Lab 05 - Polymorphism

Instructions:

- A battle card game contains two card types: creature and spell. Each card type contains a name, cost, and ability. Additionally, a creature card contains attack, defensive, mana, and life; and a spell card contains duration, user, and target. Your objective is to define several classes that form a hierarchy and override methods in a header file named ‘`GameCard.h`’.
- ‘`GameCard.h`’ must contain a header guard.
- The classes must be defined within a namespace named ‘`oopl`’.
- ‘`GameCard.h`’ can only include the libraries *iostream*, *string*, *sstream*, *cctype*, *cmath*, and *iomanip*.
- Each class, except for interfaces, must define all its special member functions (default constructor, copy constructor, destructor, and assignment operator). Destructors must be empty unless stated otherwise and virtual for abstract classes. The copy constructors and assignment operators must perform shallow copies.
- Each method, excluding special member functions and friends, must include pseudocode as a comment above it to receive any credit.
- Your submissions must be submitted to the GitHub repository in the Lab05 directory.
- Cheating of any kind is prohibited and will not be tolerated.
- Violating or failing to follow any of the rules above will result in an automatic zero (0) for the lab.

Grading

Task	Maximum Points	Points Earned
1	0.25	
2	0.95	
3	0.95	
4	0.95	
5	0.95	
6	0.95	
Total	5.00	

Note: solutions will be provided for tasks colored blue only.

Task 1

- Creates an interface named *Object* that contains
  - ☐ a public string pure virtual constant method named `toString()` that takes no parameters.
  - ☐ a friend ostream operator that takes returns a display in the same format as `toString()`.

Task 2

- Create an abstract class named *Card* that publicly inherits *Object* and contains
  - ☐ a private string field named *id*.
  - ☐ a private unsigned integer field named *value*.
  - ☐ a public default constructor that assigns an empty string and 0 to *id* and *value*, respectively.
  - ☐ a public string constant method named `name()` that takes no parameters and returns *id*.
  - ☐ a protected void method named `name()` that takes a string parameter and assigns the parameter to *id*.
  - ☐ a public unsigned integer constant method named `cost()` that takes no parameters and returns *value*.
  - ☐ a protected void method named `cost()` that takes an unsigned integer parameter and assigns the parameter to *value*.
  - ☐ a public void pure virtual method named `ability()` that takes no parameters.
  - ☐ a public overridden `toString()` method that returns a string in the format

    "*n* (*c*)"

    where *n* and *c* are the values of *id* and *value*, respectively.

Task 3

- Create an abstract class named *Creature* that publicly inherits *Card* and contains
  - ☐ a private unsigned integer array field named *aspects* with a size of 5.
  - ☐ a public default constructor that assigns 0 to each element of *aspects*.
  - ☐ a public unsigned integer constant method named `attack()` that takes no parameters and returns first element of *aspects*.
  - ☐ a protected void method named `attack()` that takes an unsigned integer parameter and assigns the parameter to the first element of *aspects*.
  - ☐ a public unsigned integer constant method named `defense()` that takes no parameters and returns second element of *aspects*.
  - ☐ a protected void method named `defense()` that takes an unsigned integer parameter and assigns the parameter to the second element of *aspects*.
  - ☐ a public unsigned integer constant method named `mana()` that takes no parameters and returns third element of *aspects*.
  - ☐ a protected void method named `mana()` that takes an unsigned integer parameter and assigns the parameter to the third element of *aspects*.
  - ☐ a public unsigned integer constant method named `life()` that takes no parameters and returns fourth element of *aspects*.
  - ☐ a protected void method named `life()` that takes an unsigned integer parameter and assigns the parameter to the fourth element of *aspects*.
  - ☐ a public unsigned integer constant method named `damage()` that takes no parameters and returns fifth element of *aspects*.
  - ☐ a public void method named `damage()` that takes an unsigned integer parameter and assigns the parameter to the fifth element of *aspects* only if it does not exceed the value of the fourth element of *aspects*.
  - ☐ a public overridden `toString()` method that returns a string in the format

    "*n* (*c*) LP:*l*][*m*\nATK:*a*/DEF:*d*"

    where *n* is *id*, *c* is *value*, *l* is the forth minus the fifth elements of *aspects*, *m* is the third element of *aspects*, *a* is the first element of *aspects*, and *d* is the second element of *aspects*.

Task 4

- Create an abstract class named *Spell* that publicly inherits *Card* and contains
  - ☐ a private *Creature* pointer array field named *sources* with a size of 2.
  - ☐ a private unsigned integer array field named *lengths* with a size of 2.
  - ☐ a public default constructor that assigns null to each element of *sources* and 0 to each element of *lengths*.
  - ☐ a public void virtual method named `user()` that takes a *Creature* pointer parameter and assigns the parameter to the first element of *sources*.
  - ☐ a public void virtual method named `target()` that takes a *Creature* pointer parameter and assigns the parameter to the second element of *sources*.
  - ☐ a protected *Creature* pointer method named `user()` that no parameters and returns the first element of *sources*.
  - ☐ a protected *Creature* pointer method named `target()` that no parameters and returns the second element of *sources*.
  - ☐ a public unsigned integer constant method named `uses()` that takes no parameters and returns second element of *lengths*.
  - ☐ a protected void method named `uses()` that takes an unsigned integer parameter and assigns the parameter to the second element of *lengths* only if does not exceed the first element of *lengths*.
  - ☐ a public unsigned integer constant method named `duration()` that takes no parameters and returns first element of *lengths*.
  - ☐ a protected void method named `duration()` that takes an unsigned integer parameter and assigns the parameter to the first element of *lengths*.

Task 5

- Create a class named *Warrior* that publicly inherits *Creature* and contains
  - ☐ a private unsigned integer field named *boost*.
  - ☐ a private Boolean field named *active*.
  - ☐ a public default constructor that uses the default *Creature* values, assigns 0 to *boost* and false to *active*.
  - ☐ a public overloaded constructor that takes a string parameter and six unsigned integer parameters, respectively. It assigns the string parameter to *id*, the integer parameters to *value*, the first four elements of *aspects*, and *boost*, respectively, 0 to the fifth element of *aspects*, and false to *active*
  - ☐ a public overridden `ability()` method. If *active* is false and the fifth element of *aspects* is greater than 0, it adds *boost* to the first element of *aspects* and assigns true to *active*. And if the fifth element of *aspects* is zero and *active* is true, it removes *boost* from the first element of *aspects* and assigns false to *active*.

Task 6

- Create a class named *Restoration* that publicly inherits *Spell* and contains
  - ☐ a private unsigned integer field named *restore*.
  - ☐ a public default constructor that uses the default *Spell* values and assigns 0 to *restore*.
  - ☐ a public overloaded constructor that takes a string parameter and three unsigned integer parameters, respectively. It assigns the string parameter to *id*, the integer parameters to *value*, the first element of *lengths*, and *restore*, respectively, and 0 to the second element of *lengths*.
  - ☐ a public overridden `user()` that assigns the parameter to all elements of *sources*.
  - ☐ a public overridden `target()` that assigns the parameter to all elements of *sources*.
  - ☐ a public overridden `ability()` method. If the first element of *sources* is not null and the elements of *lengths* are not equal, it reduces the fifth element of *aspects* of the first element of *sources* by *restore* and increments the second element of *lengths* by 1. If the elements of *lengths* become equal, assign null to the elements of *sources* and 0 to the second element of *lengths*.