Chapter 8 CRT

1. A has-a relationship means a class uses another class as a field or member (composition), like a Car has-a Engine. An is-a relationship means a class is a more specific type of another class (inheritance), like a Student is-a Person.
2. The derived class object will have both methods: it will inherit go() from the base class and it will also have its own stop() method. So a derived object can call both go() and stop().
3. Implementing an abstract method means giving a first body to a method that was declared abstract (no body) in an abstract class or interface. Overriding a method means replacing an already implemented inherited method with a new version that has the same signature but different code.
4. An abstract class can have fields, constructors, both concrete and abstract methods, and a class can extend only one abstract (or concrete) class. An interface mainly defines a set of methods that must be provided by implementing classes, and a class can implement many interfaces but interfaces usually do not hold instance state.
5. The Comparable interface contains one required method: int compareTo(T other).
6. **For the given code:**

```
interface Wo {
   public int doThat();
}

public class Bo {
   private int x;
   public Bo(int z) { x = z; }
   public int doThis() { return 2; }
   public int doNow() { return 15; }
}

public class Roo extends Bo implements Wo {
   public Roo() { super(1); }
   public int doThis() { return 10; }
   private int doThat() { return 20; }
}
```

a) In Wo, doThat() is an abstract interface method (implicitly public and abstract).
b) Wo is an interface that defines a contract: any class that implements it must provide a doThat() method.
c) doThat() is implemented in Roo because Roo states it implements Wo, so it must supply a concrete body for the interface method in order to be a non-abstract class.
d) A Roo object has:

- doThis() (the overridden version from Roo);
- doNow() (inherited from Bo);
- an implementation of doThat() to satisfy Wo.
  From outside the class, only the public methods are callable; conceptually, a Roo supports doThis, doNow, and doThat.

7. e) The doThis() method in Roo overrides the doThis() in Bo, so when doThis() is called on a Roo object, Roo's version runs instead of Bo's version.

f) The statement super(1) in Roo calls the constructor of the superclass Bo with the argument 1, initializing the Bo part of the Roo object.
g) Yes. From inside Roo, the superclass version can be called with super.doThis(). From outside, calling doThis() on a Roo object uses the overridden version in Roo, but code inside Roo can still reach Bo's implementation using super.doThis().
h) Yes. Any method in Roo can call Bo's doThis() by writing super.doThis() inside its body.

8. True/False:
a) True. Inheritance lets a class define a more specialized type of an existing class.
b) False. Derived classes show an is-a relationship, not a has-a relationship.
c) False. A class can be part of a chain of inheritance (multiple levels), although it can only extend one direct superclass.
d) False. The keyword used in the class declaration is extends, not the word "inheritance".
e) True. When writing a subclass, you can override inherited methods that are not final.
f) False. Private members of a base class are not directly accessible in derived classes.
g) True. Inherited methods are called on objects with normal dot notation, as if they were defined in the subclass.
h) True. Polymorphism means an object can be treated as instances of different related types (for example, a subclass object used through a superclass or interface reference).
i) False. Abstract classes cannot be instantiated directly.
j) True in the usual sense: any concrete subclass must implement all inherited abstract methods (otherwise that subclass must also be abstract).
k) False. An abstract method has only a declaration (signature) and no method body.
l) True. Inheritance and abstraction are used together to build class hierarchies.
m) True. One interface can inherit from another interface (using extends).
n) True. When a class implements an interface, it adds the behaviors defined by that interface's methods.
o) False. Interface methods are public and abstract by default, not private.
p) False. Comparable has one abstract method, compareTo, not three.