420-935-RO Concepts de la programmation orientée objet
TP1 – Casino

Sylvain Labranche — Collège de Rosemont

Automne 2020

1 Énoncé

Vous avez la brillante idée de vous lancer dans la gestion d'un casino en ligne illégal. Avant de mettre votre casino en fonction, vous devez écrire un programme Java qui gère des casinos et le tester. Votre apprentissage de la programmation orientée objet vous permet de développer un locgiciel en respectant les concepts la POO.

Minimalement, votre programme aura 4 classes: Joueur.java, Casino.java, Jeu.java et TestCasino.java. Ces classes se retrouvent respectivement dans les paquetages Joueurs, Casinos, Jeux et Main.

Vous devez faire **à la main** le diagramme UML de vos classes.

1. La classe Casino aura les attributs suivants:

   - nom : String
   - joueurs : Joueurs[], un tableau des joueurs présents dans le casino.
   - joueursPresents : int, le nombre de joueurs présents dans le casino.
   - jeu : Jeu, le seul Jeu disponible au casino.

   La classe Casino aura les méthodes suivantes:

   - ajouterJoueur(Joueur) : Le Joueur est ajouté au tableau des joueurs du Casino. Si le casino est plein, refuse le Joueur. Le nombre de joueurs présents est augmenté de 1. La méthode *ajouterJoueur* est appelée seulement par la méthode *joindreCasino* de Joueur.
   - enleverJoueur(Joueur) : Recherche dans le tableau de joueurs si le Joueur est présent. Si oui, l'enlève du tableau et décale tous les Joueurs suivants d'une case vers la gauche (pour qu'il n'y ait pas de trous dans le tableau). Le nombre de joueurs présents est diminué de 1. La méthode *enleverJoueur* est appelée seulement par la méthode *quitterCasiono* de Joueur.
   - jouer(int) ou jouer(Joueur, int) : Le Casino met à jour le capital du Joueur selon le résultat du jeu.
   - Vous devez fournir au moins deux constructeurs, le constructeur par copie, les accesseurs et les mutateurs que vous jugez pertinents, la méthode toString et la méthode equals.

     La méthode toString() affiche le nom du Casino, la liste des joueurs présents ainsi que les règles du Jeu.

     Un Jeu n'existera pas en dehors d'un Casino: Les constructeurs de Casino créent un nouveau Jeu avec *new*.

2. La classe Joueur aura les attributs suivants:

   - nom : String
   - capital : int
   - casino : Casino

   La classe Joueur aura les méthodes suivantes:

   - joindreCasino(Casino) : Le Joueur est ajouté à la liste des joueurs présents au Casino envoyé en argument. Le Casino refuse le joueur si celui-ci n'a pas d'argent. Attention de vérifier s'il reste de la place dans le Casino!
   - quitterCasino(Casino) : Le Joueur est enlevé de la liste des joueurs présents au Casino envoyé en argument. S'il n'est pas dans la liste, un message d'erreur est affiché.
   - jouer(int mise) : Le joueur doit être dans un casino pour jouer. Le montant est joué dans le Jeu du Casino et le capital du joueur est mis à jour selon ses gains.
   - Vous devez fournir au moins deux constructeurs, le constructeur par copie, les accesseurs et les mutateurs que vous jugez pertinents, la méthode toString et la méthode equals.

3. La classe Jeu aura les attributs suivants:

   - nom : String

   La classe Jeu aura les méthodes suivantes:

   - calculerGains(int mise): Ici, vous implémentez le jeu de hasard de votre choix, avec la table des gains que vous voulez. Vous devez utiliser au moins un nombre aléatoire.
     Vous pouvez reprendre un jeu de votre ancien Casino.
   - Vous devez fournir au moins un constructeur, le constructeur par copie, les accesseurs et les mutateurs que vous jugez pertinents, la méthode toString et la méthode equals.
   - La méthode toString() affiche les règles du jeu.

4. La classe TestCasino aura une méthode *main* qui agit comme programme de test. Vous êtes responsable d'y tester **chacune** des méthodes implémentées pour **chacune** des classes.

2 Méthodologie proposée

1. Commencez par le diagramme UML. Vous aurez une meilleure idée de l'organisation de vos classes.
2. Allez-y étape par étape et testez à chaque fois. Implémentez une méthode et testez-la immédiatement après. Mettez le code de tests des méthodes fonctionnelles en commentaire dans votre programme principal.
3. Écrivez les attributs des classes.
4. Commencez par les méthodes les plus faciles (constructeurs, toString, accesseurs, mutateurs, etc.) et allez-y une à la fois.
5. Une fois la base de vos trois classes fonctionnelle, implémentez les méthodes où les classes interagissent ensemble.

Pour obtenir un nombre aléatoire:

La méthode Math.Random() retourne un nombre aléatoire entre 0 et 1 (1 n'est pas inclus).

Pour obtenir un nombre aléatoire entre 1 et n inclusivement, on fait :

$(int)(n * Math.Random()) + 1;$

Pour obtenir un nombre aléatoire qui est soit 0, soit 1, on fait :

$(int)(2 * Math.random());$

3 Modalités

Ce travail est fait individuellement. Vous pouvez collaborer, mais vous **devez** écrire votre propre code. Un plagiat, même partiel, du code d'un autre étudiant entraînera la note 0 pour les deux étudiants.

4 Remise

La date limite pour la remise est prévue **le lundi 18 janvier 2021 à 23h59**. Vous aurez plusieurs heures en classe pour travailler sur votre TP.

Vous remettez le projet dans une archive .Zip dans la boîte prévue à cet effet sur Léa.
Tout retard entraînera une diminution de 10 % de la note maximale par jour de retard.

5 Évaluation

Ce travail compte pour 30 % de votre session.

Le diagramme UML compte pour 4 %, chacune des classes Joueur, Casino et Jeu pour 7 % et votre programme de test pour 5 % La qualité du code, l'implémentation correcte des méthodes et l'exhaustivité des tests seront évalués.