Profe tenía una duda con este ejercicio .¿ Por qué al asignarle en la linea 10 a test un objeto del tipo Main al llamar al método myIB llama al método de  MyBaseClass , no al de Main?

***interface IntBase {***

***void myIB();***

***}***

***interface IntDerived extends IntBase {***

***int myIB(int n);***

***}***

***class MyBaseClass implements IntBase {***

***public void myIB() {***

***System.out.println(3);***

***}***

***}***

***public class Main extends MyBaseClass implements IntDerived {***

***public int myIB(int n) {***

***return 2 \* n;***

***}***

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

***Main der = new Main();***

***MyBaseClass test = der;***

***System.out.println(test.myIB(3));***

***}***

***}***

La pregunta de este ejercicio es: ¿Qué pasa al compilar el  programa y por qué?

RTA:

1. El programa no compila por la línea que está en rojo.
2. *test* de la clase *MyBaseClass* es un objeto cuyo método *myIB* no recibe ningún parámetro.
3. Si quisiéramos invocar el método *myIB* de la clase *Main* (que es la que recibe el argumento), no deberíamos anteponer la referencia a *test*. (En ese caso tendríamos otro problema: desde el método *main* (*public static*) no podemos invocar otro método que no esté calificado como estático).
4. **Para completar el análisis de todo el ejercicio sugiero armar el diagrama de clases, prestando especial atención a las herencias, y recordar que a una referencia cualquiera se le puede asignar un objeto de una subclase.**

**Ampliando un poco el punto (4)**

Por ejemplo, si tenemos



En java

|  |
| --- |
| ***class Vehiculo {***  ***}***  ***class Avion extends Vehiculo {***  ***}*** |

Luego, al crear objetos, por ejemplo en el programa principal.

|  |  |
| --- | --- |
| **Si se puede** | **NO se puede** |
| public class Main {    public static void main(String[] args) {  **Vehiculo vehiculo1 = new Avion();**  }    } | public class Main {    public static void main(String[] args) {  **Avion a = new Vehiculo();**  }    } |