BLG252E

THE SECOND ASSIGNMENT

INTRODUCTION

“Magic: The Gathering” is a card game with several card types and occasionally complex rules. In this assignment, according to the given main function, the necessary objects and their methods related with these playing cards will be implemented. At this point, it must be noted that several assumptions about card types and game mechanics are made in order to provide simplicity and accuracy for the assignment. Subjects related with this assignment can be listed as,

- Class Design
- Constructors & Destructors
- Operator Overloading
- Inheritance - Polymorphism

In the rest of the document, you can find the detailed description about the mentioned playing cards and the assignment. It is expected from you to design acceptable objects with optimal usage. In this case, besides proper compilation and execution of your program, there are other evaluation criteria.

CARDS

There are six types of cards in the Magic. However, three of them are considered for this assignment. These types are (i) land, (ii) creature and (iii) enchantment-aura. Each card has a color (white, blue, black, green or red) and a name. Other specific features and description of the cards can be described as follows.

- Land: Each land card provides one mana in their color at each turn. In figure 1, a green land (Forest) card and a black land (Swamp) card are given as instances.

Figure 1. Example Land Cards

- Creature: Creatures are the main elements of the battlefield. Each creature has a cost, attack point, hit point and a description besides its color and name. In figure 2, a blue creature (Mahamoti Djinn) card and a red creature (Balduvian Barbarians) card are given as instances.

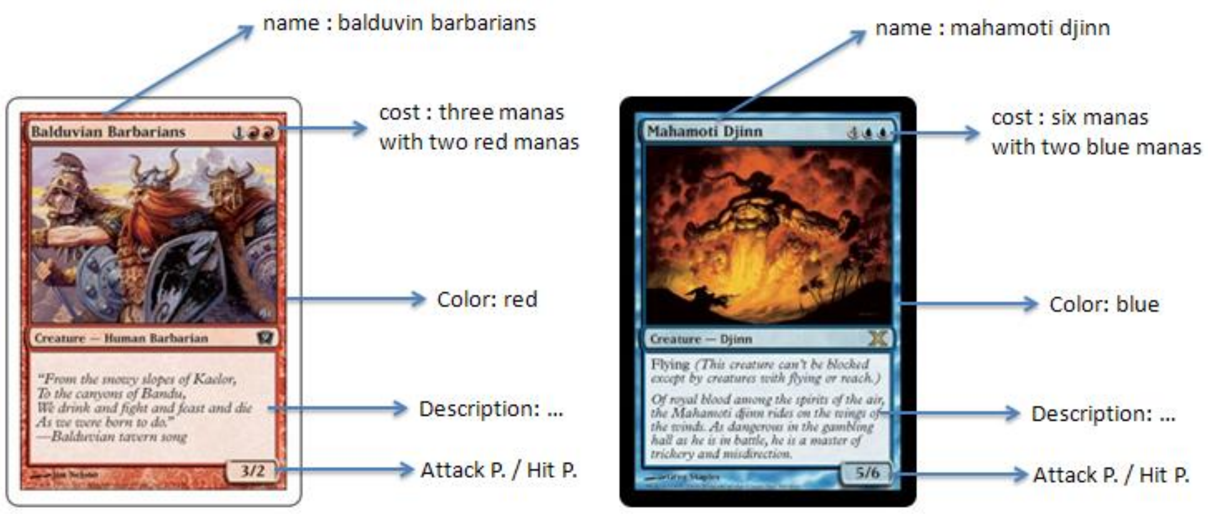

Figure 2. Example Creature Cards

- Enchantment – Aura: This type of cards can be attached onto creatures and provide them additional specifications. Besides name and color, enchantment cards have description, effect and cost features. In figure 3, a white enchantment (Holy Strength) card and a green enchantment (Treetop Bracers) card are given as instances. <u>You may assume that enchantment cards only changes the attack power or hit point of the creature.</u>

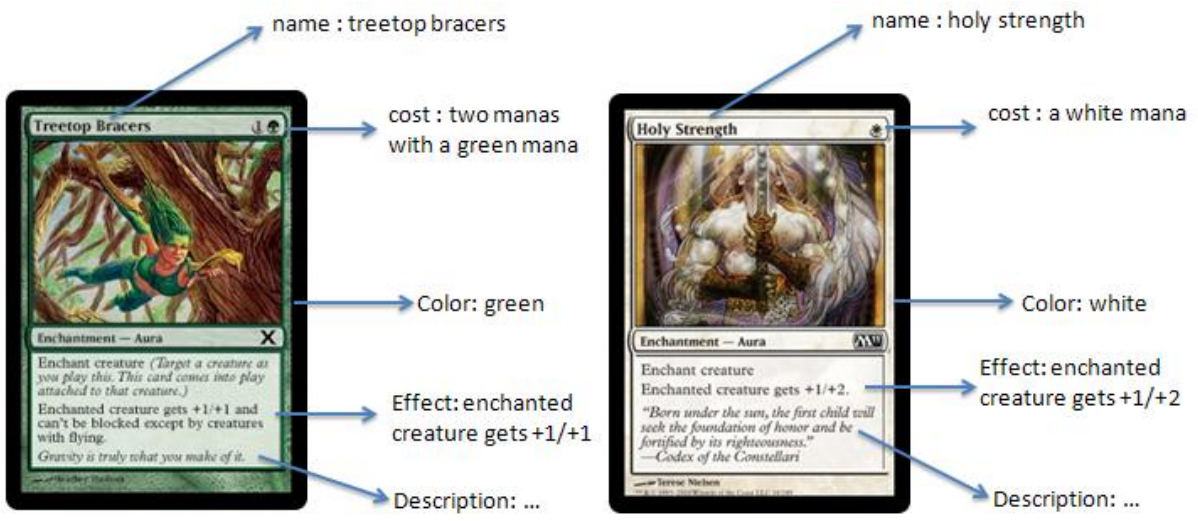

Figure 3. Example Enchantment Cards

TEST CODE

Be sure that your implementation fulfills the necessary methods. Also <u>according to your needs, you can implement other methods or overloading functions in your submission.</u> In the test code, <u>you may assume that enchantments can be attached onto creatures only in same color with them.</u>

```
int main() {
    srand((unsigned)time(0));
    Land l1("black", "snow covered swamp");
    Land l2("blue", "island");
    l1.print();
    // for the following array, default constructor is called 5 times
    // let's randomly initialize land cards in the default constructor srand called at the beginning of the code
    Land l3[5];
    // cost object: we need total 3 manas, one of them must be
    // red mana (the last one) color of other two manas are unimportant
    // order of the parameters: total, white, blue, black, green, red
    Cost *cost = new Cost(3,0,0,0,0,1);
    // parameters : name, color, ap, hp, cost, description-flavor text
    Creature c1("anaba shaman", "red", 2, 2, cost, "just try taking this bull by the horns.");
    delete cost;
    cost = new Cost(5,1,0,0,0,0);
    Creature c2("angel of mercy","white",3,3, cost,"when angel of mercy enters the battlefield, you gain 3 life.");
    delete cost;
    cost = new Cost(1,0,0,0,0,1);
    Enchantment e("firebreathing", "red", 1,0, cost,"enchanted creature gets +1/+0 until end of turn.");
    // when a creature is enchanted, update its attack power and hit point
    if (c2.canEnchantable(e)){
            cout << e.getName() << " is enchantable for " << c2.getName() << endl;
            c2 + e;
    }
    else if (c1.canEnchantable(e)){
            cout << e.getName() << " is not enchantable for " << c2.getName() << endl;
            cout << e.getName() << " is enchantable for " << c1.getName() << endl;
            c1 + e;
    }
    else
            cout << e.getName() << " can't be enchanted to any creature!" << endl;
    if (c1.isAffordable(l3,5)) // if you look closely, you can see that 5 is size :)
            cout << c1.getName() << " is affordable according to land deck l3 " << endl;
    if (e.isAffordable(l3,5))
            cout << e.getName() << " is affordable according to land deck l3" << endl;
    // show time
    Land *l4[8];
    for (int i=0; i<5; i++)
            l4[i]=&l3[i];
    l4[5]=&c1;
    l4[6]=&c2;
    l4[7]=&e;
    for (int i=0; i<8; i++)
            l4[i]->print();
    return 0;
}
```

"Land" object has a two-parameter constructor and a default constructor with respect to the test code above. In the default constructor, determine land card types randomly. You can use values in table 1 for class variables.

Color	**Name**
white	plains
blue	island
black	swarm
green	forest
Red	mountain

Table 1. Land card types

"print" method is common for all classes. On the other hand, you have to implement "isAffordable" method for creature and enchantment objects, similarly "canEnchantable" method for creature object. All other methods and variables are quite understandable according to the given main function. Don't forget to consider the assumptions above. I underlined all those assumptions, by the way.

LAST WORDS

This assignment affects 5% of your grade. Plagiarism and any other forms of cheating will have serious consequences like the first assignment. If you have any questions about homework, you can ask it to me via mail (ataka@itu.edu.tr), through ninova or directly (office no: 4308 except for Wednesdays).

P.s. Be careful when copying the main function in this document to your source code. Some signs such as " can be transformed into a different character.

Ahmet Aycan ATAK

06.04.2012