Lab 04 - Generics

Instructions:

- **In many situations, selecting a few distinct entities from a larger collection is necessary. Your objective is to define a generic class named *Selection*, along with additional classes and functions, in a header file named 'Selections.h'.**
- **The header file must contain a header guard.**
- **The classes must be defined within a namespace named 'dsl'.**
- **The header file can only include the libraries *iostream*, *string*, *sstream*, *cstdlib*, *ctime*, and *cctype*.**
- **Each method excluding special member functions must include pseudocode as a comment above it to receive any credit.**
- **Your submissions must be submitted to the GitHub repository in the Lab04 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	0.50	
2	2.00	
3	2.00	
4	0.50	
Total	5.00	

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

Task 1

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

Task 2

- Define the generic class *Selection* that publicly inherits *Object* and contains
  - ☐ a private generic pointer array field named *deck* with a size of 5.
  - ☐ a public default constructor that assigns null to each element of *deck*.
  - ☐ a public empty destructor.
  - ☐ a public copy constructor that performs a shallow copy.
  - ☐ a public overloaded assignment operator that performs a shallow copy.
  - ☐ a public generic pointer method named `get()` that takes an integer parameter. It returns the element of *deck* whose index matches the parameter if the parameter is valid (in range [0,4]); otherwise, it returns null.
  - ☐ a public void method named `set()` that takes an integer parameter and a generic pointer parameter. It assigns the generic pointer parameter to the element of *deck* whose index matches the integer parameter; otherwise, it does nothing.
  - ☐ a public overridden `toString()` method that returns a string of a list of nonnull elements of *deck* each on a separate line preceded by `"idx:\n"` where *idx* is the element's index.

Task 3

- Define the class *Fighter* that publicly inherits *Object* and contains
  - ☐ a private string field named *_name*.
  - ☐ a private integer field named *_attack*.
  - ☐ a private integer field named *_life*.
  - ☐ a private integer field named *_damage*.
  - ☐ a private deleted default constructor.
  - ☐ a private static method named `genName()` that takes no parameters and returns a randomly generated string composed of an uppercase letter concatenated to a four-digit number.
  - ☐ a public overloaded constructor that takes two integer parameters and assigns an invocation of `genName()`, the first parameter, second parameter, and 0 to *_name*, *_life*, *_attack*, and *_damage*, respectively.
  - ☐ a public copy constructor.
  - ☐ a public assignment operator.
  - ☐ a public empty destructor.
  - ☐ a public constant getter method for *_name* named `name()`.
  - ☐ a public constant getter method for *_attack* named `attack()`.
  - ☐ a public constant getter method for *_life* named `life()`.
  - ☐ a public constant getter method for *_damage* named `damage`.
  - ☐ a public Boolean method named `hit()` that takes an integer parameter. It increments *_damage* by the parameter and returns true only if the parameter is positive.
  - ☐ a public void method named `reset()` that takes no parameters and assigns 0 to *_damage*.

- ☐ a public Boolean constant method named `defeated()` that takes no parameters and return true if *_damage* is greater or equal to *_life*; otherwise, it returns false.
- ☐ A public overridden `toString()` method that returns a string in the format

  `"n (l) [a>:<d]"`

  where *n*, *l*, *a*, and *d* are the values of *_name*, *_life*, *_attack* and *_damage*, respectively.

Task 4

- Create a cpp file named '`main.cpp`' that
  - ☐ define a *Fighter* function named `GenerateFigther()` that takes no parameters. It creates and returns a *Fighter* object whose *_life* and *_attack* fields are assigned randomly generated multiples of 500 in the range [500,10,000).
  - ☐ declares and initializes a *Fighter* array of size 100 using the `GenerateFighter()` function, declares and initializes a *Selection* array of size 2 with randomly elements of the *Fighter* array, and displays the elements of both arrays in the main function.