Travail demandé

1. **Partie1** :
   Pour la première partie, vous devez imprimer (en recto verso) et compléter le questionnaire qui se trouve sur la page du cours et le remettre au département informatique (la procédure est sur la page du cours).
   Les questions doivent vous orienter et vous guider dans la conception de vos algorithmes.

2. **Partie2** :
   Pour la deuxième partie, vous devez écrire votre programme dans un fichier Java (voir les contraintes que vous devez respecter) et le remettre sur Oto (la procédure pour remettre un travail sur Oto est disponible sur la page du cours).
   La boîte Oto sera disponible uniquement trois jours avant la date limite.

Remarques importantes

— Vous devez consulter la page du cours pour avoir les dates limites des remises.
— Si durant votre travail vous avez une question :
   1. Consultez la page contenant les réponses aux questions des étudiants pour vérifier si une réponse a déjà été donnée à votre question.
   2. Si vous ne trouvez pas de réponse à votre question, contactez votre enseignant (évitez de poser vos questions la veille de la date limite car vous risquez de ne pas avoir de réponse à temps).
   3. L'enseignant ne répond à aucune question se référant à vos réponses avant la fin de l'épreuve. Autrement dit, vous ne devez à aucun moment de l'épreuve poser une question relative à votre code ou à vos réponses au questionnaire.

Boîte Oto	**Enseignant**
Tp1	ben_ro

Il s'agit de mettre en oeuvre un jeu de stratégie. Le jeu se joue à deux (à tour de rôle). Pour jouer, nous avons besoin de trois piles de cartes (chaque pile doit avoir au minimum deux cartes au début du jeu).
À tour de rôle, chaque joueur doit obligatoirement enlever au moins une carte d'une pile de son choix (et au plus trois cartes). Le joueur qui retire les dernières cartes perd la partie (même s'il reste une seule carte, le dernier joueur est obligé de la prendre).
Voici un exemple qui illustre le déroulement d'une partie du jeu :

Pile1	Pile2	Pile3	Coups joués
5	3	7	État initial des piles
3	3	7	État des piles après le coup suivant : Joueur1 prend 2 cartes de la Pile1
3	3	4	État des piles après le coup suivant : Joueur2 prend 3 cartes de la Pile3
3	2	4	État des piles après le coup suivant : Joueur1 prend 1 carte de la Pile2
0	2	4	État des piles après le coup suivant : Joueur2 prend 3 cartes de la Pile1
0	0	4	État des piles après le coup suivant : Joueur1 prend 3 cartes de la Pile2
0	0	1	État des piles après le coup suivant : Joueur2 prend 3 cartes de la Pile3
0	0	0	État des piles après le coup suivant : Joueur1 prend 1 carte de la Pile3
-	-	-	Le Joueur1 a perdu car c'est lui qui a pris la dernière carte

Votre programme doit commencer par demander le nombre initial de cartes pour chaque pile. Ensuite, il invite chaque joueur (à tour de rôle) à effectuer un coup en leur demandant de saisir le numéro de la pile cible ainsi que le nombre de cartes à retirer de la pile choisie.
Comme vous pouvez le remarquer dans l'exemple donné plus haut, au troisième coup du Joueur1 *(Joueur1 prend 3 cartes de la Pile2)*, si un joueur tente de retirer plus de cartes que ce qu'il existe dans une pile, il faut vider la pile. Par contre, si un joueur tente de retirer des cartes d'une pile vide, votre programme doit afficher un message d'erreur et l'inviter à rejouer son coup.
À la fin d'une partie, votre programme doit afficher le vainqueur (Joueur1 ou Joueur2) avant de terminer son exécution.

Remarques concernant les validations : Seules les validations « logiques » sont requises ici, c'est-à-dire que vous n'avez qu'à vérifier si les nombres entrés sont bien dans les intervalles permis. Vous supposerez qu'un nombre entier sera entré lorsque requis (donc pas de validation lexicale ou syntaxique des données).

Messages à afficher par votre programme

Vous devez absolument utiliser la classe `MessagesTp1` disponible sur la page du cours pour les instructions d'affichage. Chaque instruction correspond à un message spécifique. Vous devez choisir l'instruction qui correspond au message désiré.
Cela dit, en ce qui concerne l'instruction qui sert à afficher l'état des piles, c'est à vous de la produire. Elle doit produire un affichage qui correspond à la forme suivante (chaque caractère est important) :

```
11:5:2
```

Voici l'instruction à utiliser pour cet affichage (vous pouvez utiliser des noms de variables différents de ceux utilisés ici) :

```
System.out.println( nbCartesP1 +":"+ nbCartesP2 +":"+ nbCartesP3 );
```

Notez aussi que l'affichage de l'état des piles doit être effectué uniquement avant chaque coup.

Un caractère en plus ou un caractère en moins peut affecter les tests.
Veillez à respecter les spécifications pour l'affichage et assurez-vous que pendant la remise de votre programme sur Oto aucun message d'erreur n'est affiché.

1 Tests de base

Votre programme doit réussir les tests de base suivants. Un test est réussi uniquement si votre programme réussi à le reproduire de façon identique (en respectant les minuscules et les majuscules et en respectant l'ordre des lignes affichées).

En plus des tests de base, votre programme doit répondre à toutes les spécifications présentées dans l'énoncé. Vous devez donc tester votre programme avec vos propres tests.

Votre fichier sera testé avec différents tests. Un fichier qui ne passe aucun test aura la note zéro.

Seule une partie de ces tests sera vérifiée au moment de la remise de votre travail. Si Oto n'affiche aucun message d'erreur, cela ne signifie pas que votre programme passe tous les tests.

— Test 01

```
Nombre de cartes pour la pile 1:
2
Nombre de cartes pour la pile 2:
2
Nombre de cartes pour la pile 3:
2
** Etat des piles:
2:2:2
=> Coup du joueur 1
Pile cible:
1
Nombre de cartes:
2
** Etat des piles:
0:2:2
=> Coup du joueur 2
Pile cible:
2
Nombre de cartes:
2
** Etat des piles:
0:0:2
=> Coup du joueur 1
Pile cible:
3
Nombre de cartes:
2
Gagnant: joueur 2
```

— Test 02

```
Nombre de cartes pour la pile 1:
1
Une pile doit avoir au moins 2 cartes.
Nombre de cartes pour la pile 1:
2
Nombre de cartes pour la pile 2:
2
Nombre de cartes pour la pile 3:
2
** Etat des piles:
2:2:2
=> Coup du joueur 1
Pile cible:
1
Nombre de cartes:
2
** Etat des piles:
0:2:2
=> Coup du joueur 2
Pile cible:
2
Nombre de cartes:
2
** Etat des piles:
0:0:2
=> Coup du joueur 1
Pile cible:
3
Nombre de cartes:
2
Gagnant: joueur 2
```

— Test 03

```
Nombre de cartes pour la pile 1:
2
Nombre de cartes pour la pile 2:
0
Une pile doit avoir au moins 2 cartes.
Nombre de cartes pour la pile 2:
2
Nombre de cartes pour la pile 3:
2
** Etat des piles:
2:2:2
=> Coup du joueur 1
Pile cible:
1
Nombre de cartes:
2
** Etat des piles:
0:2:2
=> Coup du joueur 2
Pile cible:
2
Nombre de cartes:
2
** Etat des piles:
0:0:2
=> Coup du joueur 1
Pile cible:
3
Nombre de cartes:
2
Gagnant: joueur 2
```

— Test 04

```
Nombre de cartes pour la pile 1:
2
Nombre de cartes pour la pile 2:
2
Nombre de cartes pour la pile 3:
-1
Une pile doit avoir au moins 2 cartes.
Nombre de cartes pour la pile 3:
2
** Etat des piles:
2:2:2
=> Coup du joueur 1
Pile cible:
3
Nombre de cartes:
2
** Etat des piles:
2:2:0
=> Coup du joueur 2
Pile cible:
2
Nombre de cartes:
2
** Etat des piles:
2:0:0
=> Coup du joueur 1
Pile cible:
1
Nombre de cartes:
2
Gagnant: joueur 2
```

— Test 05

```
Nombre de cartes pour la pile 1:
-1
Une pile doit avoir au moins 2 cartes.
Nombre de cartes pour la pile 1:
3
Nombre de cartes pour la pile 2:
-1
Une pile doit avoir au moins 2 cartes.
Nombre de cartes pour la pile 2:
3
Nombre de cartes pour la pile 3:
-1
Une pile doit avoir au moins 2 cartes.
Nombre de cartes pour la pile 3:
3
** Etat des piles:
3:3:3
=> Coup du joueur 1
Pile cible:
2
Nombre de cartes:
3
** Etat des piles:
3:0:3
=> Coup du joueur 2
Pile cible:
1
Nombre de cartes:
3
** Etat des piles:
0:0:3
=> Coup du joueur 1
Pile cible:
3
Nombre de cartes:
3
Gagnant: joueur 2
```

— Test 06

```
Nombre de cartes pour la pile 1:
2
Nombre de cartes pour la pile 2:
2
Nombre de cartes pour la pile 3:
2
** Etat des piles:
2:2:2
=> Coup du joueur 1
Pile cible:
1
Nombre de cartes:
3
** Etat des piles:
0:2:2
=> Coup du joueur 2
Pile cible:
2
Nombre de cartes:
3
** Etat des piles:
0:0:2
=> Coup du joueur 1
Pile cible:
3
Nombre de cartes:
3
Gagnant: joueur 2
```

— Test 07

```
Nombre de cartes pour la pile 1:
4
Nombre de cartes pour la pile 2:
4
Nombre de cartes pour la pile 3:
4
** Etat des piles:
4:4:4
=> Coup du joueur 1
Pile cible:
1
Nombre de cartes:
4
Vous devez retirer au moins une carte et au plus trois
Nombre de cartes:
3
** Etat des piles:
1:4:4
=> Coup du joueur 2
Pile cible:
1
Nombre de cartes:
3
** Etat des piles:
0:4:4
=> Coup du joueur 1
Pile cible:
2
Nombre de cartes:
4
Vous devez retirer au moins une carte et au plus trois
Nombre de cartes:
3
** Etat des piles:
0:1:4
=> Coup du joueur 2
Pile cible:
2
Nombre de cartes:
3
** Etat des piles:
0:0:4
=> Coup du joueur 1
Pile cible:
3
Nombre de cartes:
3
** Etat des piles:
0:0:1
=> Coup du joueur 2
Pile cible:
3
Nombre de cartes:
3
Gagnant: joueur 1
```

— Test 08

```
Nombre de cartes pour la pile 1:
2
Nombre de cartes pour la pile 2:
2
Nombre de cartes pour la pile 3:
2
** Etat des piles:
2:2:2
=> Coup du joueur 1
Pile cible:
1
Nombre de cartes:
0
Vous devez retirer au moins une carte et au plus trois
Nombre de cartes:
2
** Etat des piles:
0:2:2
=> Coup du joueur 2
Pile cible:
2
Nombre de cartes:
-1
Vous devez retirer au moins une carte et au plus trois
Nombre de cartes:
2
** Etat des piles:
0:0:2
=> Coup du joueur 1
Pile cible:
3
Nombre de cartes:
4
Vous devez retirer au moins une carte et au plus trois
Nombre de cartes:
2
Gagnant: joueur 2
```

— Test 09

```
Nombre de cartes pour la pile 1:
2
Nombre de cartes pour la pile 2:
2
Nombre de cartes pour la pile 3:
2
** Etat des piles:
2:2:2
=> Coup du joueur 1
Pile cible:
0
Numero de pile inexistant: choisir 1, 2 ou 3
Pile cible:
4
Numero de pile inexistant: choisir 1, 2 ou 3
Pile cible:
-1
Numero de pile inexistant: choisir 1, 2 ou 3
Pile cible:
1
Nombre de cartes:
1
** Etat des piles:
1:2:2
=> Coup du joueur 2
Pile cible:
1
Nombre de cartes:
1
** Etat des piles:
0:2:2
=> Coup du joueur 1
Pile cible:
2
Nombre de cartes:
1
** Etat des piles:
0:1:2
=> Coup du joueur 2
Pile cible:
2
Nombre de cartes:
1
** Etat des piles:
0:0:2
=> Coup du joueur 1
Pile cible:
3
Nombre de cartes:
1
** Etat des piles:
0:0:1
=> Coup du joueur 2
Pile cible:
3
Nombre de cartes:
1
Gagnant: joueur 1
```

— Test 10

```
Nombre de cartes pour la pile 1:
2
Nombre de cartes pour la pile 2:
2
Nombre de cartes pour la pile 3:
2
** Etat des piles:
2:2:2
=> Coup du joueur 1
Pile cible:
1
Nombre de cartes:
2
** Etat des piles:
0:2:2
=> Coup du joueur 2
Pile cible:
1
La pile choisie est vide, veuillez choisir une autre pile
Pile cible:
-1
Numero de pile inexistant: choisir 1, 2 ou 3
Pile cible:
1
La pile choisie est vide, veuillez choisir une autre pile
Pile cible:
2
Nombre de cartes:
2
** Etat des piles:
0:0:2
=> Coup du joueur 1
Pile cible:
2
La pile choisie est vide, veuillez choisir une autre pile
Pile cible:
4
Numero de pile inexistant: choisir 1, 2 ou 3
Pile cible:
3
Nombre de cartes:
2
Gagnant: joueur 2
```

— Test 11

```
Nombre de cartes pour la pile 1:
1
Une pile doit avoir au moins 2 cartes.
Nombre de cartes pour la pile 1:
2
Nombre de cartes pour la pile 2:
2
Nombre de cartes pour la pile 3:
2
** Etat des piles:
2:2:2
=> Coup du joueur 1
Pile cible:
0
Numero de pile inexistant: choisir 1, 2 ou 3
Pile cible:
1
Nombre de cartes:
4
Vous devez retirer au moins une carte et au plus trois
Nombre de cartes:
3
** Etat des piles:
0:2:2
=> Coup du joueur 2
Pile cible:
2
Nombre de cartes:
2
** Etat des piles:
0:0:2
=> Coup du joueur 1
Pile cible:
3
Nombre de cartes:
2
Gagnant: joueur 2
```

— Test 12

```
Nombre de cartes pour la pile 1:
5
Nombre de cartes pour la pile 2:
5
Nombre de cartes pour la pile 3:
5
** Etat des piles:
5:5:5
=> Coup du joueur 1
Pile cible:
1
Nombre de cartes:
3
** Etat des piles:
2:5:5
=> Coup du joueur 2
Pile cible:
2
Nombre de cartes:
2
** Etat des piles:
2:3:5
=> Coup du joueur 1
Pile cible:
3
Nombre de cartes:
3
** Etat des piles:
2:3:2
=> Coup du joueur 2
Pile cible:
1
Nombre de cartes:
2
** Etat des piles:
0:3:2
=> Coup du joueur 1
Pile cible:
2
Nombre de cartes:
3
** Etat des piles:
0:0:2
=> Coup du joueur 2
Pile cible:
3
Nombre de cartes:
2
Gagnant: joueur 1
```

— Test 13

```
Nombre de cartes pour la pile 1:
1
Une pile doit avoir au moins 2 cartes.
Nombre de cartes pour la pile 1:
2
Nombre de cartes pour la pile 2:
2
Nombre de cartes pour la pile 3:
2
** Etat des piles:
2:2:2
=> Coup du joueur 1
Pile cible:
0
Numero de pile inexistant: choisir 1, 2 ou 3
Pile cible:
1
Nombre de cartes:
4
Vous devez retirer au moins une carte et au plus trois
Nombre de cartes:
3
** Etat des piles:
0:2:2
=> Coup du joueur 2
Pile cible:
2
Nombre de cartes:
2
** Etat des piles:
0:0:2
=> Coup du joueur 1
Pile cible:
3
Nombre de cartes:
2
Gagnant: joueur 2
```

2 Contraintes à respecter pour le fichier JAVA à remettre

Nom du fichier à remettre (respectez les minuscules et les majuscules)	Tp1.java
Nom de la classe à définir (respectez les minuscules et les majuscules)	Tp1
Nombre maximal de lignes dans votre fichier JAVA (incluant les lignes vides et les commentaires)	117
Nombre maximal de `return`	0
Nombre maximal de `if` (les `else if` sont aussi comptabilisés)	10
Nombre maximal de `switch`	2
Nombre maximal de boucles `for`	0
Nombre maximal cumulé de boucles `while` et `do-while`	6
Nombre maximal d'instructions d'affichage `System.out.println` (vous devez utiliser la classe `MessagesTp1` donnée pour effectuer l'affichage des messages texte)	17

Signatures des méthodes demandées pour ce travail (respectez les minuscules et les majuscules)

```
public static void main(String [] args)
```

1. Le fichier demandé doit contenir la définition d'une seule classe Java.
2. Votre fichier ne doit pas contenir des méthodes autres que celles demandées.
3. Vous devez avoir une seule instruction par ligne.
4. Chaque déclaration de variable doit faire l'objet d'une instruction à part.
5. Vos lignes de code ne doivent pas dépasser 80 caractères (incluant les espaces, les tabulations et les commentaires).
6. Un niveau d'indentation doit correspondre à 4 espaces.
7. Vous devez respecter les conventions de nommage des variables et des constantes.
8. Au besoin, vous devez utiliser la classe `Clavier` pour saisir des données. Vos programmes seront testés avec cette classe et dans BlueJ.
9. Vous devez toujours utiliser les classes `DecimalFormat` et `DecimalFormatSymbols` pour l'affichage des réels comme dans l'exemple plus bas.
10. Votre fichier ne doit contenir aucune occurrence des termes interdits suivants, sinon la note zéro sera appliquée : `LineNumberReader`, `ArrayList`, `System.arraycopy`, `Arrays`, `System.exit`, `Scanner`, `StringBuilder` et `extends`.
11. La méthode `Clavier.lireChar()` est interdite, il faut utiliser `Clavier.lireCharLn()`.
12. La méthode `System.out.print()` est interdite, il faut utiliser `System.out.println()`.
13. Toute constante réelle utilisée dans votre code doit être associée à une constante symbolique.
14. L'opérateur ternaire de la forme suivante est interdit `(condition)? valeur_si_vrai : valeur_si_faux;`
15. L'utilisation du caractère de tabulation est interdite.
16. Pour l'affichage, vous n'avez pas le droit d'utiliser "\n" pour échapper à la limitation du nombre de `System.out.println`.
17. Un fichier qui ne compile pas aura la note zéro.

Exemple d'utilisation de la classe `DecimalFormat`

```
import java.text.*;

public class TestDecimalFormat {
    public static void main( String[] args ) {
        DecimalFormatSymbols dfs = new DecimalFormatSymbols();
        dfs.setDecimalSeparator('.');
        DecimalFormat df = new DecimalFormat( "0.00", dfs );

        double var = 1.2300000000001;
        System.out.println( df.format( var ) );
    }
}
```