1. What is a class constructor and why is it needed?

A ***class constructor*** is a special method used to initialize objects when they are created. Each class can have one or more constructors, which are responsible for setting the initial state of an object (such as initializing member variables).

* Initialize object state: Constructors allow us to set initial values for an object at creation to ensure it's in a valid state.
* Simplify code: Constructors can accept parameters, allowing necessary values to be provided at the time of object creation instead of requiring additional setter method calls.
* Enforce required parameters: With parameterized constructors, you can ensure that necessary information is provided when creating an object.

**Example:**



In this Car class, we have defined three constructors:

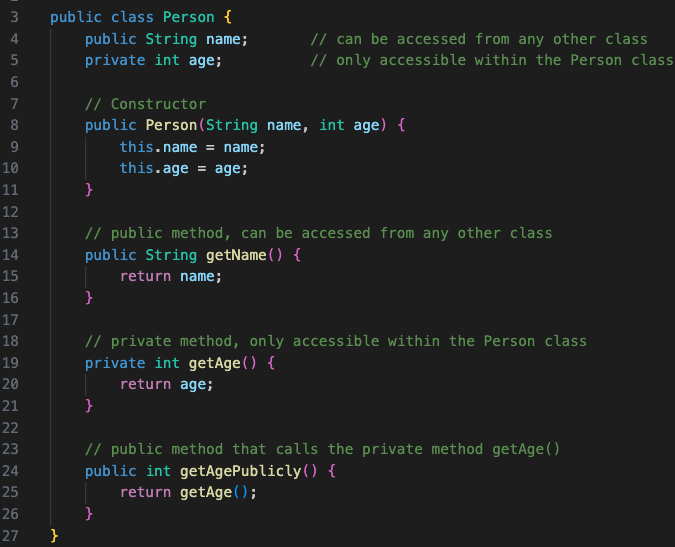
* **Constructor 1**: A no-parameter constructor that sets default values for all attributes.
* **Constructor 2**: A constructor with model and year parameters, allowing the user to specify the model and year, while the color defaults to "White."
* **Constructor 3**: A constructor with model, year, and color parameters, enabling the user to specify all attributes.

1. What is the meaning of the following access modifiers: “public,” “private,” “protected,” and “default”?

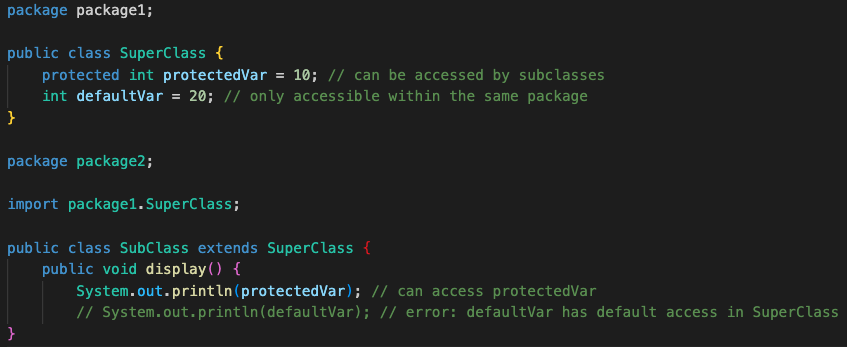
**Access modifiers** control the visibility and accessibility of class members:

* **public**: The member is accessible from anywhere. Other classes can access it regardless of their package.
* **private**: The member is only accessible within the class where it is defined, making it inaccessible to other classes. This is useful for encapsulation, preventing direct modification of a class's internal state.
* **protected**: The member is accessible within the same package and in subclasses, even if those subclasses are in different packages. This is commonly used for members that should be available to subclasses but not fully public.
* **default** (package-private): When no access modifier is specified, the member is accessible only within the same package. This level restricts access to classes within the same package and is also known as package-private.

**Example:**

****

****

****

1. What is the meaning of the following non-access modifiers: “final” and “abstract”?

***Non-access modifiers*** don’t control visibility but specify the behavior of a member or class.

**Final:**

* For variables: When a variable is declared as final, its value **cannot be changed once initialized** (it becomes a constant).

Example:

*public static final double PI = 3.14159;*

* For methods: When a method is declared as final, it **cannot be overridden** by subclasses.

Example:

*public final void breathe() { System.out.println("Breathing..."); }*

* For classes: When a class is declared as final, it **cannot be subclassed**.

Example:

*public final class Constants {*

*public static final double GRAVITY = 9.8;*

*// This class cannot be subclassed*

*}*

**Abstract:**

* For methods: Declaring a method as abstract means it **has no implementation and must be implemented by subclasses**. Abstract methods **can only exist within abstract classes**.
* For classes: Declaring a class as abstract means **it cannot be instantiated directly**. Abstract classes are **meant to be inherited by subclasses** and typically contain abstract methods.

Example:

*public abstract class Animal {*

*public abstract void makeSound();*

*}*

*public class Dog extends Animal {*

*@Override*

*public void makeSound() {*

*System.out.println("Bark");*

*}*

*}*

1. What is a Java package?

A **Java** **package** is a logical grouping of related classes and interfaces. Packages help with:

* Organizing code: Similar to folders, packages allow you to group related classes, making them easier to manage.
* Avoiding naming conflicts: Classes in different packages can have the same name since they are referenced with their fully qualified names (packageName.className).
* Controlling access: Packages offer flexibility in setting class and method accessibility (e.g., package-private access).

Example:

*package main;*

*// import animals package 中的類別*

*import animals.Animal;*

*import animals.Dog;*

*public class Main {*

*public static void main(String[] args) {*

*Animal dog = new Dog("Buddy");*

*System.out.println("Animal's name: " + dog.getName()); // Output: Animal's name: Buddy*

*dog.makeSound(); // Output: Bark*

*}*

*}*