1. **Which members of the Circle class are encapsulated?**
   In the `Circle` class, the instance variables, such as `radius`, are typically encapsulated by being marked as `private`. This restricts direct access from outside the class, and access is provided through public methods like `getRadius()` and `setRadius()`.
2. **What name must the constructor of a class have?**
   The constructor must have the **same name as the class** and no return type. For example, in a `Circle` class, the constructor must be named `Circle`.
3. **Explain the difference between the private and public access modifiers.**
   The `private` modifier restricts access to members, allowing them to be accessed only within the same class, while the **`public`** modifier allows members to be accessed from any other class. `Private` ensures data security, while `public` provides external access.
4. **Consider the following code.Is the last statement valid or invalid?Explain**

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

The last statement, `dot.radius = 5;`, is **invalid** if the `radius` member of the `Circle` class is declared as `private`. Since `private` members cannot be accessed directly from outside the class, this statement would result in a compilation error. To make this valid, you would need to use a **setter method** if one is provided, for example, `dot.setRadius(5);` assuming the `setRadius()` method is public and allows modification of the `radius` variable.

## 5. Use the following class to answer the questions below

```
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) What is the name of the class?**
The name of the class is **Roo**.

**b) What is the name of the data member?**
The name of the data member is **x**.

**c) List the accessor method.**
The accessor method is **getX()**, which returns the value of the x variable.

**d) List the modifier method.**
The modifier methods are **setX(int z)** and **calculate()**, as they modify the state of the x variable.

**e) List the helper method.**
The helper method is **factor()**, which is used within the `calculate()` method to return a factor value.

**f) What is the name of the constructor?**
The name of the constructor is **Roo()**, as it matches the name of the class.

**g) How many method members are there?**
There are **4 method members**: `setX(int z)`, `getX()`, `calculate()`, and `factor()`.

## 6. What is the difference between a class and an object?

- **a) The class**: A **class** is a blueprint or template from which objects are created. It defines the properties (data members) and behaviors (methods) that its objects will have.
- **b) The objects**: An **object** is an instance of a class. It contains actual values for the properties defined in the class and can invoke methods defined in the class.
- **c) A data member**: A **data member** (or instance variable) is a variable that holds data associated with an object of a class. Each object of the class can have its own values for these data members.
- **d) The method members**: A **method member** is a function defined inside a class that specifies behaviors or actions the class can perform. Methods are used to operate on the class's data members or perform actions.

## 9. Use the following class data member definitions to answer the questions below:

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

**private double y;**
This is an **instance variable** of type `double`. It holds a value for each object of the `Moo` class.
**private static int x;**
This is a **static variable** of type `int`. It is shared by all instances of the `Moo` class, meaning all objects of `Moo` refer to the same `x` variable.
**private static final int z;**
This is a **static final variable**. It is shared across all instances of the class, and since it's marked as `final`, its value cannot be changed once initialized.