**CS173: Intermediate Computer Science**

**Reading 10**

Name: \_\_\_\_\_\_\_Daniel Lee\_\_\_\_\_\_\_\_

Read the assigned pages below from our course textbook. Complete the responses to the questions in this document and then save as a docx or pdf file. Submit your work by the assigned deadline on the Canvas course page or in class. Responses may be neatly handwritten or typed. **Put your name at the top!**

Readings: From the course textbook please read Chapter 12 (you can skip the section on GUIs). We will be focused on ADTs and classes for a while, so you do not need to have a crystal clear understanding of the whole chapter right away, but it is good to start thinking about these concepts and come in with questions.

You should come away understanding:

* What an ADT is
* How to define a class
* How to create instances of a class
* Why we separate the specification from the implementation in an ADT

**1) Define Abstract Data Type (ADT).**

A data type whose properties (domain and operations) are specified independently of any particular implementation.

**2) Which is most important in the ADT: *how* or *what*? Explain.**

The most important aspect in the ADT is on the what, not the how. For example, with respect to automobiles, most of us know what the engine does, but fewer of us know precisely how the engine works. Abstraction allows us to drive cars without having to be automotive engineers. In the world of software, abstraction is essential for managing immense, complex projects.

**3) What role does the *data representation* play in an ADT implementation?**

It plays the role of representing the abstract values of an abstract data type.

**4) There are generally five categories of methods associated with an ADT. List the five different kinds of methods and give a brief description of each kind (see blue box on Page 682).**

Constructor: An operation that initializes a new instance (variable) of an ADT.

Transformer: An operation that changes the value of the ADT; also known as a mutator.

Observer: An operation that allows us to observe the state of an instance of an ADT without changing it; also known as an accessor.

Destructor: An operation that cleans up the state of an ADT instance just prior to releasing its storage for reuse.

Iterator: An operation that allows us to process—one at a time—all the components

**5) How is a C++ class different from an ADT? How are they related?**

Unlike ADT, C++ class is a data type in a programming language that is used to represent an abstract data type. The relation between ADT and C++ class is that C++ class is a way to represent the data values and operations specified on the ADT.

**6) How is a C++ class different from a C struct?**

In a struct, members are public by default; in a class, they are private by default.

**7) What do the *private* and *public* access modifiers do to class methods and data? When do we generally use public and when do we generally use private in creating a C++ class?**

Class members declared following public and before private constitute the public interface; clients can access these members directly. Class members declared after the word private are not directly accessible to clients. If client code attempts to access a private member, the compiler gives an error message.

Because information hiding is so fundamental to data abstraction, we generally use private for data variables, and the functions that manipulate them are public.

**8) How is a C++ class different from an object of that class? How are they related?**

A class defines a type, and a class is a pattern for creating values of that type. With a class, the values are objects. We say that we instantiate the class to make an object. An object is an instance (concrete example) of its class. We can understand this relationship between class and object as a blueprint and a instance created from that blueprint.

**9) Discuss how *information hiding* plays a role in class design and implementation.**

Information hiding plays a role in ensuring that internal private components of a class are not modified by external client codes and be modified only through public interface, promoting secure code and preventing unconscious modification of these private member variables.

**10) What is the difference between the *specifications file* and the *implementation file* in class design? Why have two different files for a single class?**

In C++, we can optionally place the class declaration and the class implementation in separate files. One file—the specification file—is a header (.h ) file containing only the class declaration. The second file—the implementation file—contains the function definitions.

The main purpose of this is to simplify the programmer's job by reducing complexity. Users of an abstraction should not have to look at its implementation to learn how to use it, nor should they have to write programs that depend on implementation details.