River Crane

Professor Conrad

CIS 18A

30 January 2020

Abstract Classes and Interfaces

Abstract classes are a useful part of computer programming which help promote software reusability, save time, and reduce errors. An abstract class is used to become a superclass for other classes in a program. The abstract class will contain methods, but will not have objects. The subclasses which inherit from the abstract class will provide a body and objects for a program to use. Abstract classes are made for other classes to inherit from and have a similar structure. Software reusability is promoted from this because many classes can use the same base class to create multiple concrete classes with specialized purposes for an application. Time is saved when using abstract classes because they promote software readability. A great example would be a programmer understanding where concrete classes named pickup, semi, and tow originated from. With an abstract superclass named truck, all three of those subclasses would come from the same place. Programs with easy to read inheritance hierarchies help save time when creating and reviewing code. Errors are less encountered when using abstract classes, as written code appears more logical. Abstract classes will point out only the common attributes among subclasses, so a programmer will be able to easily identify where subclasses belong, and troubleshoot where an error is located. The errors will most likely be connected to a concrete subclass and its abstract superclass. In java an abstract class will be extended, whereas an interface will implement. Interfaces identify how different parts of a program can be interacted with, and contain abstract methods and constants only. Interfaces will present the capabilities of a program and what a user can do with them, but they will not display how the program operates internally. Interfaces are beneficial when specific tasks need to be performed within an application, and then return a selected output. Abstract classes and interfaces have demonstrated practical ways programmers can benefit from using them. They are building blocks to creating efficient programs with great readability, efficiency, and reusability.