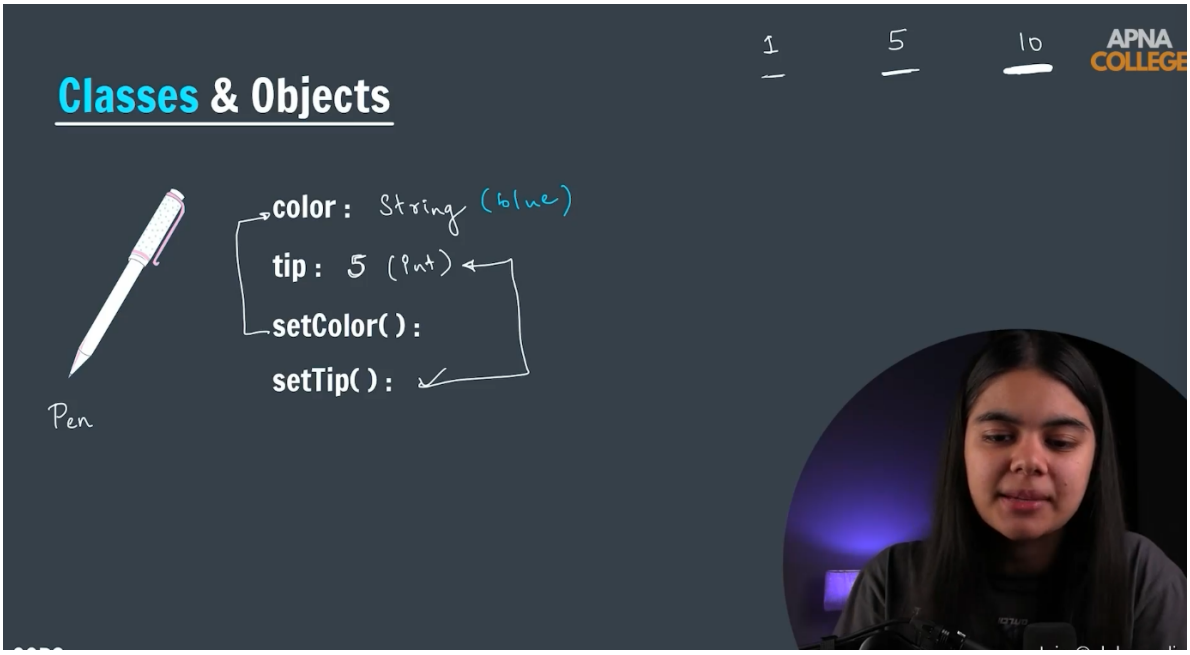
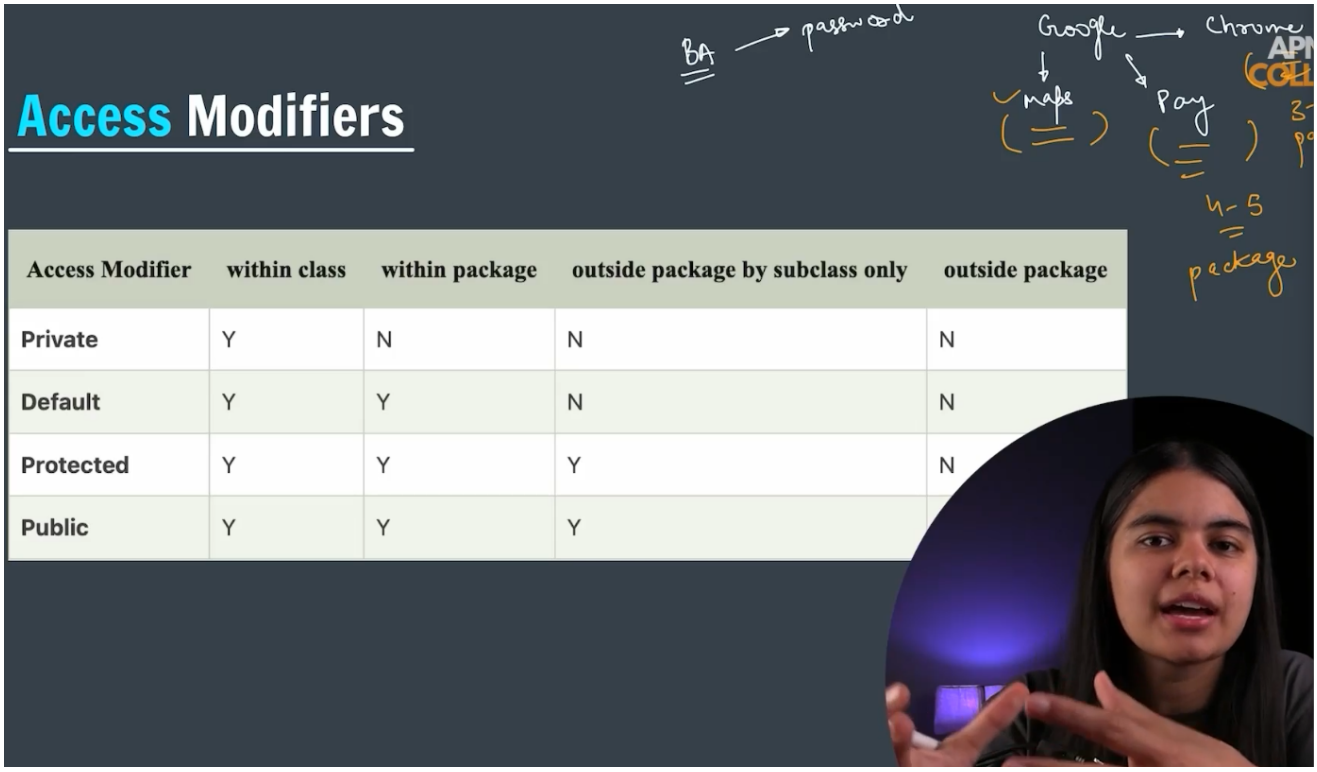
OOPS

▼ Objects :

- Objects are entities in the real world.
- Objects can have its own properties and functions.
- Class is a blueprint of an object defining the functions and properties of the objects.
  - Attributes : e.g. color of a pen
  - Functions : Change the color of the object



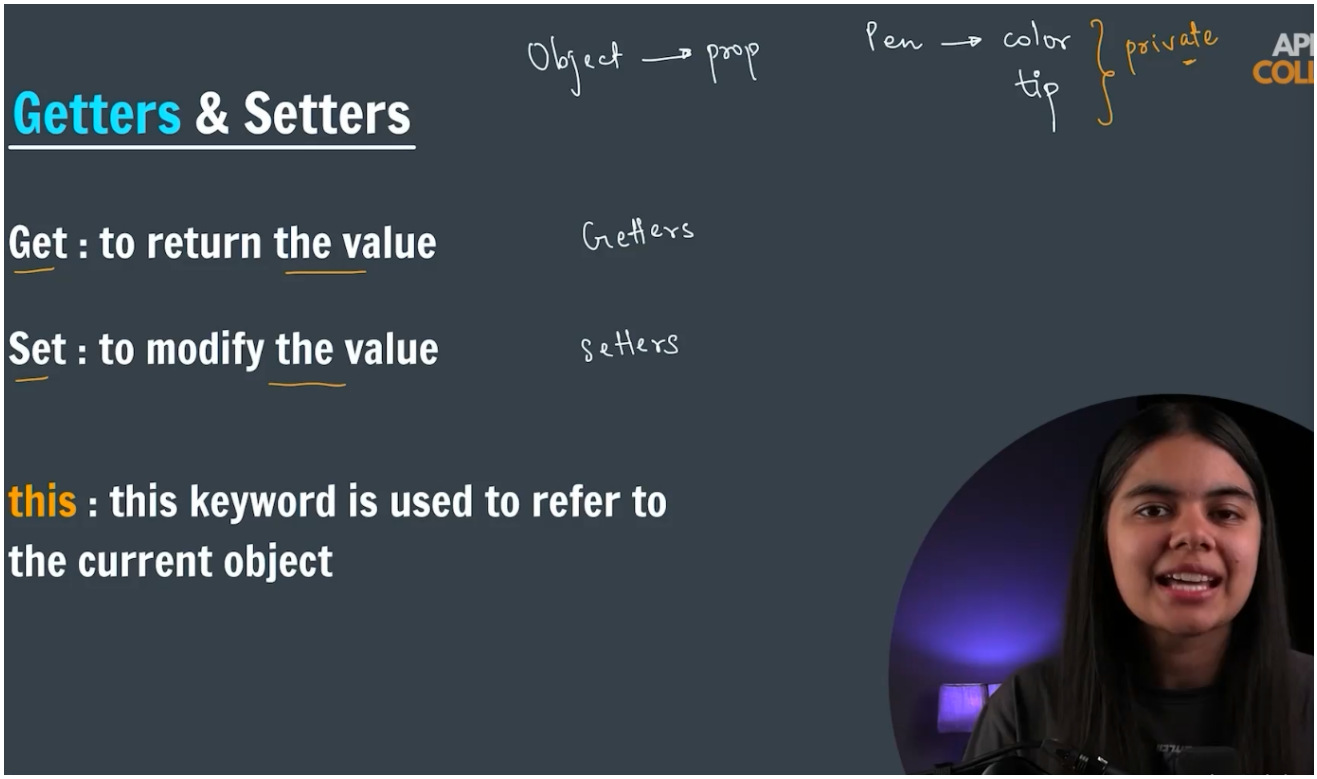
▼ Access Modifiers :

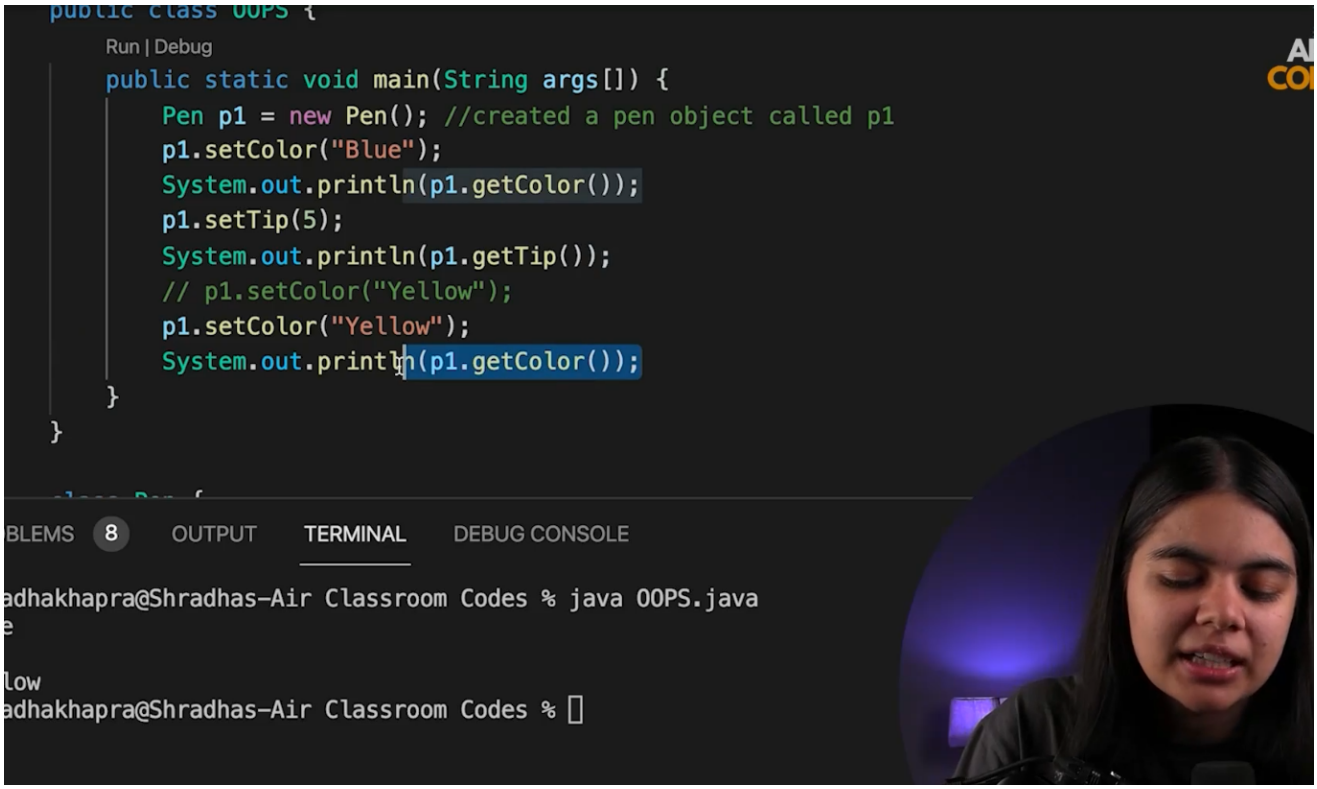


Access Modifier	within class	within package	outside package by subclass only	outside package
Private	Y	N	N	N
Default	Y	Y	N	N
Protected	Y	Y	Y	N
Public	Y	Y	Y	

▼ Getters and Setters :

- Get : to return the value
- Set : to modify the value





this.color , here this is used to refer the current object. Mainly used when there are multiple properties with the same name.

▼ Concepts :

▼ Encapsulation :

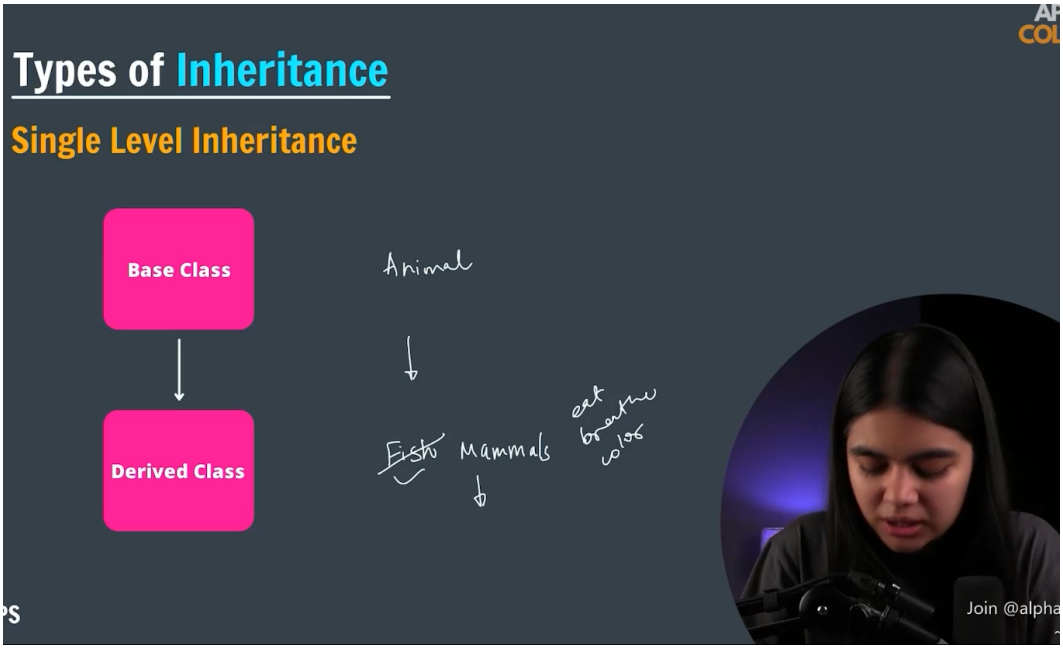
- Encapsulation is defined as the wrapping up of data (properties) & methods (function) under a single unit. It also implements data (sensitive data or Useless data) hiding .

▼ Inheritance :

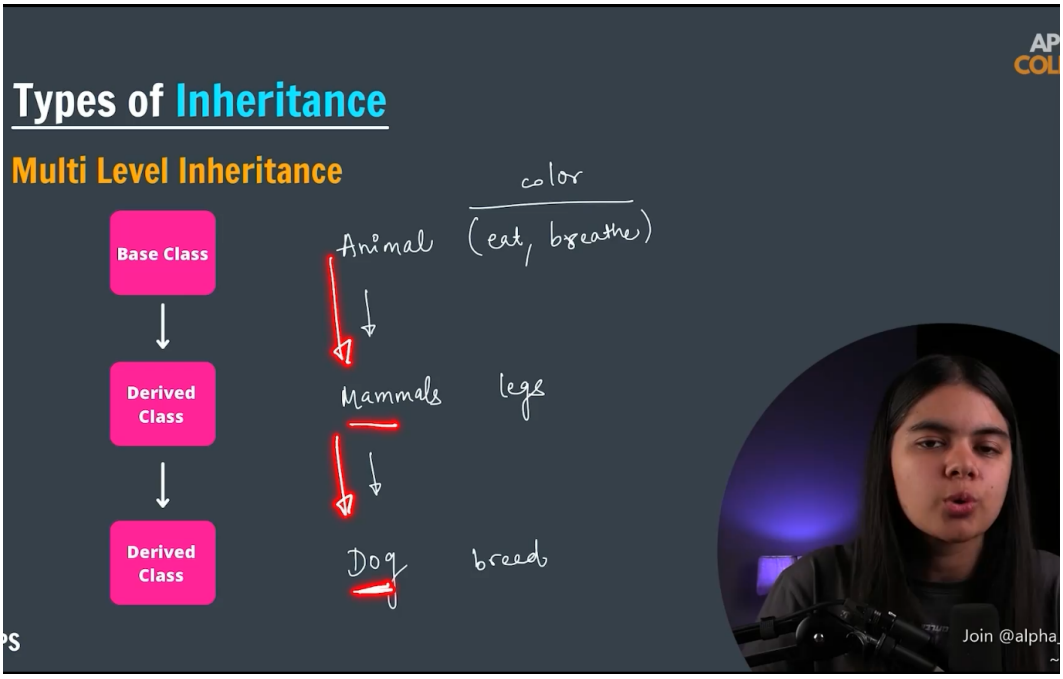
- Inheritance is when properties & methods of base class (parent class) are passed on to a derived class (child class).
- ‘extends’ keyword is used to establish inheritance connection

▼ Types of Inheritance

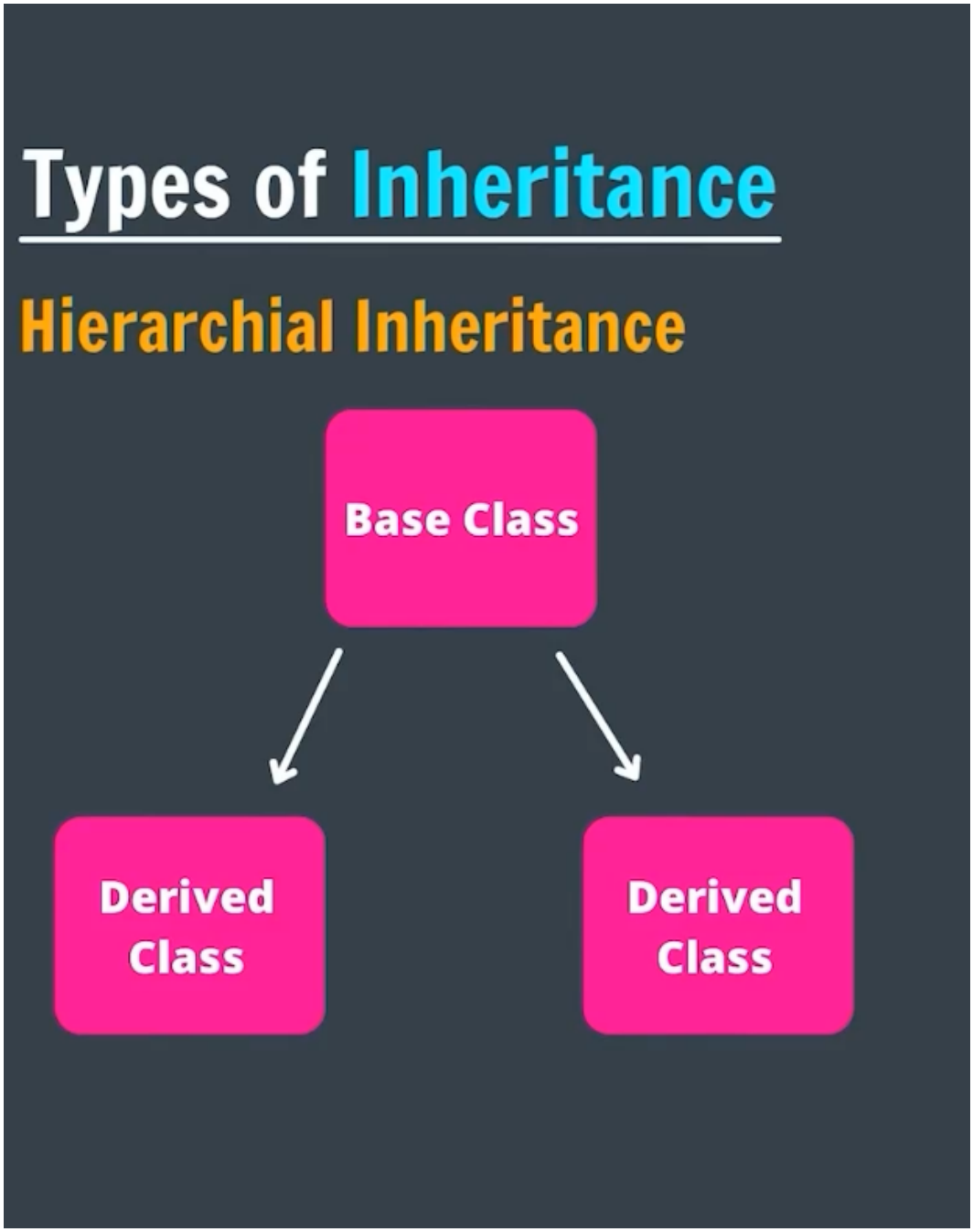
- Single Level Inheritance



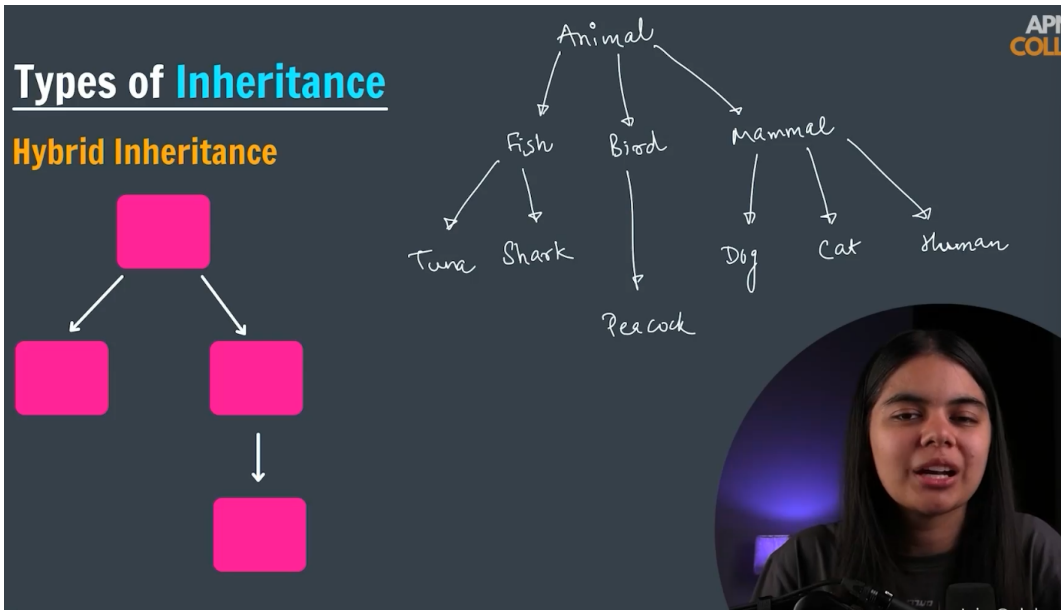
- Multi-level Inheritance : More than one inheritance class can inheritance form base class



- Hierarchical Inheritance : Multiple Inherited classes form one single base class



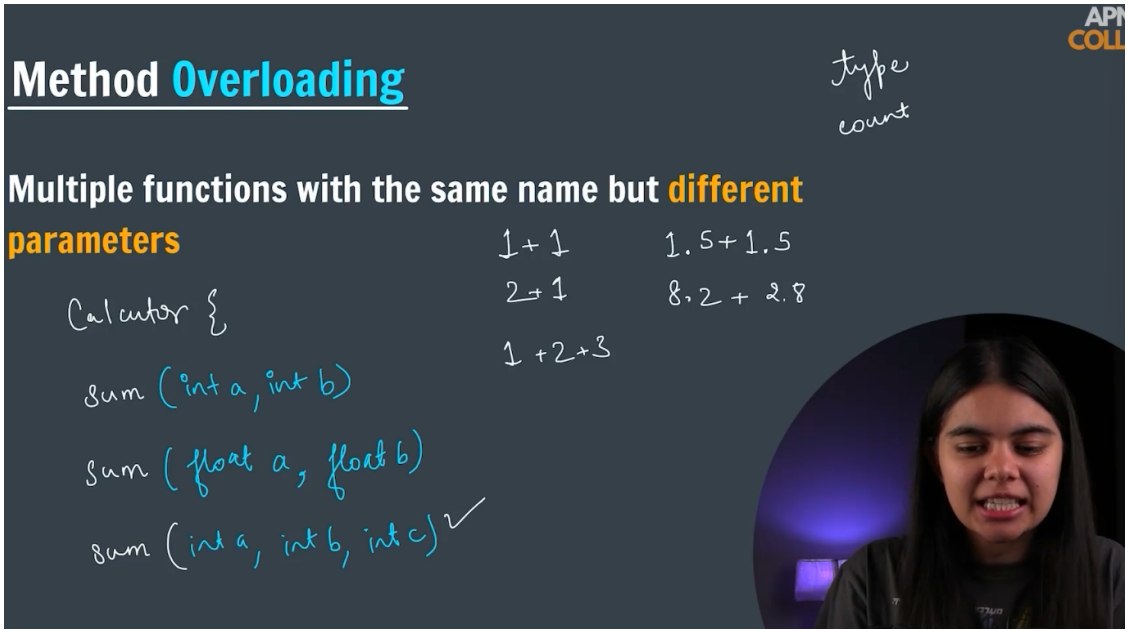
- Hybrid Inheritance :



- Multiple Inheritance : Is not available in JAVA (can be done using interfaces) , only in C++

▼ Polymorphism :

- When we try to do the same thing using different methods.
- Compile Time Polymorphism / Static :
  - Method / Function Overloading : Multiple functions with the same name but different parameters



- Run Time Polymorphism / Dynamic :

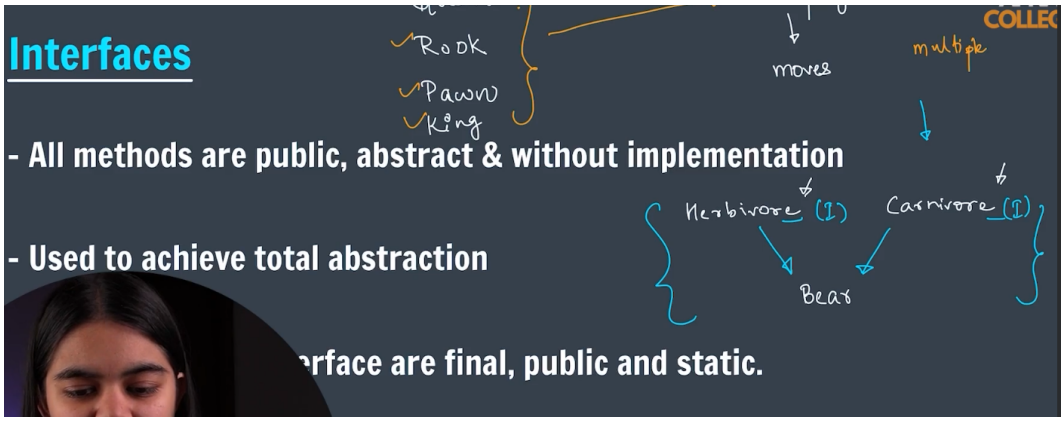
- Method / Function Overriding : Parent and child classes both contain the same function with a different definition.
- Always the child class function would be called

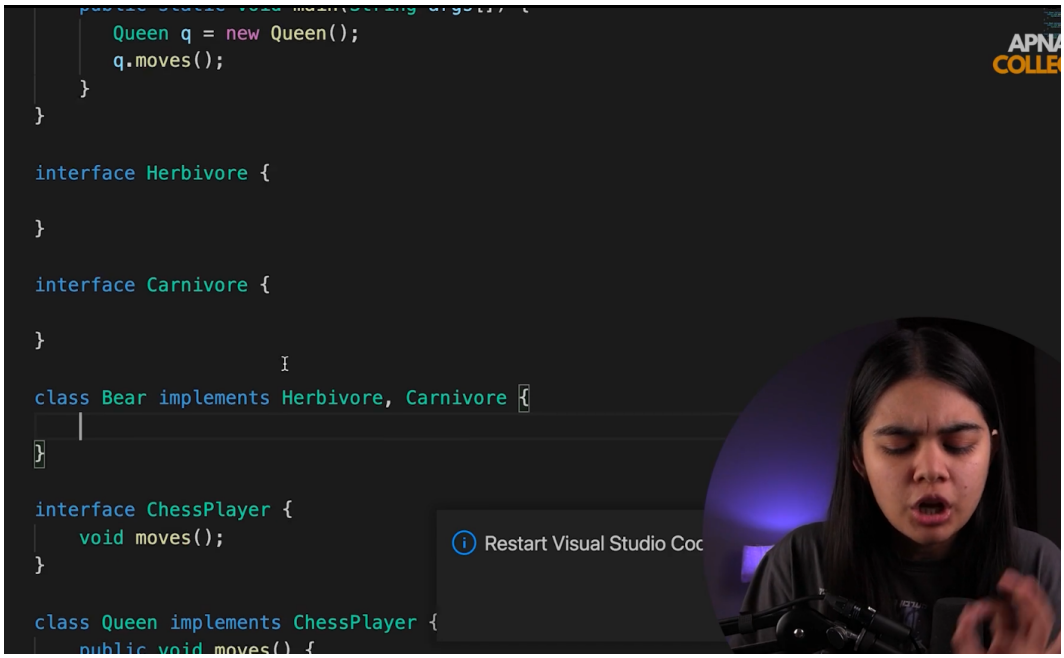
▼ Abstraction :

- Hiding all the unnecessary details and showing only the important parts to the user.
- How Abstraction is implemented :
  - Abstract Classes
    - e.g. The implementation need not to be defined. The implementation can be individually decided by each sub- class on there on .
    - It provides a kind of idea for the subclasses

```
abstract void walk();
```

  - Interfaces : Interface is a blueprint of a class
    - Is used to implement multiple inheritance

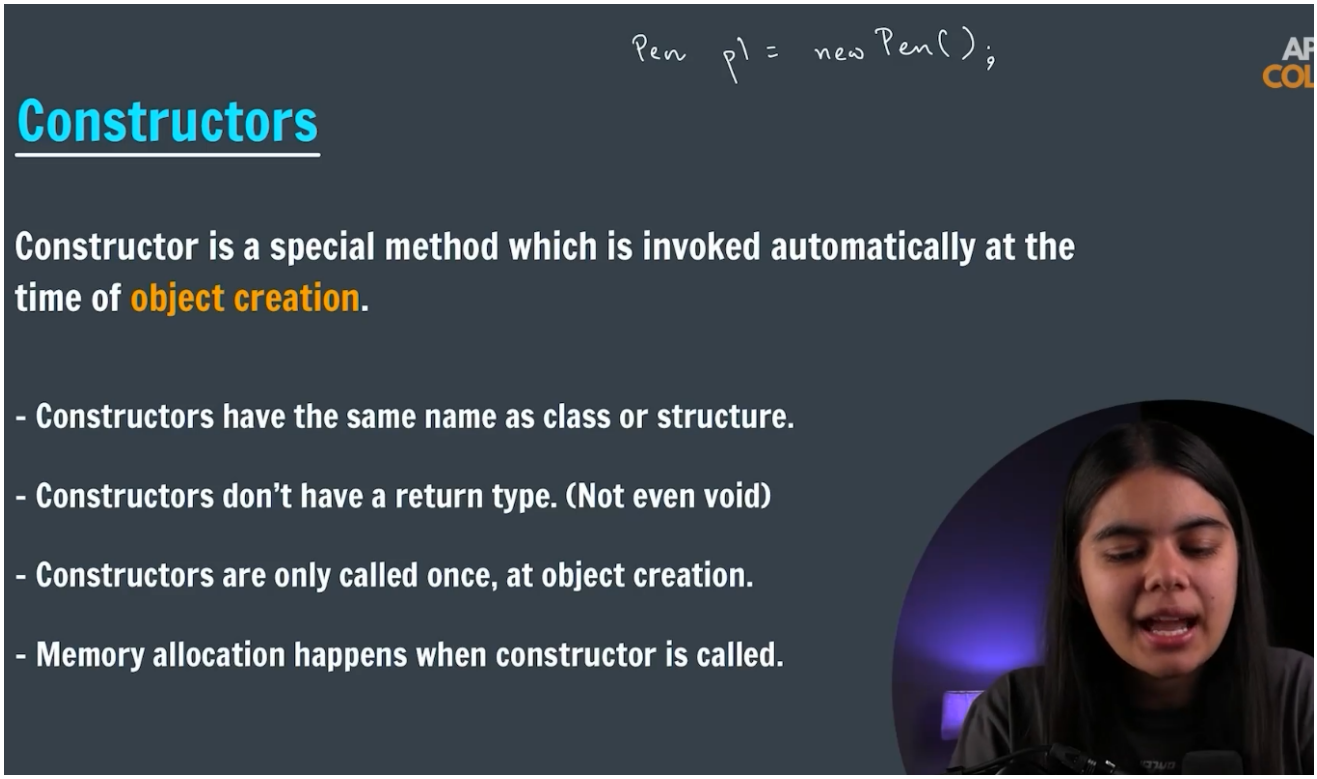




- 100% abstraction is implemented using interfaces.
- It is defined using interface keyword
- Classes are extended whereas Interface is implemented
-

▼ Constructor :

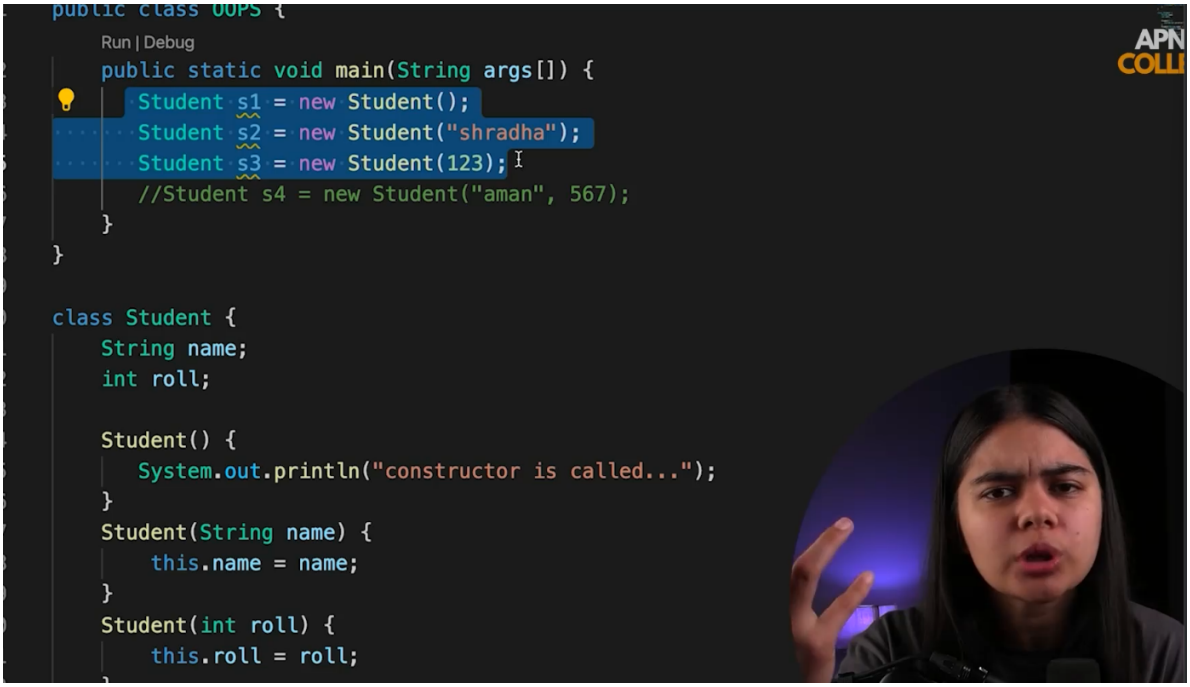
Constructor is a special method which is invoked automatically at the time of object creation.



- If a constructor is not defined , JAVA assigns an default constructor to the object
- To set up some initialization values such as name or age we need to define the constructor ourself

▼ Types of Constructors :

  - Non-parameterized
  - Parameterized
  - Copy Constructor



- Constructor Overloading : When we have created multiple constructors , so that multiple types of initialization statements could work.
  - It is a type of polymorphism

▼ Copy Constructor :

- Copy the properties of one object to another properties
- It is found as default in C++ but not in JAVA.

▼ Destructor :

Garbage Collector : Randomly checks and removes any unnecessary items from the memory

▼ Packages in JAVA :

- Package is a group of similar types of classes, interfaces and sub-packages.

▼ Keywords :

▼ Static Keyword

- Static keyword in Java is used to share the same variable or method of a given class.
- The static variable would be same for any and every other objects
- Static variable is only assigned value once.

- Same value can be used without saving the string multiple types.
- It also saves memory as all other events of the static keyword are used to refer to that already existing memory point.

▼ Super Keyword

- Super keyword is used to refer immediate parent class object.
- It is used to call the constructor of the immediate parent class
- Java by default call the super keyword.
- We can also alter the values of the parent using the super keyword.

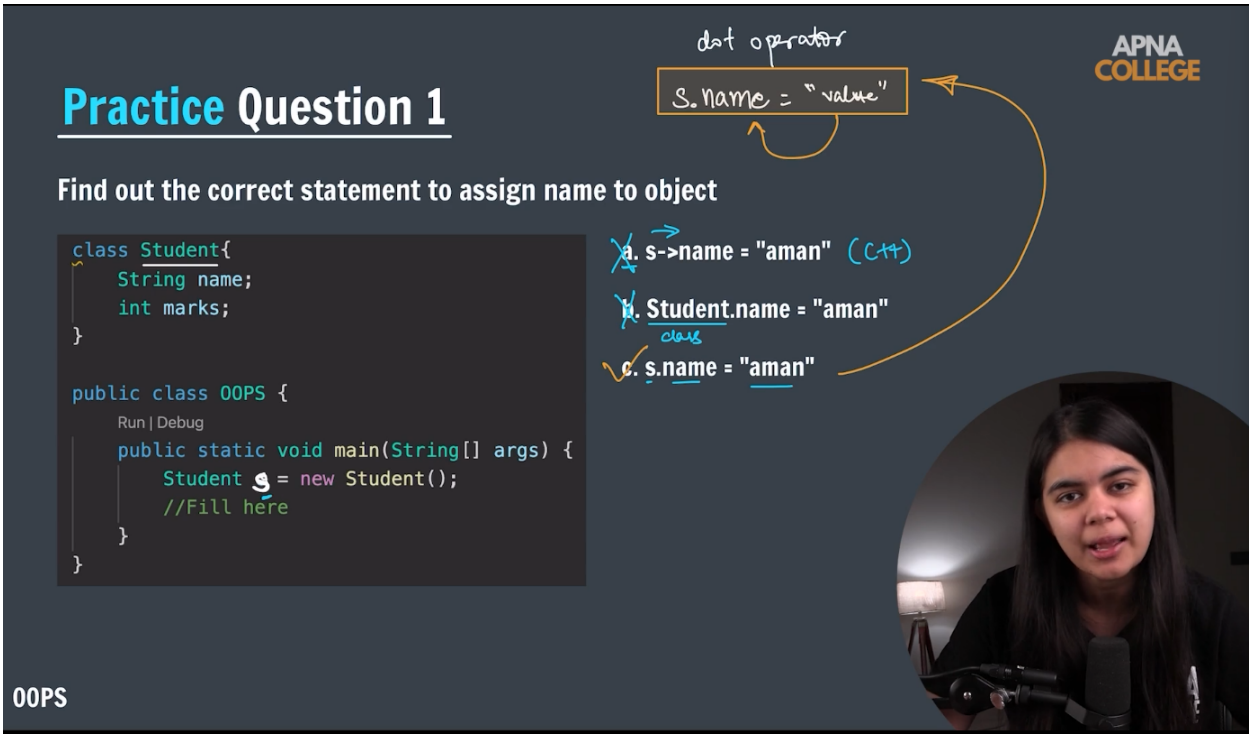
```
  class Horse extends Animal {
Horse() {
super.color = "brown";
System.out.println("horse constructor is called");
  }
}
```

  - to access parent's properties
  - to access parent's functions
  - to access parent's constructor

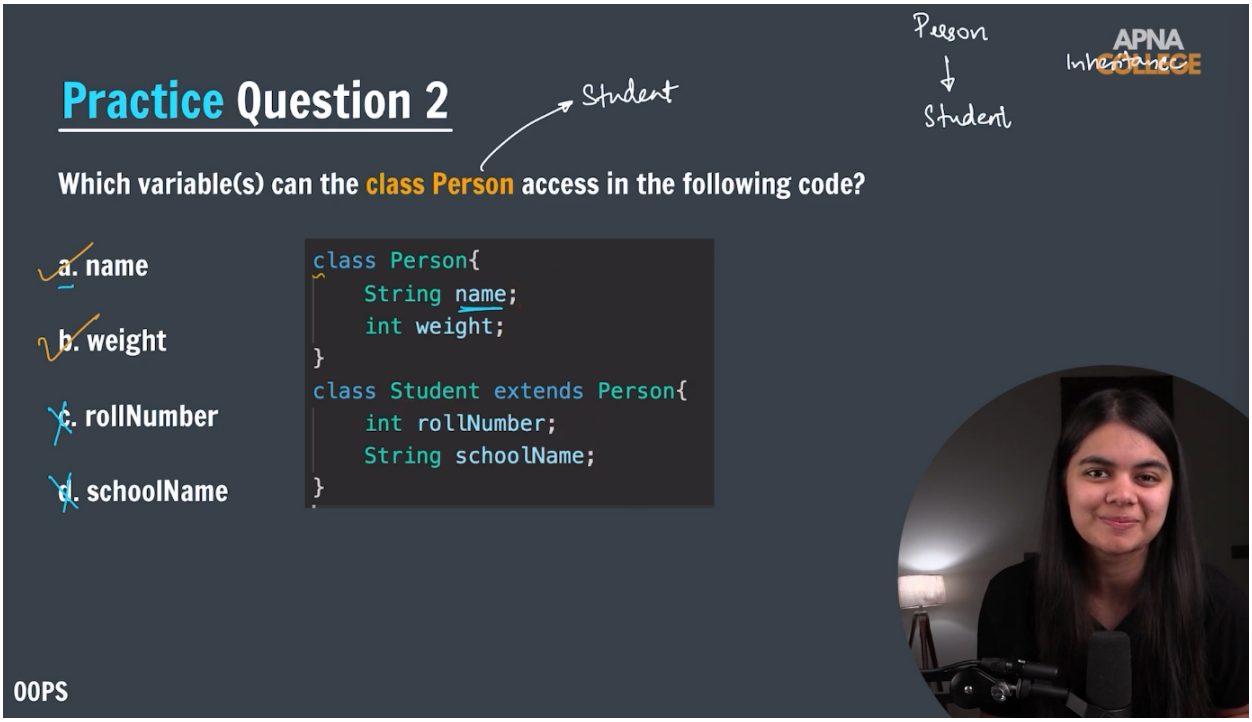
▼ Question

▼ Find out the correct statement to assign name to object

- s.name = value (It is used to assign name to the name , s here is the object)

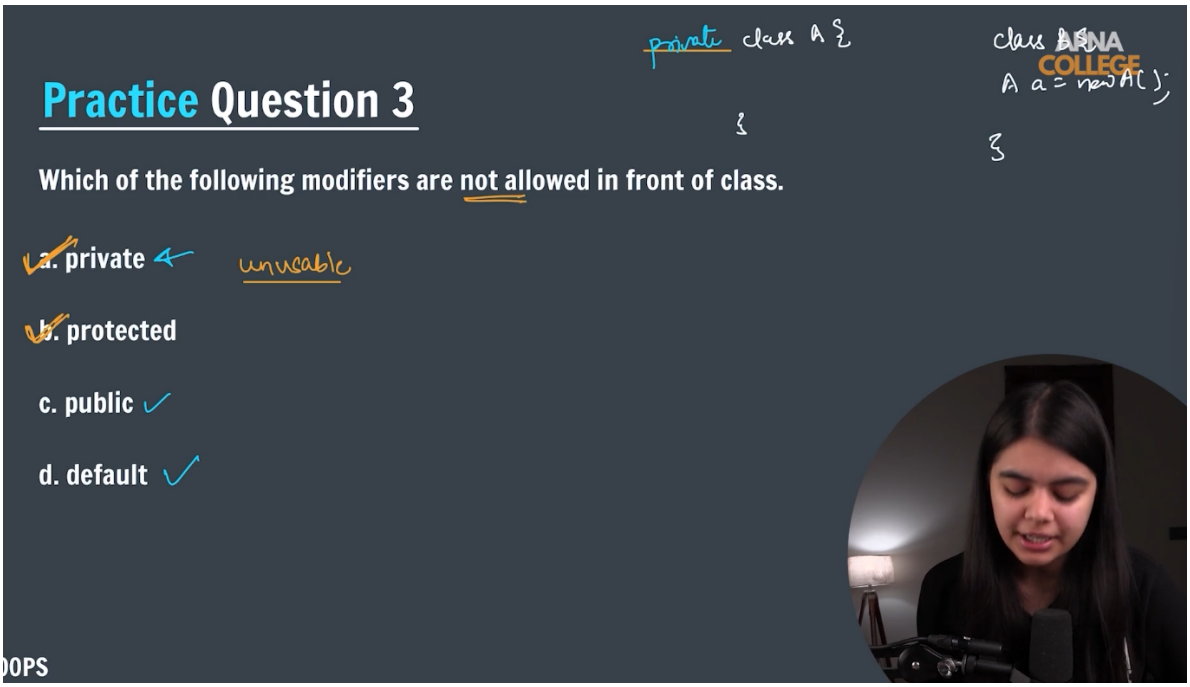


▼ Which variable(s) can the class Person access in the following code?

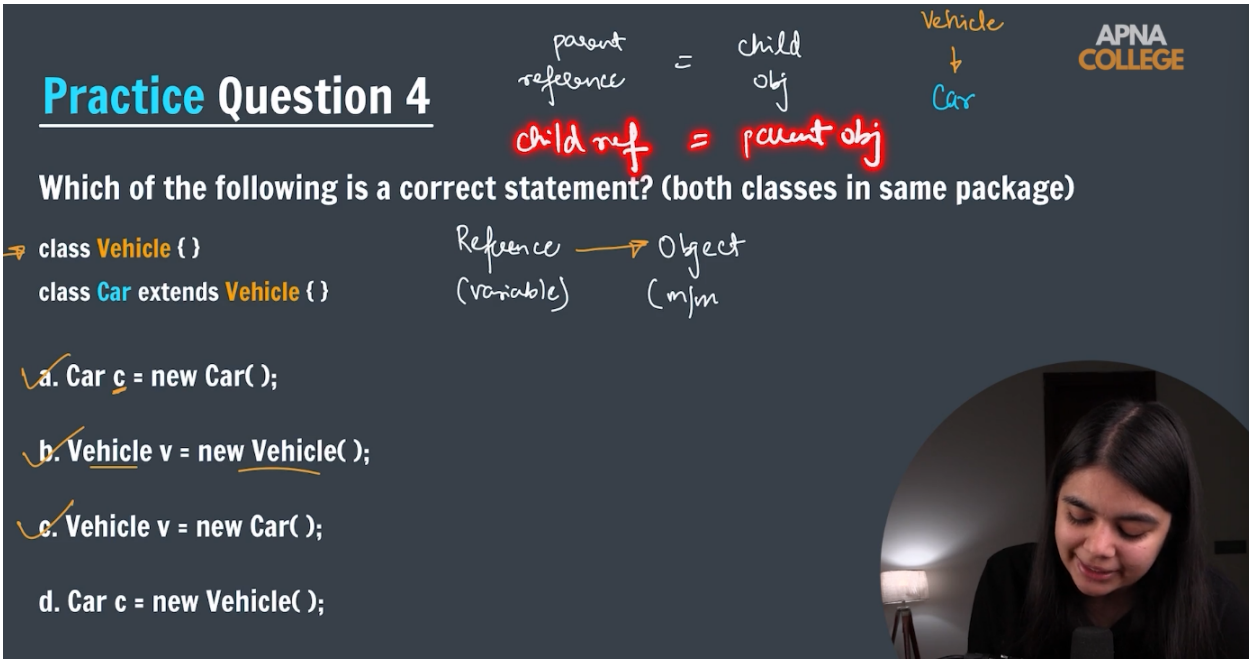


▼ Which of the following modifiers are not allowed in front of class.

- In JAVA no outer/base class is allowed to be neither Public nor Protected as both these properties render the class as unusable



▼ Which of the following is a correct statement?



▼ Function Overriding