# CS 3005: Programming in C++

### **Factories**

## Introduction

In object oriented programming, a factory is a class that can create instances of other classes. It is a common pattern to create a factory that is capable of constructing an instance from a class within a class hierarchy. The object returned is a polymorphic instance of the base class.

Read more at the <u>Wikipedia article</u> on the factory method.

### Assignment

In this assignment, you will create factory classes for the waveform and envelope hierarchies.

enum

An enumeration ( $\underline{enum}$  in C++) is a user defined type allowing the user to specify symbols to represent a group of related unique constants.

The syntax is as follows:

enum TypeNameYouDeclare { CONSTANT1, CONSTANT2, ... };

This declares a new type TypeNameYouDeclare, with possible values CONSTANT1, CONSTANT2, etc.

### **Class Data Members**

A class data member is declared with the static modifier. For example:

```
class X {
public:
    const static int ONE;
};
```

A class data member is initialized like a global variable in the implementation file. For example:

```
const static int X::ONE = 1;
```

### **Class Methods**

A class method can be called from the class, or from an instance of the class. It is allowed to use class data members, but there is no this pointer to an instance, so instance data members and methods are not accessible.

Simple example of declaration in the header file:

```
class X {
public:
   static int add(int a, int b);
};
```

Then in the implementation file:

```
static int X::add(int a, int b) {
    // not allowed to access instance data members
    return a + b;
}
```

# **Programming Requirements**

**Create** [library-waveform/WaveformFactory.{h,cpp}]

#### WaveformFactory Class

### **Data Members:**

The WaveformFactory class will not have any private data members. The class will not need to be instantiated.

#### public Class Data Members:

• const static std::vector<std::string> WaveformName; This vector stores the names of all waveforms that can be created. The names must be entered in the same order as the constants in WaveformId. The names are "sine", "square", and "error". Class data members are declared in the header file, and initialized in the implementation file. The initialization looks like a global variable initialization. It is outside of any code block.

#### public Enumerations:

• WaveformId needs to have constants WF\_SINE, WF\_SQUARE, WF\_ERROR.

#### public Methods:

- static std::unique\_ptr<Waveform> create(WaveformId id, const std::string& name); Class method to
  create a waveform using a WaveformId. name is passed to the waveform's constructor. If the id is not a
  valid type, the returned pointer should be set to nullptr.
- static std::unique\_ptr<Waveform> create(const std::string& id, const std::string& name); Class method
  to create a waveform using a string from WaveformName to identify the type of waveform. name is passed
  to the waveform's constructor. This method should lookup the correct WaveformId from the id, and call
  the other create method with that WaveformId.
- static WaveformId stringToWaveformId(const std::string& id); Given the string [id], find the
  corresponding WaveformId. Returns WF\_ERROR if the string [id] does not correspond to a known waveform.
- static bool validStringId(const std::string& id); Returns true if id is a valid waveform name, false
  otherwise.
- virtual ~WaveformFactory() = default; Just in case someone instantiates this class. Provide a virtual, empty destructor.

#### **Update** [library-waveform/Makefile]

Add [WaveformFactory. {h, cpp}] in the appropriate places to add them to the library and install the header file.

#### **Create** library-envelope/EnvelopeFactory. {h, cpp}

#### EnvelopeFactory Class

#### **Data Members:**

The EnvelopeFactory class will not have any private data members. The class will not need to be instantiated.

#### **public** Class Data Members:

• const static std::vector<std::string> EnvelopeName; This vector stores the names of all envelopes that can be created. The names must be entered in the same order as the constants in EnvelopeId. The names are "AD", "ADSR", and "error". Class data members are declared in the header file, and initialized in the implementation file. The initialization looks like a global variable initialization. It is outside of any code block.

#### public Enumerations:

• EnvelopeId needs to have constants EN\_AD, EN\_ADSR, EN\_ERROR.

#### public Methods:

• static std::unique\_ptr<Envelope> create(EnvelopeId id, const std::string& name); Class method to

create an envelope using an EnvelopeId. name is passed to the envelope's constructor. If the id is not a valid type, the returned pointer should be set to nullptr.

- static std::unique\_ptr<Envelope> create(const std::string& id, const std::string& name); Class method
  to create an envelope using a string from EnvelopeName to identify the type of envelope. name is passed
  to the envelope's constructor. This method should lookup the correct EnvelopeId from the id, and call
  the other create method with that EnvelopeId.
- static EnvelopeId stringToEnvelopeId(const std::string& id); Given the string id, find the
- corresponding EnvelopeId. Returns EN\_ERROR if the string id does not correspond to a known envelope.
  static bool validStringId(const std::string& id); Returns true if id is a valid envelope name, false otherwise.
- virtual ~EnvelopeFactory() = default; Just in case someone instantiates this class. Provide a virtual, empty destructor.

#### **Update** library-envelope/Makefile

Add [EnvelopeFactory. {h, cpp}] in the appropriate places to add them to the library and install the header file.

### **Additional Documentation**

TBA

# **Grading Instructions**

To receive credit for this assignment:

- your code must be pushed to your repository for this class on GitHub
- all unit tests must pass
- all acceptance tests must pass
- all programs must build, run, and execute as described in the assignment descriptions.

# **Extra Challenges (Not Required)**

TBA