CECS 277 – Project 2

Pokémon - Part 2

Modify the code you wrote for Project 1 (fix all errors reported from the rubric). Add in the following new features to your game:

1. Gym – The finish is now blocked by a Pokémon gym and requires the Trainer to defeat the gym leader to move to the next level.
    a. The gym leader has a single random pokémon to defeat (level +2).
    b. The encounter is similar to a wild pokémon encounter, but the user should not have the option of throwing a poké ball or running away.
    c. If the Trainer fails (all pokémon's hp = 0), then the encounter ends, and the user can repeat again when they're ready.
2. Max 6 pokémon – the maximum number of pokémon the Trainer can have is 6.
    a. When the Trainer catches a new pokémon, check the size of the Trainer's pokémon ArrayList, if the size is 7, prompt the user to choose from a list of each of the pokémon to remove.
3. Buffs/Debuffs – pokémon can now be buffed and debuffed to increase and decrease the pokémon's stats. Trainers can buff their pokémon by using potions (full heal plus a random buff on the chosen pokémon), or by reaching the next level (after the gym leader is defeated, all pokémon in the Trainer's ArrayList get buffed). Pokémon will be debuffed randomly during fights (25% chance a trainer's pokémon will debuff an enemy pokémon, and a 10% chance an enemy pokémon will debuff the trainer's pokémon).
    a. Buffs: AttackUp – increases a pokémon's damage by 1-2 and adds '+ATK' to their name. HpUp – increases a pokémon's hp and maxHp by 1-2 and adds '+HP' to their name.
    b. Debuffs: AttackDown – decreases the pokémon's damage by 1 and adds a '-ATK' to their name. HpDown – decreases a pokémon's hp and maxHp by 1 and adds '-HP' to their name.

Updates to the Pokémon class:

1. getAttackTypeMenu – returns the basic and special attack menu string.
2. getNumAttackTypeMenuItems – returns 2.
3. getType – returns the integer corresponding to the elemental type of the pokémon for use in the battle table (Fire = 0, Water = 1, Grass = 2).
4. getAttackMenu – returns the menu listing the options of that pokémon's moves (ex. slam/tackle/punch). This is overridden in the base pokémon classes for the special elemental attacks (based on atkType).
5. getNumAttackMenuItems – returns the number of moves in the above menu. This is overridden in the base pokémon classes for the special elemental attacks.
6. getAttackString – returns the partial string for the chosen move (ex. "SLAMMED" if choice was 1 for the basic attack). This is overridden in the base pokémon classes for the special elemental attacks.
7. getAttackDamage – returns the randomized damage for the chosen move. Also overridden in the base pokémon classes for special elemental attacks.
8. getAttackMultiplier – returns the attack multiplier for that class's moves (ex. for elemental attacks it'll return the result of the battle table). This can be overridden

in the base pokémon classes for the special elemental attacks and/or in the decoration buff/debuff classes.

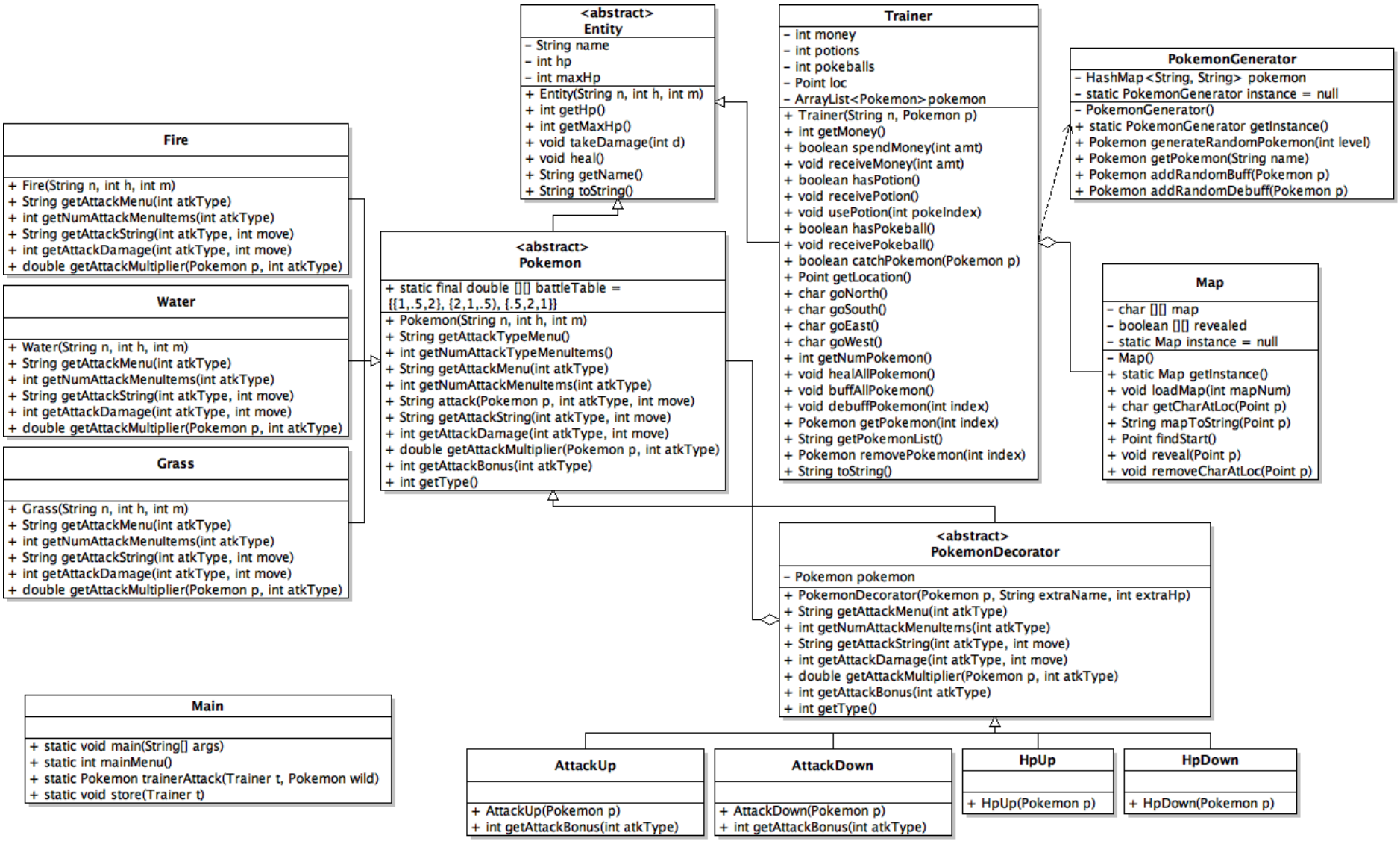
9. getAttackBonus – returns the attack bonus that will be added to the calculated damage. This can be overridden in the base pokémon classes for the special elemental attacks and/or in the decoration buff/debuff classes.
10. attack – calculates the total damage, deals it to the defending pokémon, and builds the full attack string that will be returned to be displayed during a fight (ex. "Oddish is SLAMMED by Charmander and takes 7 damage").

Incorporate the following design patterns to help improve the design of the program:

1. Singleton:
    a. Map – is now a singleton, so there is only one map instance for the whole program. This means it can be removed as the Trainer's instance variable and replaced when needed using the getInstance method.
    b. Pokémon Generator – is a singleton. Use the getInstance method whenever you need to create a new pokémon or to buff a pokémon.
2. Decorator
    a. Pokémon Decorator – will have base pokémon types of Fire, Water, or Grass. Those base types will then be decorated with different types of buffs and debuffs to update the pokémon's stats.
3. Factory Method
    a. Pokémon Generator – constructor should read from the file into a HashMap to store the different pokémon names and their associated elemental type.
    b. Methods are used to generate random or specific pokémon and to buff/debuff pokémon.
        i. generateRandomPokemon should randomly pick a pokémon from the HashMap, and then construct a pokémon of the corresponding elemental base type. Then for each level greater than one, repeatedly decorate it with a random buff (ex. a level 3 pokémon might be a Fire pokémon decorated with +ATK and +HP), this will cause the constructed pokémon to gain an additional title, increase its hp and maxHp, and/or do additional damage.
        ii. getPokemon passes in a string with the name of a pokémon and constructs an object of the correct corresponding type.
        iii. addRandomBuff/Debuff randomly chooses a buff/debuff to apply to the pokémon (note: this should not reset the pokémon's hp (ex. if the pokémon's hp was 14/20, then after being buffed with the +HP it should be 15/21).

Notes

- If you have changed groups for this project, then use the code from the student with the highest grade. Fix any errors listed on the graded rubric before starting.
- Give appropriate default values to variables (ie. pokémon should start with 20-25 hp, rather than 100 hp).
- Please do not add any extra instance variables or methods to the UML.
- Ask questions about any methods you do not fully understand. Ask me if you think that there is an error in the description or if you want to add a helper method.


<abstract>
Entity
- String name
- int hp
- int maxHp
+ Entity(String n, int h, int m)
+ int getHp()
+ int getMaxHp()
+ void takeDamage(int d)
+ void heal()
+ String getName()
+ String toString()
Trainer
- int money
- int potions
- int pokeballs
- Point loc
- ArrayList<Pokemon> pokemon
+ Trainer(String n, Pokemon p)
+ int getMoney()
+ boolean spendMoney(int amt)
+ void receiveMoney(int amt)
+ boolean hasPotion()
+ void receivePotion()
+ void usePotion(int pokeIndex)
+ boolean hasPokeball()
+ void receivePokeball()
+ boolean catchPokemon(Pokemon p)
+ Point getLocation()
+ char goNorth()
+ char goSouth()
+ char goEast()
+ char goWest()
+ int getNumPokemon()
+ void healAllPokemon()
+ void buffAllPokemon()
+ void debuffPokemon(int index)
+ Pokemon getPokemon(int index)
+ String getPokemonList()
+ Pokemon removePokemon(int index)
+ String toString()
PokemonGenerator
- HashMap<String, String> pokemon
- static PokemonGenerator instance = null
- PokemonGenerator()
+ static PokemonGenerator getInstance()
+ Pokemon generateRandomPokemon(int level)
+ Pokemon getPokemon(String name)
+ Pokemon addRandomBuff(Pokemon p)
+ Pokemon addRandomDebuff(Pokemon p)
Fire
+ Fire(String n, int h, int m)
+ String getAttackMenu(int atkType)
+ int getNumAttackMenuItems(int atkType)
+ String getAttackString(int atkType, int move)
+ int getAttackDamage(int atkType, int move)
+ double getAttackMultiplier(Pokemon p, int atkType)
<abstract>
Pokemon
+ static final double [][] battleTable =
{{1,.5,2}, {2,1,.5}, {.5,2,1}}
+ Pokemon(String n, int h, int m)
+ String getAttackTypeMenu()
+ int getNumAttackTypeMenuItems()
+ String getAttackMenu(int atkType)
+ int getNumAttackMenuItems(int atkType)
+ String attack(Pokemon p, int atkType, int move)
+ String getAttackString(int atkType, int move)
+ int getAttackDamage(int atkType, int move)
+ double getAttackMultiplier(Pokemon p, int atkType)
+ int getAttackBonus(int atkType)
+ int getType()
Map
- char [][] map
- boolean [][] revealed
- static Map instance = null
- Map()
+ static Map getInstance()
+ void loadMap(int mapNum)
+ char getCharAtLoc(Point p)
+ String mapToString(Point p)
+ Point findStart()
+ void reveal(Point p)
+ void removeCharAtLoc(Point p)
Water
+ Water(String n, int h, int m)
+ String getAttackMenu(int atkType)
+ int getNumAttackMenuItems(int atkType)
+ String getAttackString(int atkType, int move)
+ int getAttackDamage(int atkType, int move)
+ double getAttackMultiplier(Pokemon p, int atkType)
Grass
+ Grass(String n, int h, int m)
+ String getAttackMenu(int atkType)
+ int getNumAttackMenuItems(int atkType)
+ String getAttackString(int atkType, int move)
+ int getAttackDamage(int atkType, int move)
+ double getAttackMultiplier(Pokemon p, int atkType)
<abstract>
PokemonDecorator
- Pokemon pokemon
+ PokemonDecorator(Pokemon p, String extraName, int extraHp)
+ String getAttackMenu(int atkType)
+ int getNumAttackMenuItems(int atkType)
+ String getAttackString(int atkType, int move)
+ int getAttackDamage(int atkType, int move)
+ double getAttackMultiplier(Pokemon p, int atkType)
+ int getAttackBonus(int atkType)
+ int getType()
Main
+ static void main(String[] args)
+ static int mainMenu()
+ static Pokemon trainerAttack(Trainer t, Pokemon wild)
+ static void store(Trainer t)
AttackUp
+ AttackUp(Pokemon p)
+ int getAttackBonus(int atkType)
AttackDown
+ AttackDown(Pokemon p)
+ int getAttackBonus(int atkType)
HpUp
+ HpUp(Pokemon p)
HpDown
+ HpDown(Pokemon p)