Critical Thinking

1) The **private data fields** (like `radius` in a `Circle` class). Encapsulation is just keeping the inside of the object protected.
2) The constructor must have the **exact same name as the class**
3) Private vs Public

- `private`: Hidden. Only code *inside* that class can see or use it. (Like your secret diary.)
- `public`: Open. Any other class can use it. (Like a front door.)

4) The statement `dot.radius = 5;` is **Invalid (Compiler Error)**.

- **Reason:** You can't directly touch a **private** member (`radius`) from outside the class. You must use a public method, like `dot.setRadius(5);`.

5) Name of the class? **Roo** b) Name of the data member? **x** c) Accessor method (gets the data)? **`getX()`** d) Modifier method (sets/changes the data)? **`setX()`** e) Helper method (private utility method)? **`factor()`** f) Name of the constructor? **`Roo()`** g) How many method members are there?
6) **Class vs. Object?**
Class: The blueprint or recipe. (e.g., The idea of a cookie.)

Object: A real instance created from the blueprint. (e.g., A specific, baked cookie you can hold.)

7)

Role in Simulation	Appropriate OOP Name(s)
a) The class	Band
b) The objects	**TheTwoToos**, **EggRolls**, **Goop** (These are the specific instances of the `Band` class)
c) a data member	**SetList**, **BandName**, **MembersCount** (Stores information about the band)
d) the method members	**TuneUp()**, **PlayMusic()**, **TakeABow()** (Actions the band can perform)

9) **Class Data Member Definitions (`public class Moo`)**

- `y` is an **instance variable** (each object gets its own copy).
- `x` is a **static class variable** (shared by all objects).
- `z` is a **static class constant** (shared and unchangeable)

11) **Compare and contrast overriding methods to overloading methods.**

Overloading allows multiple methods with the same name but different parameters in the same class, while **overriding** changes the implementation of an inherited method in a child class.