Cuestiones:

Examina de nuevo la clase Bicho y su clase hija BichoDormilon del apartado anterior y trata de responder las siguientes cuestiones, justificando tu respuesta. Luego comprueba los resultados, ejecutando un programa con las sentencias propuestas:

```
class Bicho {
     public int hambre;
     private int peso;
     Bicho () { hambre =50; peso=50; }
     Bicho (int h, int p) { hambre =h; peso=p;}
     public void come () { hambre -=5; peso++; }
}
class BichoDormilon extends Bicho {
     public int sueño;
     BichoDormilon () { sueño = hambre * 2; }
     BichoDormilon (int i) { super(i,0); sueño = i+20; }
@Override
public void come () {hambre -=10; sueño +=5; }
public void aDormir () {super.come(); sueño =0; }
}
```

a) Vamos a definir una variable Bicho pero llamar sobre ella al constructor de BichoDormilon. ¿Qué mostraría el siguiente código? **Crearíamos una instancia del objeto Bicho el cual es el padre con los datos del objeto hijo**

```
     Bicho bi2 = new BichoDormilon(10); //10,0  30
     System.out.println("bi2 -> hambre: " + bi2.hambre); // 10
     System.out.println("bi2 -> sueño: " + bi2.sueño);   // no se mostrara o daría
error por que la instancia del objeto bicho no tiene acceso al parámetro hambre por
que perteneces a el objeto BichoDormilon.
```

b) ¿Solucionaríamos el problema del printl anterior así?

```
     System.out.println("bi2 -> sueño: " + ((BichoDormilon) bi2).sueño);
     así si se mostraría , por que estamos haciendo un casting de bichoDormilon y
llamando al parámetro.
```

c) ¿Y así?

```
BichoDormilon bd4 = (BichoDormilon) bi2;

System.out.println("bi2 -> sueño: " + bd4.sueño); //25
```

bd4 es una variable que apunta a bi2 con lo cual si que se puede sacar todo lo que se quiera por que se le esta haciendo el casting bi2 entonces si tiene el parámetro sueño accesible.