méthode **newInstance()** de la classe **Class** et appeler la méthode **unePartie()** en utilisant la méthode invoke de la classe **Method**.

Seconde partie : JUnit

On développe une application destinée à des enfants qui apprennent à faire des additions. Dans cette application, on utilise la classe **arithmetique.Addition** qui mémorise une addition et son résultat. Le squelette de cette classe est fourni sur Arche.

Construire un nouveau projet dans **IntelliJ** pour y placer cette classe. Ajouter un package de tests.

Écrire cette classe fonction après fonction. Dès que l'une des fonctions est écrite, compléter une classe de test **JUnit** avec les tests adéquats.

Le test de cette classe vous pose-t-il un problème ? Lequel ? Avez-vous une solution à ce problème ?