Role Playing Game: Forest Adventure

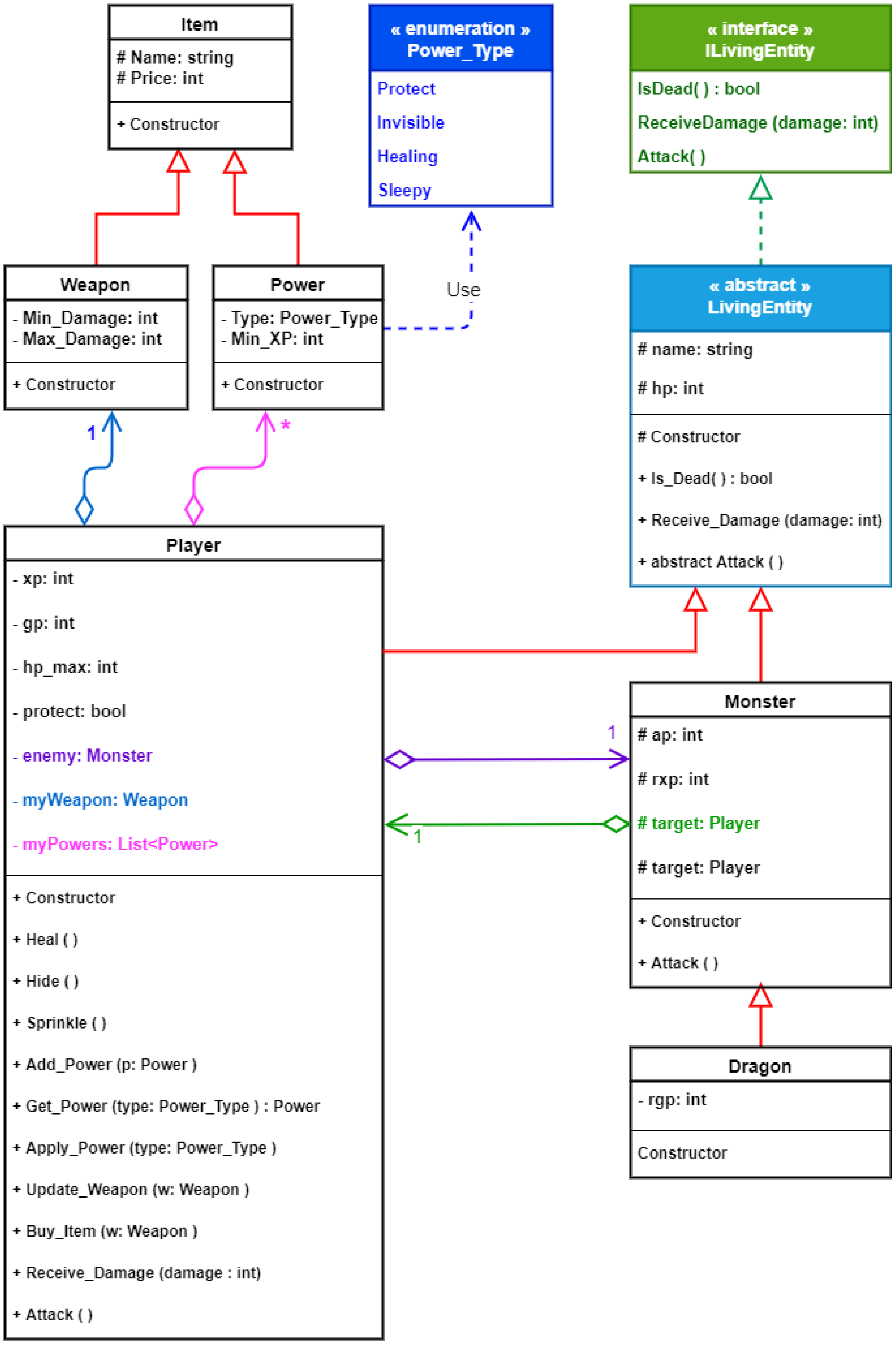
You have to develop a C# Console Application that implements a text-based Role Playing Game. The main player's mission is to collect gold pieces and treasures in a mysterious forest. While exploring the forest, the player will collect gold pieces, weapons, and magic powers, he will also fight against monsters to stay alive.

The player can meet monsters (Goblins, Ogres and Dragons) on his way, he has to kill them to stay alive. The player needs a weapon to kill the monsters. He will be given a wood stick at the beginning of the game, and he can upgrade it by a more efficient one (rock, torch, or magic sword) if he finds it on his way.

The player can also find many items that have magic powers (shields for protecting from the monster's attack, healing potions for healing, magic cape for hiding, sleepy dust, etc), These powers are very useful for his survival, however each power item has a minimum experience (MIN_XP) that determines how much player's **experience** is **required** to be used, ie The player cannot use a power if he has not enough experience. Each item power is used only once; ie the player cannot reuse the same item again. He can also use his gold pieces to buy other powers if he needs them.

This game does not end; the winner is the player having the highest score (gold pieces). If the player dies after a monster attack, the game will be over. As usual, at the beginning of the game the player gives a name to his own personal character, this name should be unique. Because, the player can pause the game and quit the application. The next time the players run the game if he enters the same name, the game will restore his status (score, weapons, powers, etc) to be able to continue without losing his previous score.

The following class diagram represent the classes that you will develop:


Item
# Name: string
# Price: int
+ Constructor
« enumeration »
Power_Type
Protect
Invisible
Healing
Sleepy
« interface »
ILivingEntity
IsDead( ) : bool
ReceiveDamage (damage: int)
Attack( )
Weapon
- Min_Damage: int
- Max_Damage: int
+ Constructor
Power
- Type: Power_Type
- Min_XP: int
+ Constructor
Use
« abstract »
LivingEntity
# name: string
# hp: int
# Constructor
+ Is_Dead( ) : bool
+ Receive_Damage (damage: int)
+ abstract Attack ( )
1
*
Player
- xp: int
- gp: int
- hp_max: int
- protect: bool
- enemy: Monster
- myWeapon: Weapon
- myPowers: List<Power>
+ Constructor
+ Heal ( )
+ Hide ( )
+ Sprinkle ( )
+ Add_Power (p: Power)
+ Get_Power (type: Power_Type) : Power
+ Apply_Power (type: Power_Type)
+ Update_Weapon (w: Weapon)
+ Buy_Item (w: Weapon)
+ Receive_Damage (damage : int)
+ Attack ( )
1
1
Monster
# ap: int
# rxp: int
# target: Player
# target: Player
+ Constructor
+ Attack ( )
Dragon
- rgp: int
Constructor

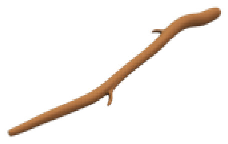
Question 1: Class Item (05 points)

There are two types of collectable items weapons and powers. All these items share common properties.

- Create a **class Item** with the following fields: string name and int price.
- Encapsulate all the fields.
- Define a default constructor and a constructor with 2 parameters (string name, int price)

Question 2: Class Weapon inherits Item (05 points)

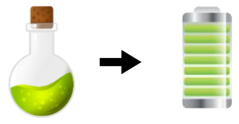
The player needs a weapon to kill the monsters. He will be given a wood stick at the beginning of the game, and he either can upgrade it by a more efficient one if he finds it on his way or he can buy another weapon using his gold pieces.

Name : Wood Stick	**Name : Big Rock**	**Name: Torch**	**Name: Magic Sword**
			
Min Damage = 5	**Min Damage = 20**	**Min Damage = 35**	**Min Damage = 50**
Max Damage = 15	**Max Damage = 30**	**Max Damage = 45**	**Max Damage = 60**
Price : free	**Price : 100 GP**	**Price : 300 GP**	**Price : 500 GP**

- Create a **class Weapon** that **inherits class Item**
- Add the following fields: int minDamage, int maxDamage. Encapsulate all the fields.
- Define a default constructor.
- Define a constructor with 4 parameters (string name, int price, int minDamage, int maxDamage)

Question 3: Class Power inherits Item (05 points)

The player can use those some magic powers to kills the monsters and the dragons. The player can find and collect different objects powers. He can also buy those powers using his gold pieces. Each power has a price, a type that defines his effect, and minimum experience points required, if the player has less experience he cannot use those powers.

Name : Wood Shield	**Name : Magic Cape**	**Name: Healing Potion**	**Name : Sleepy Dust**
			
Type = Protect	**Type = Invisible**	**Type = Healing**	**Type = Sleepy**
Min XP required = 1	**Min XP required = 3**	**Min XP required = 2**	**Min XP required = 6**
Price : 100 GP	**Price : 300 GP**	**Price : 200 GP**	**Price : 600 GP**

- Create **enumeration PowerType** with values: **Protect, Invisible, Healing, Sleepy**
- Create **class Power** that **inherits class Item.**
- Add the following fields: PowerType type and int minXp. Encapsulate all the fields.
- Define a default constructor.
- Define a constructor with 4 parameters (string name, int price, PowerType type, int minXp)

Question 4: Interface ILivingEntity (05 points)

There are different types of living entities (player, monsters and dragons), and they are sharing common behaviors. To ensure that those classes implement all the required methods, you have to create an interface **ILivingEntity** with the following methods:

- **bool IsDead( ) ;**
- **void ReceiveDamage (int damage);**
- **void Attack( );**

Question 5: Abstract Class LivingEntity implements ILivingEntity (05 points)

There are different types of living entities (player, monsters and dragon). All these objects share common properties name and hp, and common methods that are in described in the interface **ILivingEntity.**

- Create an **abstract** class **LivingEntity** that **implements ILivingEntity**
- Add the followings fields: string name and int hp. Encapsulate the fields.
- Define a default constructor
- Define a constructor with 2 parameters (string name, int hp)

Implement the interface methods

- Implement the method **public bool IsDead( ) { }** that returns True if the current HP <= 0.
  The children of class **LivingEntity** cannot override this method.
- Implement the method **public virtual void ReceiveDamage(int damage) { }** that decreases the current HP by the given damage. This method should be virtual, because the children of class **LivingEntity** can override this method.
- Do not implement the method **void Attack( );** Keep it abstract.

Create Class Dice

- Create a class Dice that inherits from class Random. Apply singleton design pattern: make the default constructor private, add a private and static variable instance of type Dice, add a public and static method GetInstance() that check if the instance is null, it will initialize instance using the constructor and then it returns instance.

Question 6: Class Monster inherits LivingEntity (05 points)

There are different types of monsters: Goblins, Ogres and Dragons.

HP (Health Points)	Health Points represents the monster's life energy. Each monster has a max amount of HP. He will lose his energy (HP) every time is attacked by the player. The monster dies if his lose all his HP.
AP (Attack Power) **Ogre** **Goblin**	Attack Power represents the maximum damage received by the player when he gets attacked by the monster. **Dragon** has **HP = 120** and **AP = 40** **Ogre** has **HP = 80** and **AP = 20** **Goblin** has **HP = 40** and **AP = 10**
RXP (Reward Experience Points)	Reward Experience Points represents the experience points that the player will gain after killing the monster. **Dragon** : **RXP = 3** , **Ogre** : **RXP = 2**, **Goblin** : **RXP = 1**
RGP (Reward Gold Pieces)	Reward Gold Pieces represents the amount of gold pieces that the player will gain after killing the dragon. Only dragons are treasure guardians, the player will obtain their gold pieces after killing them. Dragon has **RGP = Random** value between **1000** and **10000.** If the player sprinkles a magic powder (sleepy dust), the monsters will fall asleep. If the monster is a dragon, the player will take his reward gold pieces and run away.

- Create a class **Monster** inherits from the abstract **class LivingEntity**
- Add the following fields: int ap, int rxp, Player target. Encapsulate all the fields.
- Define a default constructor
- Define a constructor with 4 parameters (string name, int hp, int ap, int rxp)
- Override the method **public void Attack( ) { }** that uses a random damage number between 0 and AP(Attack Power) of the monster. Hint: You have to use the instance of class Dice to generate a random damage: int random = Dice.GetInstance().Next(0, AP); Then, you have to apply the random damage on the target (Player), by calling the method ReceiveDamage(random) of the target player.

Question 7: Class Dragon inherits Monster (05 points)

- Create a class Dragon inherits from the class Monster
- Add the following fields: int rgp. Encapsulate the field.
- Define a default constructor

- Define a constructor with 4 parameters (string name, int hp, int ap, int rxp), the **rpg** will be initialized by a random value between 1000 and 10000.

Question 8: Class Player inherits LivingEntity (Total 55 points)

Just like in your favorite video game, the player has different “stats” that affect your success in the game.

HP (Health Points)	Health Points represents the player’s life energy. The player has a max amount of HP. He will lose his energy (HP) every time he get attacked by a monster. The player can use the Healing power to restore his HP. The player dies if he loses all his HP.
XP (Experience Points)	Experience Points are a measure of your progress in the game. The player gains one XP every time he kills a Monster. The more XP you gain, the more you’ll level up and get to learn cool new powers; So the primary focus when you first start playing should be gaining experience points.
GP (Gold Pieces)	Gold Pieces are collected by the player while he is exploring the forest. The winner of the game is the player with the highest GP.

Question 8-1: Constructors and Fields of class Player (05 points)

- Create a class **Player** inherits the abstract **class LivingEntity**, with following fields: int xp, int gp, int hp_max, bool protect, Monster enemy, List<Power> myPowers, Weapon myWeapon.
- Encapsulate all the fields.
- Define a default constructor and a Define a constructor with 2 parameters (string name, int hp). Don’t forget to initialize the field xp and gp by 0, the field protect by false, the object myWeapon by an instance of Weapon (using data of the weapon wood stick), and the list myPowers by an empty list (using the default constructor).

Question 8-2: Methods for implementing power effects (10 points)

- Define the method **public void Heal( ) { }** that reset the current HP by the value of HpMax.
- Define the method **public void Hide( ) { }** that reset the enemy to null to make the player run away.
- Define the method **public void Sprinkle( ) { }** If the player sprinkles a magic powder (sleepy dust), his enemy falls asleep. If the enemy is a dragon, the player will take his reward gold pieces and then call the method Hide () to make the player run away. If the enemy is a monster (ogre or goblin) he will just run away.

Question 8-3: Methods for managing the powers of Player (10 points)

- Define the method **public void AddPower (Power newPower) { }** to add the given object newPower to the list myPowers.
- Define the method **public Power GetPower (PowerType type) { }** to find the power object of the given type in the list myPowers, if the object is found it returns it, else it returns null.
- Define the method **public void ApplyPower (PowerType type) { }** that calls the method **GetPower** (type) to find the power object of the given type in the list myPowers; If the object is found, it will check if the player has enough experience points to use it; if yes, it will remove that object from the list myPowers (because each power can be used only once) and then it applies the power effect:
  - If the type = PowerType.Healing: you have to call the method Heal( ).
  - If the type = PowerType.Invisible: you have to call the method Hide( ).
  - If the type = PowerType.Protect: you have to set the field protect to true.
  - If the type = PowerType.Sleepy: you have to call the method Sprinkle( )

Question 8-4: Methods for buying new weapons and powers (15 points)

A. Define the method **public void UpdateWeapon (Weapon aWeapon) { }** that checks if the maxDamage of the myWeapon is less than the maxDamage of the aWeapon, then it updates myWeapon by the aWeapon.

B. Define the method **public void BuyItem (Item newItem) { }** First, you have to check if the player has enough GP (gold pieces are greater or equal to the price of newItem), then you update the player's GP (decreased by the price of the newItem). Finally, in order to update player's inventory, you have to check the type of newItem. If newItem is instance of class Weapon, then you have to update myWeapon by the aWeapon. If newItem is instance of class Power, then you have to cast it into Power and add it to the list of powers using the method AddPower (Power newPower).

Question 8-5: Methods for Battle (Total 15 points)

A. Override the method **public void ReceiveDamage (int damage) { }** that checks if the property protect is true. If yes, it decreases the current HP by the 50% of damage. If the property protect is false, then it decreases the current HP by damage.

B. Override the method **public void Attack( ) { }** that generate a random damage between minDamage and maxDamage of player's weapon. Then, you have to apply that random damage on the enemy, by calling the method ReceiveDamage(random) of the enemy. Then, you have to check if the enemy is dead. If yes, you have to upgrade the **XP** (experience points) of player using the **RXP** (reward experience points) of the enemy, and if the enemy is instance of **Dragon**, the player will update his **GP** using the **RGP** (reward gold pieces) of the dragon. Hint: You have to cast the enemy into Dragon

Question 9: Integration of all classes with GameManager (10 points)

To answer this question, you have to download the project template RPG_Game.zip from Omnivox, then integrate all your classes that you implemented in the above questions. After integration the game will be ready to use. The project RPG_Game includes the following classes:

- **Static Class GameFactory:** that defines many static methods to create the game elements: weapons, powers, monsters, etc.
- **Static Class Message:** that defines three static methods Danger(string message) to display a red message on the screen, Warning(string message) to display a yellow message on the screen and finally Succes(string message) to display a green message on the screen.
- **Static DataXML:** that defines three static methods **Save(string fileName, List<Player> list)** that uses an XML Serializer in order to *Serialize* the content of the given List<Player> into an XML file.The method **public List<Player> Load(string fileName, string path)** that uses an XML Serializer in order to *Deserialize* the content of the given XML file into a List<Player>.

W d n Static Class GameManager: that defines three static methods:Explore(), StartBattle() and GameOver(), StartGame(string name) that load the list of players from XML file. It will check if the given player name already exist. If yes, it will set the object currentPlayer of GameManager. Else, it will create a new Player with the given name and HP = 100.

Question 11: Class Library (Bonus 10 points)

Create a class Library C# with all the above classes. Then, building your project to generate .DLL file. Create an empty Console Application and copy the same class Program (including Main( ) method) from the project template RPG_Game. Add the .DLL file as a new reference and then compile and execute. The bonus part should be submitted as a separate assignment on LEA. It is very important that you submit the Full source code of the above questions as Final Evaluation Assignment