Lab9 CIS43 Due: 7/13/2016

Name: Nikhil Vytla

***Exercise: Ex 10.8 and 10.10***



Abstract Class: Class containing one or more abstract methods (Only used as superclasses in inheritance hierarchies).

Interface: Reference type in Java – a collection of abstract methods that allows you to tell objects *what* to do, but not *how* to do it.

Similarities & Differences

|  |  |
| --- | --- |
| Abstract Class | Interface |
| * Supports both concrete *and* abstract methods * Can support public, private, and protected members * Uses keyword **extends** * Class can only extend one abstract class * Cannot be instantiated * Can extend from a class or an abstract class * Actually a CLASS | * Only supports abstract methods * Only supports public methods * Variables are limited to constants, must be static and final * Mandatory implementation of all methods. * Uses keyword **implements** * Class can implement multiple interfaces * Can extend only from an interface * NOT a CLASS, more of a CONTRACT |
| ***Uses*** | ***Uses*** |
| * Single inheritance – it provides a common base that can be used for implementation to derived classes * Declaring non-public members – increases security of program * Ease in modifying/adding methods – no need to change the entire class when adding methods * Providing default implementation to the abstract method | * Multiple inheritance – like behavior – classes can implement multiple interfaces, so the interface allows for a situation not far from multiple inheritance * Exactly implement all the abstract methods to the class implementing the interface and no default body to a method can be provided |



Functional Interface: An interface with a single abstract method (called the *functional method*).

* Can be implemented and instantiated using anonymous class syntax
* Provide target types for lambda expressions, method references, and in multiple contexts

(such as assignment context, method invocation, or cast context).