Classes in C++

Michael Brodskiy
Professor: S. Shazli

April 3, 2023

- The `const` modifier forbids the compiler from modifying any variables
  - This is common practice to prevent modification of variables in large code bases
- Getters are also known as accessors
- The `virtual` keyword
- A `static` variable is the same across different instances of a class
- Suppose that:
  - You want to write a function to compare two ints
  - You want to write a function to compare two strings
    - Function overloading
- Polymorphism in C++
  - In C++: `PromisedType* var_p = new ActualType();`
  - C++ has the notion of templates
    - A function or class that accepts a `type` as a parameter
    - Supposing `template <typename T>` is passed to a function, the function may be used by describing the data type passed while passing it:
      - For an int: `function<int>(nameOfInt)`
      - For a double: `function<double>(nameOfDouble)`
      - For a string: `function<std::string>(nameOfString)`
  - The compiler doesn't generate any code when it sees the template function
    - It doesn't know what code to generate yet, since it doesn't know what types are involved
  - When the compiler sees the function being used, it then understands what types are involved
- Templates are useful for classes as well
  - One of the main motivations of templates
  - Imagine we want a class that holds a pair of things that we can set and get the value of, but we don't know what data type the things will be
  - Thing is replaced with template argument when the class is instantiated
    - The class template parameter name is in scope of the template class definition and can be freely used there
    - Class template member functions are template functions with template parameters that match those of the class template

            - These member functions must be defined as template functions outside of the class template definition (if not written inline)
            - The template parameter name does not need to match that used in the template class definition

- Abstract Methods and Classes
    - Sometimes we want to include a function in a class but only implement it in derived classes
        - In Java, we would use an abstract method
        - In C++, we use a "pure virtual" function
        - For example, `string noise() = 0`
        - A class containing any pure virtual methods is abstract
            - You can't create instances of an abstract class
            - Extend abstract classes and override methods to use them
        - A class containing only pure virtual methods is the same as a Java interface
            - Pure type specification without implementations