**Assignment 5**

**Chapters 11**

1. What are the two kinds of abstractions in programming languages?

Data and process abstraction.

1. Define Abstract Data Type.

The Abstract Data Type can be defined as a type for objects whose behavior is defined by a set of value and a set of operations (subprograms).

1. From where are C++ Objects allocated?

Stack, heap

1. What is the purpose of a C++ Constructor?

Function to initialize the data members of instances, and allocate storage if part of the object is heap-dynamic.

1. What is the purpose of a C++ Destructor?

Function to cleanup after an instance is destroyed, usually just to reclaim heap storage.

1. Where are all Java Methods defined?

Heap

1. Why does Java not have destructors?

Java does not have destructor because Java has its own garbage collector which can free resources

1. Fill in the table for C++ access control. What is the access specification for X, Y, Z defined in A as shown below, in class B, C, and D.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Base Class A | Class B : public A | | | Class C : protected A | | | Class D : private A | | |
| Public : int X;  Protected : int Y;  Private : int Z; | (1) X is | **public** | in B | (4) X is | **protected** | in C | (7) X is | **priivate** | in D |
| (2) Y is | **protected** | in B | (5) Y is | **protected** | in C | (8) Y is | **private** | in D |
| (3) Z is | **Not accessible** | in B | (6) Z is | **Not accessible** | in C | (9) Z is | **Not accessible** | in D |
|  |  |  |  |  |  |  |  |  |

1. Explain why naming encapsulations are important for developing large programs.

When developing large program, to have naming encapsulations can make programmer to change their code without breaking others’ code. Programmer should always hide implementation details, like private their design key variables and create other public methods for other to use. In this case, in a large program, I can use the public method create by others without bother another code inside others’ function.

1. Describe the fundamental difference between C# structs and its classes.

In C#, the structs are public by default, but its classes are private by default. They also have syntactical differences, such that struct has no reference type, and class has no value type. The structs should be very small types, and should be immutable in C#

1. Discuss the advantages of C# properties, relative to writing accessor methods in C++ and Java.

The advantages of C# properties relate to accessor methods in C++ and Java are:

In C#, the getter method can provide read-only access function with there is no corresponding setter method created.

The constraints can be included in the setter.

The actual implementation of data member can be changed without affecting the clients if getters and setters are the only access.

1. Describe some of the benefits to information hiding.

Information hiding allows an implementation to be hidden behind an interface that doesn't change even if the implementation does. In this case, the object-oriented software is usually more stable interface.

Information hiding also has the ability to prevent certain aspects of a function to be accessible to its clients. In the large program, the concept of encapsulation often used interchangeably with information hiding.

Chapter 12

1. Compare the multiple inheritance of C++ with that provided by interfaces in Java. Provide one programming situation where multiple inheritance has a significant advantage over interfaces?

C++ inheritance is implementation inheritance. Example like a class inheriting from two of more superclasses actually inherits the code from those classes. Java’s interface mechanism is an interface inheritance. Example like a class implementing two or more interfaces simply inherits the methods of the interface.

1. Describe the differences between single inheritance and multiple inheritance.

Single inheritance, the derived class inherits only a single base class

Multiple inheritance can inherits more than one base classes.

1. What is a polymorphic variable?

The polymorphic variable is a variable that can hold values of different types during the course of execution.

1. Compare dynamic binding of C++ and Java.

In C++, a method can only be dynamically bound if all of its ancestors are marked virtual. Otherwise, the binding is static by default. In Java, method binding is dynamic by default. Static binding only occurs if the method is marked final.

1. Explain the two problems with abstract data types that are ameliorated by inheritance.

The first benefit using inheritance with abstract data types is one won’t have to copy/paste his code from one data type to another, which will increase the readability in general. The second benefit is that a method can accept a certain class as an argument, and get a child class of this one. This will allow the user to have a wider set of functionality.

1. Describe the issue of how closely the parameters of an overriding method must match those of the method it overrides.

It must match the order, parameter and type as exactly same.

1. What is the purpose of the finalize clause in Java?

When the garbage collector start reclaim the storage occupied by the object, the finalize clause will be called implicitly in Java.

1. What is an Abstract Method? What is a an Abstract Class?

Abstract method has nothing in it.

Abstract class containing abstract method

1. What is the message protocol of an object?

Message protocol is the entire collection of methods of an object.

1. What is a nesting class?

Nesting class is the class which the new class is nested.

1. Explain Dynamic Dispatch.

Dynamic dispatch is the process of selecting which implementation of a polymorphic operation to call at [run time](https://en.wikipedia.org/wiki/Run_time_(program_lifecycle_phase)). It is commonly employed in, and considered a prime characteristic of, [object-oriented programming](https://en.wikipedia.org/wiki/Object-oriented_programming) (OOP) languages and systems

1. What exactly does it mean for a subclass to have an is-a relationship with its parent class?

Is -a relationship actually states a subclass inheritance with its parent class. The subclass has all the public method in the parent class. In a other words, every class in is-a relationship derives from object class, so every class is called an object in object-oriented program.