**Design Pattern Assignment 1:**

**Name: Alazar Alemayehu Id: ATR/4624/06**

1. Discuss about the following OOP concepts
2. **Sub class and Sub-type**

There are important differences between subtypes and subclasses in supporting reuse. Subclasses allow one to reuse the code inside classes - both instance variable declarations and method definitions. Thus they are useful in supporting code reuse *inside* a class. Subtyping on the other hand is useful in supporting reuse externally, giving rise to a form of polymorphism. That is, once a data type is determined to be a subtype of another, any function or procedure that could be applied to elements of the super type can also be applied to elements of the subtype.

Notice that the subtype relation depends only on the public interfaces of objects, not their implementations. In particular, if one type is a subtype of another, it is not necessary for objects of those types to have arisen from classes that are in the subclass relation. By the same argument, objects of the same type need not have arisen from the same class. They may have been generated by classes with different collections of (hidden) instance variables and method bodies, but whose (visible) methods have the same types. This is similar to the impact of information hiding in programming languages which support abstract data types.

1. **Classes and Interfaces**

**Classes** are blue print for Object they represent. They define an abstract datatype, with its particular fields and methods. Each object is an instance of a class, and follows the class prototype that defines the variables and methods common to all objects of a certain kind. Each instance of a class must be instantiated, after it is declared. E.g., Class Person defines all the person in this world with the attribute and the function it does. Therefore, when it defines the class Person in Java we use the following

Class Person {

Int leg; int ear; int eye;

Int hand; String sex;

Public void eats(){}

Public void runs(){}

}

In the above example we can see that the class person have the attribute defined as fields and the operation he does defined as function.

When it is instantiated we use the key word new: i.e. Person person1 = new Person();

**An abstract class** is like a class, but differs in that it only defines "generic" methods in the class, but does not actually implement them. Those methods will be implemented later by specialized sub-classes. This is how we get interfaces - interfaces are the implementation of the abstract class.

**An interface**, or protocol as it is sometimes called, is a device that is used to allow unrelated objects to interact with one another, by implementing an agreed upon system of behavior. When a class implements an interface, the class agrees to implement all of the methods defined in the interface. Interfaces are useful since they capture similarity between unrelated objects without forcing a class relationship. Furthermore, interfaces may consist of either abstract methods or entire abstract classes. One class uses an interface by using the **"implements"** keyword

1. **User Defined Datatypes**

**User defined data types** are those that user / programmer himself defines. For example, classes, interfaces.

In very-very simple words we can say,

int a

Here a is a variable of int data type.

MyClass obj

Here obj is a variable of data type MyClass and we call them reference variables as they can be used to store the reference to the object of that class.5

1. **Classifying types and Operations**

Basically there are three type of data types:

1. Primitive data types
2. Derived data types
3. User defined data types

**Primitive data types** are the general and fundamental data types that we have in Java and those are byte, short, int, long, float, double, char, Boolean.

**Derived data types** are those that are made by using any other data type for example, arrays.

As described in C section user defined data types are types those types that the programmer defines. For example, Classes and Interfaces.

Basically operations are classified based on criteria:

Criteria 1: Implementation: Based on this operation are classified into abstract and concrete data types. Concrete operations are operations whose implementation is done while abstract operations are operations which haven’t defined operations and they only have template.

Criteria 2: Return Type: based on this criteria operation are defined as void and non-void operations. Void operations return nothing while non-void operations return some kind of data.