Travail pratique 1: Jeu de roche-papier-ciseau

<u>En équipe de 2 ou 3 (les noms des membres doivent apparaître dans l'entête du code!!!)</u>

1. Spécifications

Vous devez réaliser un programme qui représente un jeu de roche-papier-ciseau entre deux joueurs humains. Si vous ne connaissez pas le jeu, chaque joueur choisit un des trois objets, et on détermine qui a gagné la manche selon les simples règles suivantes:

- roche gagne contre ciseau
- ciseau gagne contre papier
- papier gagne contre roche

Au lancement du programme, un message de bienvenue s'affiche, et on demande le nombre de manches qu'on souhaite jouer dans la partie, comme suit:

```
-----------------------------------------------
--- Bienvenue au jeu de roche-papier-ciseau ---
-----------------------------------------------

Combien de manches voulez-vous jouer?
```

Le programme lit le nombre entier décimal entré dans le terminal.
<u>Important</u>*: si le nombre entré est pair, il faut augmenter sa valeur de 1 puisqu'il serait sinon impossible de déterminer un gagnant à la fin du match s'il y avait égalité.*

Ensuite, pour chaque manche du match, on affiche le nombre de manches restantes et on demande au joueur 1 son choix:

```
Il reste 3 manche(s) à jouer.
JOUEUR 1, quel est votre choix? [r/p/c]
```

La seule entrée valide est l'un des 3 caractères **r**, **p**, ou **c**. Si l'utilisateur entre n'importe quel autre caractère, ou bien une chaîne de caractères, cela cause une erreur et la fin du programme avec le message suivant:

```
Erreur d'entrée! Programme terminé.
```

Si l'entrée était valide, on demande ensuite au joueur 2 son choix (et vérifie sa validité):

```
JOUEUR 2, quel est votre choix? [r/p/c]
```

Le programme compare ensuite le choix des deux joueurs et annonce le gagnant de la manche, puis affiche le score actuel sous la forme "Score: *Score1-Score2*":

```
JOUEUR 1 a gagné cette manche! Score: 1-0
```

Si le choix des deux joueurs est identique, on indique plutôt qu'il s'agît d'une manche nulle:

```
Manche nulle...
```

Le programme répète cette procédure pour chaque manche, jusqu'à ce que <u>le nombre choisi de manches ait été joué</u> OU <u>qu'un des deux joueurs ait déjà gagné</u>.

Note: il est possible qu'un des joueurs gagne avant que toutes les manches soient jouées

Quand le match est terminé, on affiche le gagnant du match et le score:

```
JOUEUR 2 A GAGNÉ LE MATCH! FÉLICITATIONS!
SCORE FINAL: 1-3
```

Puis le programme termine.

2. Réalisation

La réalisation de ce programme se fera par groupes de deux ou trois membres. Chaque membre de l'équipe devra maîtriser tous les aspects du programme.
Le travail pratique comporte deux parties :

1. Le programme en Java traduisant l'algorithme. (**remise: 12 févier 2021, 23h59**)
   À déposer sur Moodle: *rpc.java*
2. Le programme en assembleur Pep/8. (**remise: 26 février 2021, 23h59**)
   À déposer sur Moodle: *rpc.pep*

Les fichiers doivent être déposés sur **Moodle**. Le travail peut être remis jusqu'à 3 jours en retard à raison d'une pénalité de 10 points par jour. Après ces 3 jours, les travaux ne seront plus acceptés et la note de 0 sera automatique pour cette portion du travail.

Attention: L'objectif de la partie Java est de vous aider à concevoir un algorithme qui fonctionne mais qui soit transposable en Pep/8. C'est pourquoi les contraintes particulières suivantes doivent êtres respectés :

- N'utilisez pas de fonctionnalités avancées de Java (fonctions de bibliothèques, programmation par objets, etc).
- N'utilisez pas de multiplication, de division ou toute opération arithmétique inconnue de Pep/8.
- N'utilisez pas de chaînes de caractères (String, StringBuffer, etc.) pour autre chose que de l'affichage.
- N'utilisez pas l'opérateur *new* d'instanciation.
- N'utilisez pas de tableaux, genre char[] ou int[].
- Pour les entrées-sorties, **vous devez utiliser la classe fournie, Pep8.java** qui simule certaines instructions du processeur Pep/8 (deci, deco, chari, charo, stro, stop), pas la classe *Scanner* (ou autre) de Java
- Votre programme doit tenir dans une seule classe, et ne pas faire appel à des sous-programmes (méthodes, fonctions, etc).
- Ne créez pas de structure de packages.

---

2. Évaluation

Élément	Java	Pep/8
Tests Le programme sera testé avec les entrées du test public donné en partie 3, ainsi qu'avec des entrées privées. Le programme doit se comporter de la façon spécifiée, pour toutes les entrées possibles.	/10	/30
Documentation Le programme doit être bien lisible, avec entête de programme, et les commentaires doivent être clairs, précis, utiles et doivent aider à comprendre le fonctionnement du programme	/10	/20
Qualité logicielle et de programmation Le code doit être bien structuré et le plus simple possible, en évitant les redondances. Les erreurs couvrent tous les cas limites et les messages d'erreurs sont clairs.	/20	/10
Total	**40%**	**60%**

3. Test public

```
-----------------------------------------------
--- Bienvenue au jeu de roche-papier-ciseau ---
-----------------------------------------------

Combien de manches voulez-vous jouer?
4


Il reste 5 manche(s) à jouer.
JOUEUR 1, quel est votre choix? [r/p/c]
r
JOUEUR 2, quel est votre choix? [r/p/c]
c
JOUEUR 1 a gagné cette manche! Score: 1-0

Il reste 4 manche(s) à jouer.
JOUEUR 1, quel est votre choix? [r/p/c]
r
JOUEUR 2, quel est votre choix? [r/p/c]
r
Manche nulle...

Il reste 4 manche(s) à jouer.
JOUEUR 1, quel est votre choix? [r/p/c]
p
JOUEUR 2, quel est votre choix? [r/p/c]
c
JOUEUR 2 a gagné cette manche! Score: 1-1

Il reste 3 manche(s) à jouer.
JOUEUR 1, quel est votre choix? [r/p/c]
r
JOUEUR 2, quel est votre choix? [r/p/c]
p
JOUEUR 2 a gagné cette manche! Score: 1-2

Il reste 2 manche(s) à jouer.
JOUEUR 1, quel est votre choix? [r/p/c]
c
JOUEUR 2, quel est votre choix? [r/p/c]
r
JOUEUR 2 a gagné cette manche! Score: 1-3

JOUEUR 2 A GAGNÉ LE MATCH! FÉLICITATIONS!
SCORE FINAL: 1-3
```