Chapter 7 – Critical Thinking Answers

1. Which members of the Circle class are encapsulated?
The members declared as `private` are encapsulated. These are usually the instance variables such as `radius`, which can only be accessed through public methods.

2. What name must the constructor of a class have?
The constructor must have the same name as the class and no return type.

3. Explain the difference between the private and public access modifiers.

- `public`: Can be accessed from anywhere in the program.
- `private`: Can only be accessed within the same class.

4. Consider the code below:

```
Circle dot = new Circle(2);
dot.radius = 5;
```

This statement is invalid if `radius` is declared as `private` in the `Circle` class. You must use a public method like `setRadius()` to change it.

5. Roo class:

```
public class Roo {
    private int x;
    public Roo() {
        x = 1;
    }
    public void setX(int z) {
        x = z;
    }
    public int getX() {
        return(x);
```

```
    }
    public int calculate() {
        x = x * factor();
        return(x);
    }
    private int factor() {
        return(0.12);
    }
}
```

a) Name of the class: Roo
b) Name of the data member: x
c) Accessor method: getX()
d) Modifier method: setX(int z)
e) Helper method: factor()
f) Name of the constructor: Roo()
g) Number of method members: 4 (setX, getX, calculate, factor)

6. What is the difference between a class and an object?
A class is a blueprint or template that defines data and behavior.
An object is an instance of a class created in memory.

9. Use the class data member definitions:

```
public class Moo {
    private double y;
    private static int x;
    private static final z;
}
```

- y is an instance variable.
- x is a static class variable shared by all objects.
- z is a static constant, but this line is incomplete because it has no data type or value. It should be something like:

```
    private static final double z = 10.5;
```

11. Compare and contrast overriding and overloading methods.

Feature	Overloading	Overriding
Definition	Same method name, different parameter lists, same class.	Subclass redefines a method from its superclass.
Purpose	Perform similar tasks with different inputs.	Change or extend inherited behavior.
Location	Within the same class.	Between superclass and subclass.
Return Type	Can differ.	Must be the same or compatible.
Access Level	Any.	Cannot be more restrictive.