Critical Response Questions – Chapter 10

---

1. Explain the difference between a has-a and is-a relationship among classes.
A *has-a* relationship represents composition; one class contains another class as a field (e.g., a Car **has a** Engine).
An *is-a* relationship represents inheritance; one class is a specialized version of another (e.g., a Dog **is an** Animal).

2. If a base class has a public method go() and a derived class has a public method stop(), which methods will be available to an object of the derived class?
Both go() and stop() will be available. Derived classes inherit all accessible members of the base class.

3. Compare and contrast implementing an abstract method to overriding a method.

- Implementing an abstract method means providing the first concrete definition of a method that has no body in the abstract class.

- Overriding a method means redefining a method that already has an implementation in the base class.
  In both cases, the derived class supplies a method with identical signature.

4. Compare and contrast an abstract class to an interface.

- An abstract class may contain both abstract and non-abstract methods, fields, and constructors.

- An interface contains abstract methods (before Java 8) and may include default/static methods (after Java 8).

- A class can extend only one abstract class but can implement multiple interfaces.

- Interfaces provide a contract; abstract classes provide partial implementation.

---

6. Based on the given classes:

```
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) What type of method is doThat() in Wo?

It is an abstract method. All interface methods without a body are abstract by default.

b) What is Wo?

Wo is an interface, which defines a contract that any implementing class must fulfill.

c) Why is doThat() implemented in Roo?

Because Roo implements interface Wo, and therefore must provide a concrete implementation of all abstract methods in that interface.

d) List the methods available to a Roo object.

- doThis() (Roo version)
- doNow() (inherited from Bo)
- doThat() (implementation from Roo, though declared private — it still fulfills the interface requirement but is not publicly accessible)
- Any public or protected members inherited from Bo.

e) How does the implementation of doThis() in Roo affect the implementation of doThis() in Bo?

Roo *overrides* doThis(). The version in Roo replaces Bo's version when called on a Roo object.

f) What action does the statement super(1) in Roo perform?

It calls the constructor of the superclass Bo, passing the value 1 as the argument.

g) Can the doThis() method in Bo be called from a Roo object? If so, how?

Yes. It can be called using:

```
super.doThis();
```

From within Roo's methods.

From outside the class, you cannot directly call the superclass version because the Roo version overrides it.

h) Can a method in Roo call the doThis() method in Bo? If so, how?

Yes. A method inside Roo can call the superclass version using:

```
super.doThis();
```