Lab 05 - Factory Method Design

Instructions:

- **The factory method is a creational design pattern that provides an interface for creating object fields in a base class superclass whose type is determined by a derived class.**
- **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 and/or failing to follow any of the rules will result in an automatic zero (0) for the lab.**

Grading

Task	Name	Maximum Points	Points Earned
1		2.0	
2		1.5	
3		1.5	
Total		5	

Task 1

Create a header file named `"KeyPad.h"` that defines an interface named *Encoder* within the namespace *LB5* that must contain

- a public string pure virtual constant method named `Encipher()` that takes a string parameter.

and defines an abstract class named *KeyPad* within the namespace *LB5* that must contain

- a private string field named *contents*.
- a public default constructor that assigns the empty string to *contents*.
- a public copy constructor.
- a public assignment operator.
- a public destructor.
- a public *Encoder* pointer pure virtual method named `CreateEncoder()` that takes no parameters.
- a public void method named `AddValue()` that takes a char parameter. It appends the parameter to the end of *contents* only if the parameter is a digit character.
- a public string virtual constant method named `ToString()` that takes no parameters. It creates a *Encoder* pointer object and returns the return of the `Encipher()` method of the *Encoder* object with *contents* as its argument. However, it deallocates the *Encoder* object before the return of the function.
- a friend overloaded ostream operator that returns its output in the same format as `ToString()`.

Task 2

Create header file named `"Encoders.h"` that define the class *ShiftEncoder* that publicly inherit *Encoder* within the namespace *LB5* and must contain

- a public overridden `Encipher()` that returns a string such that each character of the returned string equals

$$c_i = \texttt{char('0' + ((p}_i \texttt{ - '0') + 4) \% 10)}$$

  where c_i and p_i are the i element of the returned string and parameter respectively.

and define the class *AffineEncoder* that publicly inherit *Encoder* within the namespace *LB5* and must contain

- a public overridden `Encipher()` that returns a string such that each character of the returned string equals

$$c_i = \texttt{char('0' + (7 * (p}_i \texttt{ - '0') + 2) \% 10)}$$

  where c_i and p_i are the i element of the returned string and parameter respectively.

Task 3

Create header file named `"EncodedKeyPads.h"` that defines the class *CXKeyPad* that publicly inherit *KeyPad* within the namespace *LB5* and must contain

- a public overridden `CreateEncoder()` method that returns a dynamically allocated *ShiftEncoder* object.
- a friend overloaded ostream operator that returns its output in the same format as `ToString()`.

and defines the class *NTKeyPad* that publicly inherit *KeyPad* within the namespace *LB5* and must contain

- a public overridden `CreateEncoder()` method that returns a dynamically allocated *AffineEncoder* object.
- a friend overloaded ostream operator that returns its output in the same format as `ToString()`.

Extra Credit

Create a header file named `"Extra.h"` define an abstract class named *Character* that must contain

- a private string field named *name*.
- a private int field named *type*.
- a private int field named *offense*.
- a private int field named *defense*.
- a public default constructor that assigns `"name"`, 0, 100, and 100 to *name*, *type*, *offense*, and *defense* respectively.
- a public copy constructor.
- a public assignment operator.
- a public empty destructor.
- a public constant getter method for each field.
- a public setter method for *name* that only assigns the parameter to *name* if the parameter consists only of letters and spaces.
- a public setter method for *type* that only assigns the parameter to *type* if the parameter is between 1 and 50 inclusively.
- a public setter method for both *offense* and *defense* that only assigns the parameter to the field if the parameter is positive and at most 5000.
- a public string virtual constant method named `ToString()` that takes no parameters. It returns a string in the format

  $$[x(y): z/w]$$

  where x, y, z and w are the values of the fields *name*, *type*, *offense* and *defense* respectively.
- a public *Character* pointer pure virtual constant method named `Clone()` that takes no parameters.
- a friend overloaded ostream operator that returns its output in the same format as `ToString()`.

(2 points)