1. **Scenario: Constructor Confusion**

You are building a class Book with a constructor that accepts a title and an author. However, when creating a Book object using:

Book b = new Book();

You get a **compile-time error**.

public class Book {

    String title;

    String author;

    public Book(String title, String author) {

        this.title = title;

        this.author = author;

    }

}

**Question:**  
What is the reason for the compile-time error?

A. The class must be declared as abstract  
B. The constructor parameters are incorrect  
C. There is no default constructor defined  
D. Book cannot be instantiated without parameters

**2. Scenario: Method Overloading or Overriding?**

class Vehicle {

    void start() {

        System.out.println("Vehicle starting");

    }

}

class Car extends Vehicle {

    void start(String mode) {

        System.out.println("Car starting in " + mode + " mode");

    }

}

**Question:**  
If you run this:

Car car = new Car();

car.start();

What will be printed?

A. Vehicle starting  
B. Car starting in null mode  
C. Compilation error  
D. Car starting

**3. Scenario: Tricky Reference Behavior**

class A {

    int num = 10;

}

public class Test {

    public static void main(String[] args) {

        A obj1 = new A();

        A obj2 = obj1;

        obj2.num = 20;

        System.out.println(obj1.num);

    }

}

**Question:**  
What will be printed and why?

A. 10 — obj2 is a separate object  
B. 20 — obj2 and obj1 refer to the same object  
C. Compile-time error  
D. 0 — default value of int

**4. Scenario: Object Nullification**

public class Test {

    public static void main(String[] args) {

        String s = "abc";

        s = null;

        System.out.println(s.length());

    }

}

**Question:**  
What will be the result?

A. 3  
B. 0  
C. NullPointerException  
D. Compile-time error

**5. Scenario: Inheritance + Method Overriding + Constructor Order**

class Animal {

    Animal() {

        speak();

    }

    void speak() {

        System.out.println("Animal speaks");

    }

}

class Dog extends Animal {

    String sound = "Bark";

    @Override

    void speak() {

        System.out.println(sound.toUpperCase());

    }

}

public class Test {

    public static void main(String[] args) {

        new Dog();

    }

}

**Question:**  
What is the output? Why is it *not* what you'd expect?

**Answer**

1. There is no default constructor defined.
2. Compilation Error
3. 20 — obj2 and obj1 refer to the same object.
4. C. NullPointerException
5. The output of the program is **Null**. Because Animal is a parent, and Dog is a child. The parent always **asks the child to speak first** (via speak()) before the child learns how to speak (sound = "Bark"). So when the child tries to speak, it doesn't know any words yet, leading to **"null"** being printed.