Collège LaSalle
Montréal

Évaluation Intra : 30 %

Identification du cours

Nom du programme – Code:	TECHNIQUES DE L'INFORMATIQUE – PROGRAMMATION EN JEUX VIDÉO – 420.BX
Titre du cours :	**CONCEPTS ET PROGRAMMATION OBJET II**
Numéro du cours :	420-JV7-AS
Groupe :	07243
Nom de l'enseignante :	Salima Hassaine
Durée de l'évaluation :	3 périodes (150 minutes)
Session :	Automne 2020

Identification de l'étudiant

Nom : _____ Numéro d'étudiant : _____

Date : _____ Résultat : _____

☐ Je déclare que ceci est un travail original et que j'ai mentionné toutes les sources du contenu dont je ne suis pas l'auteur (sources en ligne et imprimées, images, graphiques, films, etc.) dans le style de citation requis.

Consignes

- Les notes de cours ne sont pas permis.
- Aucune pause ne sera accordée pendant l'examen. Aucun étudiant ne peut quitter la salle d'examen avant que la moitié du temps alloué ne se soit écoulée. L'étudiant qui quitte la salle d'examen ne peut y revenir (PIEA – Article 5.12.4).
- L'enseignant(e) ne répondra pas aux questions.
- Les étudiants ne sont pas autorisés à communiquer entre eux.
- L'enseignant(e) a le devoir d'identifier les erreurs de langue dans les travaux des étudiants. Un pourcentage attribué à la langue va jusqu'à 20 % de la note (PIEA – Article 5.7).
- Le plagiat, la tentative de plagiat ou la coopération à un plagiat lors d'une épreuve sommative entraîne la note zéro (0). Dans le cas de récidive, dans le même cours ou dans un autre cours, l'étudiant se voit octroyer un « 0 » pour le cours concerné. (PIEA – Article 5.16).
- Veuillez écrire de façon lisible.

Répartition des points

- Question 1 : Interface IMonster — Pour un total de 05 points
- Question 2 : abstract class Monster — Pour un total de 22 points
- Question 3 : child class Pokemon — Pour un total de 40 points
- Question 4 static class Player — Pour un total de 33 points

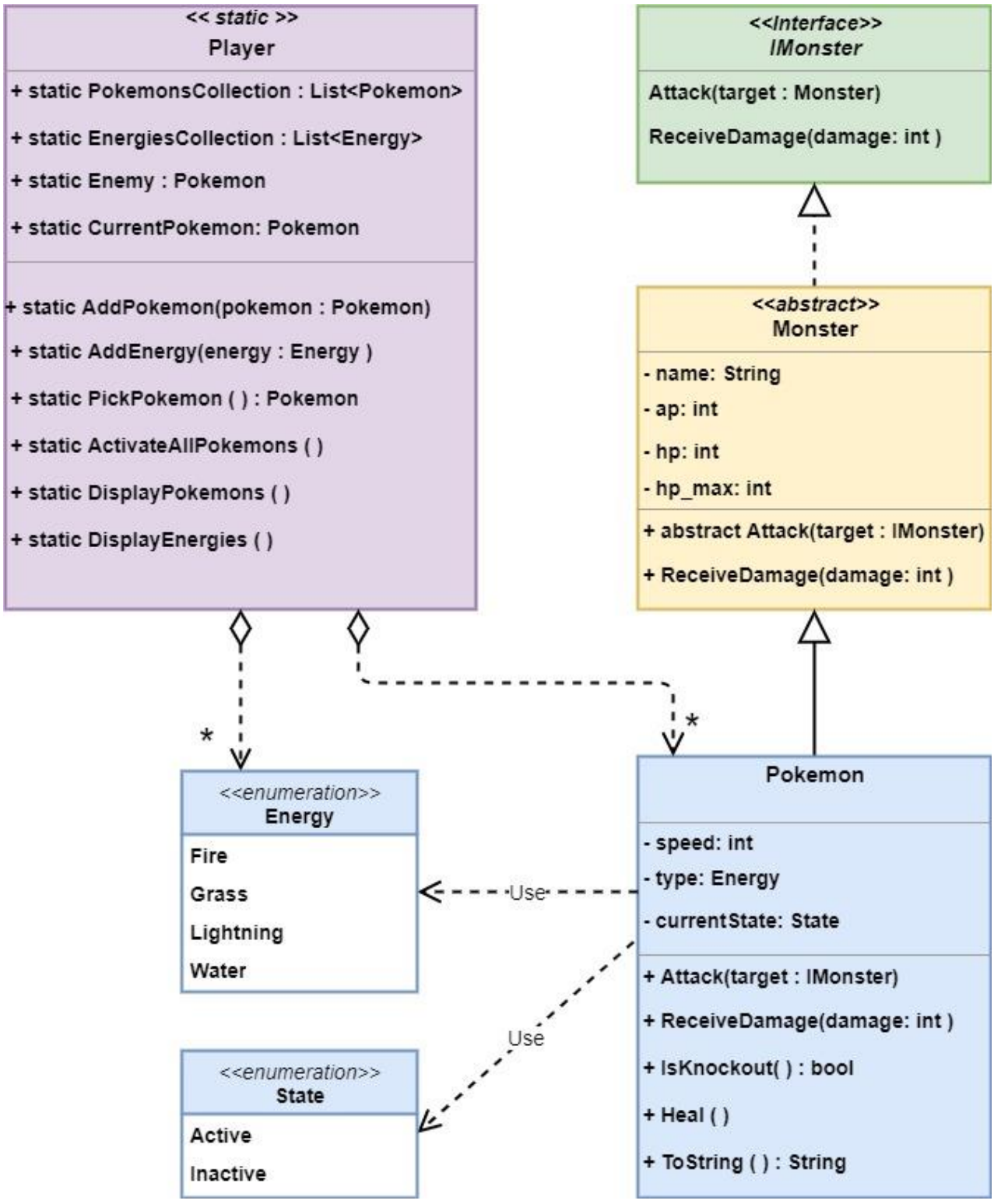
TOTAL: 100 POINTS

Role Playing Game: Pokemon (≧◉ᴥ◉≦)

You have to develop a C# Console Application that implements a text-based Role Playing Game. The main player's mission is to collect wild pokemons in a mysterious forest. Each Pokemon has name, attack power (ap), health points (hp), speed, state (active or inactive) and energy type (Fire, Water, Grass, Lightning). The player will start exploring the forest with one pokemon (Pikachu). While exploring the forest, the player will collect energies (Fire, Water, Grass, Lightning), these energies will be used to activate his inactive collected pokemons. The player can meet wild pokemons on his way, he must fight them using his current pokemon to catch them. If the player has more than one pokemon, he will choose his current pokemon from his collection, such as the current pokemon should be the fastest active pokemon (highest speed and active state) in the collection, because the fastest pokemon will start the attack. After the battle, if the hp of the wild pokemon is less or equals zero, the player wins the battle and catch the wild pokemon as inactive. If the hp of player's current pokemon is less or equals zero, than of the player's pokemon is knock out and his state change to inactive. If the player meets another wild pokemon and he does not have any active pokemon, the game will be over.

Jeu de rôle : Pokemon ʕ•ᴥ•ʔ

Vous devez développer une application console C# qui représente un jeu de rôle en mode texte. La mission du joueur principal est de collecter des pokémons sauvages dans une forêt mystérieuse. Chaque Pokémon a un nom, une puissance d'attaque (ap), des points de vie (hp), une vitesse, un état (actif ou inactif) et un type d'énergie (Feu, Eau, Herbe, Foudre). Le joueur commencera à explorer la forêt avec un pokemon (Pikachu). Tout en explorant la forêt, le joueur collectera des énergies (Feu, Eau, Herbe, Foudre), ces énergies seront utilisées pour activer ses pokémon collectés qui sont inactifs. Le joueur peut rencontrer des pokémons sauvages sur son chemin, il doit les combattre en utilisant son pokémon actuel (son compagnon) pour les attraper. Si le joueur a plus d'un pokémon, il choisira son compagnon de sa collection, tel que : le pokémon actuel doit être actif et le plus rapide (vitesse la plus élevée et état actif) de la collection, car c'est le pokémon le plus rapide qui peut commencer l'attaque. Après la bataille, si le hp du pokemon sauvage est inférieur ou égal à zéro, le joueur gagne la bataille et attrape le pokemon sauvage comme inactif. Si la hp du pokémon compagnon du joueur est inférieure ou égale à zéro, il sera assommé et son état devient inactif. Si le joueur rencontre un autre pokemon sauvage et qu'il n'a aucun pokemon actif, le jeu est terminé.


<< static >>
Player
+ static PokemonsCollection : List<Pokemon>
+ static EnergiesCollection : List<Energy>
+ static Enemy : Pokemon
+ static CurrentPokemon: Pokemon
+ static AddPokemon(pokemon : Pokemon)
+ static AddEnergy(energy : Energy)
+ static PickPokemon ( ) : Pokemon
+ static ActivateAllPokemons ( )
+ static DisplayPokemons ( )
+ static DisplayEnergies ( )
<<Interface>>
IMonster
Attack(target : Monster)
ReceiveDamage(damage: int)
<<abstract>>
Monster
- name: String
- ap: int
- hp: int
- hp_max: int
+ abstract Attack(target : IMonster)
+ ReceiveDamage(damage: int)
*
*
<<enumeration>>
Energy
Fire
Grass
Lightning
Water
Use
Pokemon
- speed: int
- type: Energy
- currentState: State
+ Attack(target : IMonster)
+ ReceiveDamage(damage: int)
+ IsKnockout( ) : bool
+ Heal ( )
+ ToString ( ) : String
Use
<<enumeration>>
State
Active
Inactive

Question 1: Interface IMonster (05 points)

Create an interface **IMonster** with the following methods:

- **void ReceiveDamage (int damage);**
- **void Attack(Monster target);**

Question 2: Abstract Class Monster implements IMonster (22 points)

Create an **abstract** class **Monster** that **implements IMonster (02 points)**

- Add the following fields: string name, int hp, int ap, and int hp_max. **(04 points)**
- Encapsulate the fields. **(04 points)**
- Define a constructor with 3 parameters (string name, int ap, int hp), such as hp_max will be initialized by hp. **(05 points)**

Methods

1. Implement the method **public virtual void ReceiveDamage(int damage) { }** that decreases the hp by the given damage. This method should be virtual, because the children of class **Monster** can override this method. **(05 points)**
2. Do not implement the method **void Attack(Monster target);** Keep it abstract. **(02 points)**

Question 3: Class Pokemon inherits Monster (40 points)

- Create a class **Pokemon** inherits from the class Monster **(01 points)**
- Add the following fields: int speed, Energy type, State currentState. **(03 points)**
- Encapsulate the field. **(03 points)**
- Define a constructor with (string name, int hp, int ap, int hp, int speed, Energy type), such as the currentState will be inactive by default. **(06 points)**

Methods

1. Override the method **public void ReceiveDamage(int damage) { }** that decreases the hp by the given damage. If the hp is less or equals zero, then the currentState will be inactive. **(05 points)**

2. Override the method **public void Attack(Monster target) { }** that checks if the currentState is active. Then, it will call the method ReceiveDamage(ap) of the target. If the currentState is inactive. Then, it will display a message: `I'm not active yet`. **(10 points)**

3. Define the method **public bool IsKnockout( ) { }** that checks if the hp is less or equals to zero. Then, it will change to currentState to inactive and returns true. Else, it returns false. **(10 points)**

4. Define the method **public void Heal( ) { }** that reset the current HP by the value of hp_max. **(02 points)**

Question 4: Class Player (33 points)

Define the following static methods in the static class Player:

1. **public static void AddPokemon(Pokemon pokemon)** that adds the given pokemon to the player's list **`PokemonsCollection`**. **(05 points)**

2. **public static void AddEnergy(Energy energy)** that adds the given energy to the player's list **`EnergiesCollection`**. If the list does not contain that energy type. **(05 points)**

3. **public static void ActivateAllPokemons( )** for each energy type collected in the list **`EnergiesCollection`** , it will activate all the pokemons of the list **`PokemonsCollection`** that are inactive and have the same type as the given energy.**Hint:** You need two nested forech loops **(10 points)**

4. **public static Pokemon PickPokemon( )** it will check all pokemons of the list **`PokemonsCollection`** and return the pokemon of highest speed and Active state. **(13 points)**

Good Luck (>‿◠)

Version FR

Question 1: Interface IMonster (05 points)

Définir l'interface **IMonster** avec les méthodes:

- **void ReceiveDamage (int damage);**
- **void Attack(Monster target);**

Question 2: Abstract Classe Monster qui implemente IMonster (22 points)

Définissez la classe **abstraite Monster** qui **implemente IMonster (02 points)**

- Ajoutez les attributs suivants: string name, int hp, int ap, int hp, and int hp_max. **(04 points)**
- Encapsulez tous les attaributs. **(04 points)**
- Définissez un constructeur avec 3 paramètres (string name, int ap, int hp), tel que hp_max sera initialisé par la même valeur que hp. **(05 points)**

Méthodes

3. Implementez la méthode **public virtual void ReceiveDamage(int damage) { }** qui diminue la la valeur de hp par les dommages reçu en paramètre. Cette méthode devrait être virtuelle, car les enfants de la classe Monster peuvent changer cette méthode. **(05 points)**

4. Ne pas implémenter la méthode **void Attack(Monster target);** gardez la abstraite. **(02 points)**

Question 3: Classe Pokemon hérite de Monster (40 points)

- Définissez la classe **Pokemon** inherits from the class Monster **(01 points)**
- Ajoutez les attributs suivants: int speed, Energy type, State currentState. **(03 points)**
- Encapsulez tous les attaributs. **(03 points)**
- Définissez un constructeur avec les paramètres (string name, int hp, int ap, int hp, int speed, Energy type), tel que le currentState sera inactif par défaut. **(06 points)**

Méthodes

5. Définissez la méthode (override) **public void ReceiveDamage(int damage) { }** qui diminue la valeur hp par les dommages reçus en paramètre. Si le hp est inférieure ou égale à zéro, l'état actuel du pokemon devient inactif. **(05 points)**
6. Définissez la méthode **public void Attack(Monster target) { }** qui vérifie si l'état actuel (currentState) est actif, alors il appellera la méthode ReceiveDamage(ap) de la cible (target). Si le currentState est inactif, alors il affichera un message: `I'm not active yet`. **(10 points)**

7. Définissez la méthode **public bool IsKnockout( ) { }** qui vérifie si la valeur de hp est inférieure ou égale à zéro, alors il change la valeur de currentState à inactif et retourne la valeur vrai. Sinon, il retourne la valeur faux. **(10 points)**

8. Définissez la méthode **public void Heal( ) { }** qui réinitialise le hp actuel par la valeur de hp_max. **(02 points)**

Question 4: Classe Player (33 points)

Définissez les méthodes statiques suivantes dans la classe statique Player :

5. **public static void AddPokemon(Pokemon pokemon)** qui ajoute le pokemon donné à la liste du joueur **`PokemonsCollection.`** **(05 points)**

6. **public static void AddEnergy(Energy energy)** qui ajoute l'énergie reçue en paramètre à la liste du joueur **`EnergiesCollection.`** Si la liste ne contient pas ce type d'énergie. **(05 points)**

7. **public static void ActivateAllPokemons( )** pour chaque type d'énergie collectée dans la liste EnergiesCollection , il activera tous les pokémon de la liste PokemonsCollection qui sont inactifs et qui ont le même type d'énergie. **Astuce :** vous aurez besoin de deux boucles foreach impbriquées **(10 points)**

8. **public static Pokemon PickPokemon( )** il vérifiera tous les pokémon de la liste **`PokemonsCollection`** et retournera le pokémon le plus rapide (grande vitesse) et qui est Actif. **(13 points)**

Bonne chance (>‿◠)