1. The private data members are the members that are encapsulated by the Circle class: instance variable *radius* and *PI*, which is a class constant. Encapsulation prevents them from being accessed directly by client code, and only allows them to be changed or accessed by public methods, such as *setRadius*, *getRadius*, and *area*.

2. The constructor must have the same name as the class itself.

3. The *public* access modifier exposes a member (variable or method) and can be used or invoked by other classes, including client code, so it can be utilized or called upon by other classes. The *private* access modifier is limited to the class only, and does not allow direct access by client code, nor does it enforce encapsulation by concealing internal data or auxiliary methods.

4. The final argument is not true. It tries to directly read and write the *radius* variable, which is a *private* variable in the circle class. The access to the private members is not possible directly through the client code it would create a compiler error. Rather, the radius should be changed using the public modifier method *setRadius()* to do so.

5a. Roo
5b. x
5c. getX()
5d. setX(int z)
5e. factor()
5f. Roo
5g. 4 methods; setX(), getX(), calculate(), factor()

6. A class is a template or blue blueprint or data type that determines the structure (state variables) and the behaviour (methods) of objects. An object is a sample of a template (a particular implementation of that template with its own state (e.g. unique values of variables) and sharing the methods of the class.

9a. z
9b. y and x
9c. y
9d. x and z

11.
Similarities:

Both entail approaches that share the same name and are applied to give flexibility in the approach. They are able to improve the reusability of code and permit specialty or diverse behaviour.

Differences:

Overriding is used when a subclass overrides (redefines) a method that is inherited by its superclass without changing its method signature (name, parameters, return type) to replace that of the superclass with that of a subclass object. It is applied to polymorphism and the customization of the runtime behaviour. In contrast, overloading consists of several methods sharing the same name (or across inheritance) that have different parameter lists (number or types of parameters). It is decided during compile time, and it is usually applied to constructors or methods that accept various inputs without alteration of the name of the method.