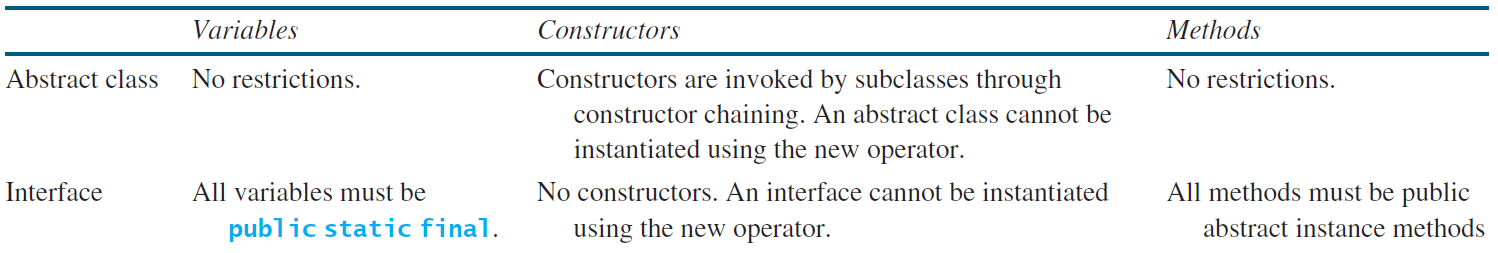
CS 150 Topics List

Chapter 13

1. Abstract Classes (13.2)
   1. Have the *abstract* modifier.
   2. Can’t be instantiated.
   3. Are used to store generic attributes and actions for other classes to inherit.
   4. They force consistency in their subclasses.
2. Abstract methods (13.2)
   1. Have the *abstract* modifier.
   2. Are not implemented where they are first defined; they contain only the header.
   3. They are placeholders for methods that will be overridden in subclasses.
   4. They usually define methods that cannot logically be implemented in the abstract class.
3. Rules for abstract classes: (13.2.2)
   1. An abstract method cannot be contained in a nonabstract class. If a subclass of an abstract superclass does not implement all the abstract methods, the subclass must be defined abstract. In other words, in a nonabstract subclass extended from an abstract class, all the abstract methods must be implemented, even if they are not used in the subclass.
   2. An abstract class cannot be instantiated using the new operator, but you can still define its constructors, which are invoked in the constructors of its subclasses. For instance, the constructors of GeometricObject are invoked in the Circle class and the Rectangle class.
   3. A class that contains abstract methods must be abstract. However, it is possible to define an abstract class that contains no abstract methods. In this case, you cannot create instances of the class using the new operator. This class is used as a base class for defining a new subclass.
   4. A subclass can be abstract even if its superclass is concrete. For example, the Object class is concrete, but its subclasses, such as GeometricObject, may be abstract.
   5. A subclass can override a method from its superclass to define it abstract. This is rare, but useful when the implementation of the method in the superclass becomes invalid in the subclass. In this case, the subclass must be defined abstract.
   6. You cannot create an instance from an abstract class using the new operator, but an abstract class can be used as a data type. Therefore, the following statement, which creates an array whose elements are of GeometricObject type, is correct.

Example: GeometricObject[] geo = new GeometricObject[10];

This array will then refer to Circle and/or Rectangle objects.

1. Interfaces (13.5)
   1. An interface is a classlike construct that contains only constants and abstract methods.
   2. They are like classes, but different.
      1. All their members are either *public static final* for data, or *public abstract* for methods. (Remember, abstract methods are not implemented).
      2. They behave similiarly to abstract classes, but they cannot have any implemented components that are not constants.
      3. They are not inherited (subclassed), but are *implemented*.
      4. A class that implements an interface inherits the interface’s constants and must either implement the interface’s methods or be declared as abstract.
2. Rules for interfaces (From http://www.tutorialspoint.com/java/java\_interfaces.htm):
   1. Checked exceptions should not be declared on implementation methods other than the ones declared by the interface method or subclasses of those declared by the interface method.
   2. The signature of the interface method and the same return type or subtype should be maintained when overriding the methods.
   3. An implementation class itself can be abstract and if so interface methods need not be implemented.
   4. A class can implement more than one interface at a time.
   5. A class can extend only one class, but implement many interfaces.
   6. An interface can extend another interface, similarly to the way that a class can extend another class.
3. Comparable<E> interface(13.7)
   1. Must implement the compareTo(E o) method.
   2. Lets you implement a way to compare two objects.
   3. Uses Generics, which we will talk about in Chapter 19.
4. Cloneable interface(13.8)
   1. Is an empty interface (a marker interface) that simply tells you that the class has a feature.
   2. The specific feature in this case is that objects of the class can be cloned (copied) using the clone() method defined in the Object class.
   3. A Cloneable class must override the clone() method.
5. Interfaces vs Abstract classes (13.8)
   1. Interfaces are preferred over abstract classes because they are more flexible.
6. Class Design Guidelines (13.10)
   1. There are 8 small sections here that you should read: 13.10.1 – 13.10.8