**CSCI 2302**

**Abstract Classes & Interface Chapter**

**Abstract Class & Interface Lab**

Contents

[Intro 1](#_Toc85572163)

[Notes 1](#_Toc85572164)

[Interfaces 1](#_Toc85572165)

[Learning Goals 4](#_Toc85572166)

[Task 4](#_Toc85572167)

[Submit 5](#_Toc85572168)

Intro: Abstract classes and Interfaces allow another way to design classes and provides flexibility in the maintenance of classes.

Notes:

An abstract class contains abstract methods, which are implemented in concrete subclasses. An abstract class is a superclass that contains common features of its subclasses. The abstract method cannot be defined in the superclass because each subclass has its own specific way that method is to be implemented/defined.

The purpose of an abstract class is to provide a common definition of a base class that multiple derived classes can share.

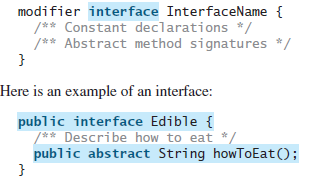
Abstract classes are a way of organizing a program. You can get the same thing done without using this way; it is a matter of program design. The advantage of using an abstract class is that you can group several related classes together as siblings. Grouping classes together is important in keeping a program organized and understandable.

The abstract class and inheritance collectively ensure that most of the code are written using abstract and higher-level classes, so that it can leverage Inheritance and Polymorphism to support future changes.

It also eliminates the need for testing and casting to get to methods/behaviors of the actual subtype. Since it is declared in the superclass and defined in the subclass, the JVM (during dynamic binding) knows which class to invoke the method.

Interfaces are a class-like constructs (they are not classes!) that specify common behaviors/methods between objects/ classes that are related *or* unrelated and can only contain constants and abstract methods. Interfaces specify what an object/class must do, not how to do it = class abstraction.

An interface is treated like a special class in Java. Each interface is compiled into a separate bytecode file, just like a regular class. You can use an interface more or less the same way you use an abstract class. For example, you can use an interface as a data type for a reference variable, as the result of casting, and so on. As with an abstract class, you cannot create an instance from an interface using the new operator.



The benefit is for full abstraction, and you can implement more than one interface at a time. Interfaces allow loose coupling.

Something you need to keep in mind:

All fields/states/variables have to be public, static, and final.

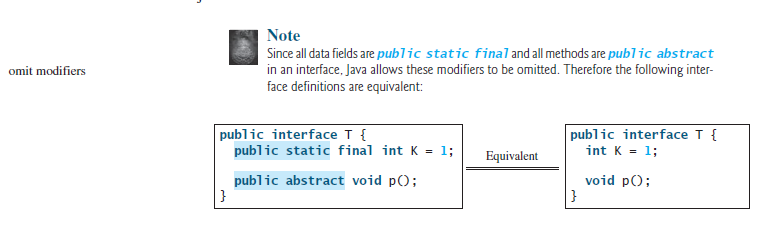
Interfaces had a big change in JDK 9.

Prior to JDK 9:

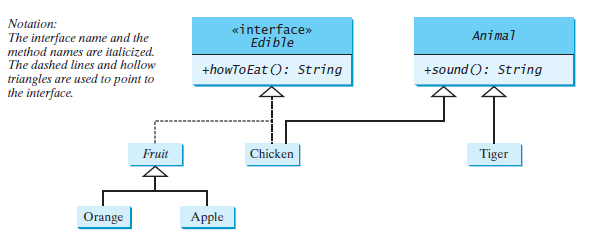
* interface methods had to be public AND abstract

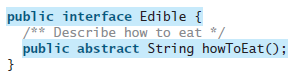
After JDK 9 and onwards:

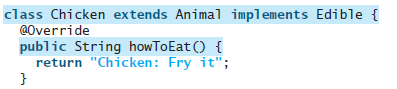
* interface methods can also be static, private, and/or private static
* by default if there is no visibility modifier specified, it will be assigned by the compiler with public

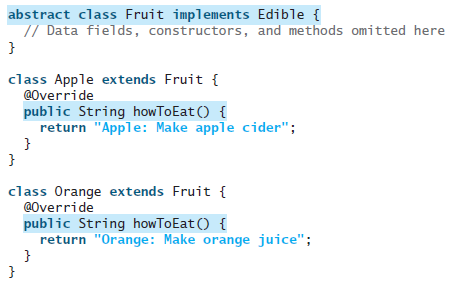


How to use an interface? By using the key word implements. You can implement as many interfaces as you want. (Remember that Java only has single inheritance and can only extend one class at a time.)

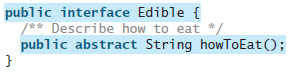








When you implement/define the method in the object/class that is implementing the interface, you override the abstract method.





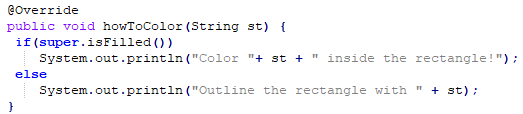
If you implements an interface – you *HAVE TO* implement/define the method.

Learning Goals: To integrate Abstract classes and abstract methods into our understanding about class/object design. To assess the advantages of using abstract classes.

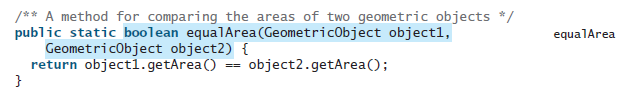
Task: Complete the following steps. Download the files in the Dropbox and save ACI \_Lab.java with mysfaUsername\_ ACI\_Lab.java.

1. Implement/Define an interface named Colorable with a method, howToColor(String st) which has a void return.
2. Modify Circle.java, Rectangle.java and the Triangle.java to implement the howToColor method.

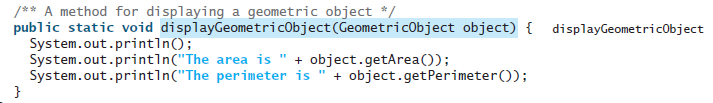
e.g. in the Rectangle class:



1. Modify the mysfaUsername\_ACI\_Lab.java to complete the following:
2. Implement the equalArea method in the I\_Lab.java



1. Implement the displayGeometricObject method in the the mysfaUsername\_ACI \_Lab.java



1. Instantiate the following objects: a Circle, a Rectangle, and a Triangle.
2. Invoke the howToColor method for each object.

Example:



1. Invoke the equalArea method three times: comparing the Circle to the Rectangle, the Circle to the Triangle, and the Rectangle to the Triangle.



1. Invoke the displayGeometricObject method for each object



Submit: Submit the mysfaUsername\_ACI\_Lab.java(that has the Colorable interface) in the Dropbox in Brightspace by D2L.