Reference: Big C++.

Exercises 5:

Exercise R5.1.

* Time class (Real-world entities) & (Mathematical abstractions)
* Graphical class (System service)
* Employee class (Real-world entities)
* Product class (Real-world entities)

Exercise R5.2.

The interface of a class is the container of the declarations of member functions and constructors, in other words, an interface is a description of the actions that an object can do, but it does not contain the data fields if they were private.

Meanwhile the implementation of a class is the definitions of those member functions and the definitions of variables and objects used in this class.

Exercise R5.3.

A member function is a function belongs to the class and it is called with a dot notation by an implicit parameter that calls it.

A non-member function is a function, which does not belong to any class. It has only explicit parameters and it is not called by a dot notation since it has no objects.

Exercise R5.4.

A mutator function is a member function that have access to modify the data fields, and this accessibility is given to it by removing the keyword *const* from its definition.

On the other hand, an accessor function is a member function that can read the data fields but it cannot modify them, and that’s thanks to the keyword *const*.

Exercise R5.5.

What happens if I forget the const in an accessor function is I could forget and modify a data field even if I should not do it. See common error 5.3.

However, when I supply a const keyword to a mutator function, then I will not be able to modify the implicit parameter and the compiler gives error before even the compilation starts.

Exercise R5.6.

The implicit parameter is the parameter that calls the function and every member function must have exactly one implicit parameter.

The explicit parameter is the input parameter to the function and it is listed in the function definition, and a member function may have zero or more explicit parameters.

Exercise R5.7.

* Member functions can have exactly one implicit parameter, no more no less.
* Nonmember functions have zero implicit parameters.
* A function can have zero or more explicit parameters.

Exercise R5.8.

A constructor is used to initialize objects when they are created. A constructor with no parameters is called a default constructor.

Exercise R5.9.

A default constructor is a constructor, which initializes the data fields with a certain values, so that every time a new object exist, it will initialize it.

The consequences of a class that doesn’t have a default constructor is that some variables will be initialized with random values and if the user of the class tried to operate on them before initializing them, an error will pop up.

Exercise R5.10.

* A class can have limitless number of constructors but in one condition, that all of them have different parameters.
* Yes, I can, but it is not preferred.
* The one that matches the parameters in the call.

Exercise R5.11.

It is not understandable.

Exercise R5.12.

Member functions are declared in the public interface of a class just like the nonmember functions do.

They are defined as follows:

First type the return type whether it was int or std::string or whatever

Second type the name of the class followed by two colons (::) and then the name of the function and then the function arguments and then the body of the function.

Exercise R5.13.

Encapsulation is the act of hiding implementation details.

It is important because it gives the author of the class the freedom to do anything he wants with the internals of the class, as long as they do not change its behavior. This is thanks to the fact that they can be sure that no one depends on these details, because they are private. The fewer dependencies there are in the code the more flexible and easier to maintain it is. <source: Stackoverflow, URL: <https://goo.gl/qwaJWN>>

Encapsulation also is important because:

As you know, encapsulation means hiding internal details of your class from the end user ( entity that uses your class).

Now I will give you a real life example.

Suppose a doctor John gives patient 3 different powders and tells him that every time he is supposed to mix 3 mg of white powder, 2 mg of red and 4 mg of green, combine them and then take them with glass of water.

On other hand, Doctor Smith gives a patient a simple 'Capsule' that contains the same amount of medicines. Now it’s not botheration to patient to know what is actually inside the capsule or go through the hassle of measuring the content.

So using a 'Capsule' made it really convenient to end user (here patient)

in similar way, when writing a class, if you expose how you are storing data, which data structure you are using to end user (here end user is the class that uses your class), then it is just unnecessary details you are flooding the end user with. You should give him simple way to use your class without exposing the complex details.

<source: Quara, URL: <https://goo.gl/a7w2x5>>

Exercise R5.14.

It hides the implementation details from the users of the class.

Exercise R5.15.

No, it should not, because the less implementation detail you reveal, the more flexibility you have to improve the class.

Exercise R5.16.

You will need to make three additional accessor functions, get\_name(), get\_score(), get\_price().

Exercise R5.18.

In a nonmember function, when you call a member function, you call it with the dot notation. i.e. the object calls it. And when you call a nonmember function you don’t use that dot notation.

In a member function, you call the two functions in the same way, without a dot notation, because when you call a member function from a member function, it means that it’s object is the implicit parameter.

Exercise R5.19.

The implicit parameter is always passed by reference unless the keyword const was declared then you will not be able to change it; an explicit parameter is passed by reference when it is declared with the ampersand notation (&).