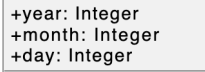


[illegible]

<p>1) Para la clase Fecha descrita en la Figura 1, determine la cantidad de métodos setters y getters que se necesitan. (1pts)</p>	<div data-bbox="1024 426 1227 497">  <pre> classDiagram class Fecha { +year: Integer +month: Integer +day: Integer +create(): Fecha +aaammdd(): String +ddmmaaaa(): String } </pre> </div> <p>Figura 1</p>
---	---

C	Procesador
-cores: Integer	
-cacheL3: Integer	
-threads: Integer	
+create(): Procesador	
+create(_cores: Integer, _cacheL3: Integer, _threads: Integer):Procesador	

Figura 2


 Procesador
-cores: Integer -cacheL3: Integer -threads: Integer
+create(): Procesador +create(_cores: Integer, _cacheL3: Integer, _threads: Integer):Procesador

Figura 3

4) Para la clase descrita en la Figura 4, de un ejemplo de <i>declaración</i> de setter . (1pts)	<div><div><div>C</div>Procesador</div><div><div>-cores: Integer</div><div>-cacheL3: Integer</div><div>-threads: Integer</div></div><div><div>+create(): Procesador</div><div>+create(_cores: Integer, _cacheL3: Integer, _threads: Integer):Procesador</div></div></div> <div>Figura 4</div>
<p>Un setter podría ser:</p> <pre>+ asignarCores(cantidadCores: int): void</pre> <p>La declaración debe incluir el control de acceso, el nombre del método, parámetro y su tipo y el retorno.</p>	
5) Para la clase descrita en la Figura 5, de un ejemplo de <i>declaración</i> de getter . (1pts)	<div><div><div>C</div>Procesador</div><div><div>-cores: Integer</div><div>-cacheL3: Integer</div><div>-threads: Integer</div></div><div><div>+create(): Procesador</div><div>+create(_cores: Integer, _cacheL3: Integer, _threads: Integer):Procesador</div></div></div> <div>Figura 5</div>
<p>Un setter podría ser:</p> <pre>+ retornarCores(): int</pre> <p>La declaración debe incluir el control de acceso, el nombre del método, parámetro y su tipo y el retorno. En este caso, la declaración debe expresar que el método no tiene parámetros.</p>	
6) Considere el diagrama de la Figura 6. Determine si la siguiente instrucción es válida o no. (2pts)	<div><div><div>C</div>Figura</div><div><div>#_area: double</div></div><div><div>+create(): Figura</div><div>+{pure virtual} area(): double</div></div></div> <div>Figura 6</div>
<p>Figura f0 = CREATE_OBJECT Figura()</p> <p>Un método virtual puro es un método que está declarado, pero que no admite implementación en la clase base. Debido a esto, una clase que tiene por lo menos un método virtual puro, no puede instanciar directamente un objeto.</p> <p>Por lo tanto, la instrucción no es válida.</p>	
7) Considere el diagrama de la Figura 7. Determine si la siguiente instrucción es válida o no. (2pts)	<div><div><div>C</div>Animal</div><div><div>#peso: double</div></div><div><div>+create(): Animal</div><div>+{virtual} retornarPeso(): double</div></div></div> <div>Figura 7</div>
<p>Animal a0 = CREATE_OBJECT Animal()</p> <p>Un método virtual es un método que puede ser sobrescrito en las clases hijas y debe estar implementado en la clase base. Debido a esto, la instrucción es válida.</p>	

8) Considere el diagrama de la Figura 8. Implemente en pseudo-código la clase **Animal**. (3pts)

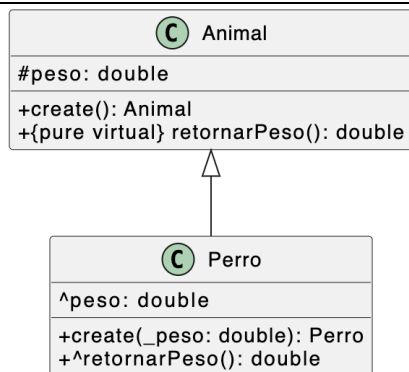


Figura 8

```
class Animal:
    protected:
        peso: double

    public:
        create() {
            //Código constructor
        }

        PURE VIRTUAL retornarPeso()
```

No es necesario que el pseudo código sea idéntico. Sólo debe expresar las mismas ideas.

9) Considere el diagrama de la Figura 9. Implemente en pseudo-código la clase **Perro**. (3pts)

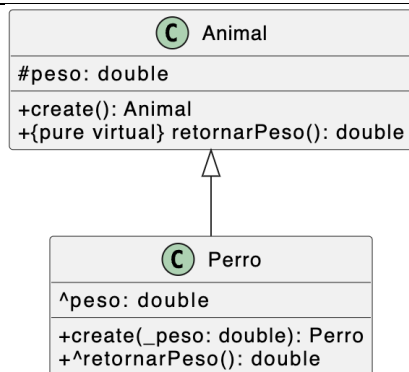


Figura 9

```
class Perro(Animal):
    public:
        create(_peso: double){
            // Código constructor, por lo menos debe incluir la modificación
            // del atributo peso de la clase base.
            peso = _peso
        }

        retornarPeso() {
            return(peso)
        }
```

No es necesario que el pseudo código sea idéntico. Sólo debe expresar las mismas ideas.

10) Considere el diagrama de la Figura 10. Además, se implementa la siguiente función:

```
FUNCTION mostrarPeso(Animal a): Void
    print(a.retornarPeso())
```

y se crea el siguiente objeto:

```
Perro p0 = CREATE_OBJECT Perro(17.5)
```

Determine si la siguiente instrucción es válida o no: `mostrarPeso(p0)` . **(3pts)**

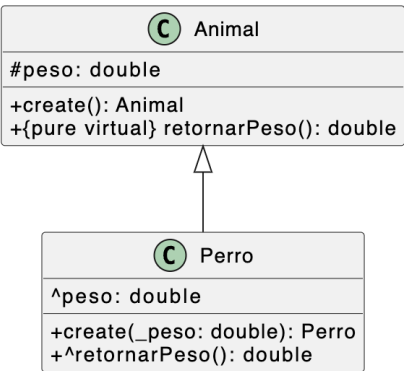


Figura 10

Desde el punto de vista de herencia, la clase Perro es derivada de la clase Animal. Por lo tanto, desde el punto de vista de POO, un objeto de la clase Perro es también un objeto de la clase Animal. Luego, la llamada a la función es válida.

11) Considere el diagrama de la Figura 11. Además, se implementa la siguiente función:

```
FUNCTION mostrarPeso(Animal a): Void
    print(a.retornarPeso())
```

y se crea el siguiente objeto:

```
Animal a0 = CREATE_OBJECT Perro(17.5)
```

Determine si la siguiente instrucción es válida o no: `mostrarPeso(a0)` . **(3pts)**

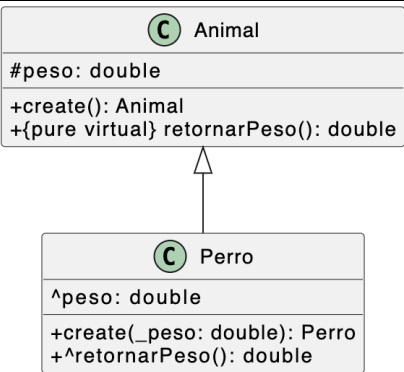


Figura 11

Desde el punto de vista de herencia, la clase Perro es derivada de la clase Animal. Por lo tanto, desde el punto de vista de POO, un objeto de la clase Perro es también un objeto de la clase Animal. Luego, la llamada a la función es válida.

12) Considere el diagrama de la Figura 12. Suponga que las clases **Animal** y **Perro** están correctamente implementadas. Se crean una instancia de cada clase:

```
Animal a0 = CREATE_OBJECT Perro(3.5)
Perro p0 = CREATE_OBJECT Perro(4.5)
```

Determine si las siguientes asignaciones son válidas o no: **(3pts)**

```
a0.peso = 3.7
p0.peso = 4.8
```

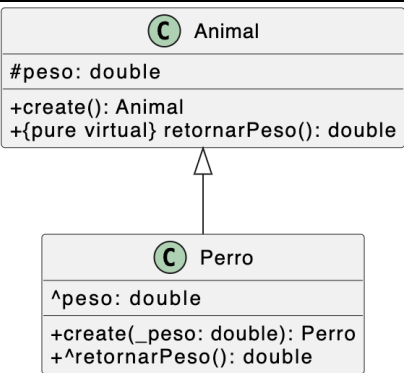


Figura 12

Ambas no son válidas, ya que el atributo peso es protected y no puede ser accesado desde fuera de la jerarquía de la clase donde fue declarado.

13) Considere el pseudo-código adjunto. Este código implementa el diseño de la Figura 13. Determine y corrija los errores de implementación. **(4pts)**

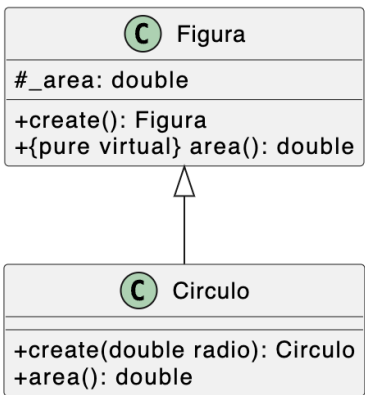


Figura 13

```
class Figura {
private:
    _area;
public:
    create();
    pure virtual area();
};

class Circulo {
public:
    create(radius: double):
        _area = radius*radius*3.1415;

    double area():
        return(_area);
}
```

```
class Figura {
private: protected:
    _area;
public:
    create();
    pure virtual area();
};

class Circulo(Figura) {
public:
    create(radius: double):
        _area = radius*radius*3.1415;

    double area():
        return(_area);
}
```

14) Realice el diagrama de jerarquía para el siguiente párrafo (4pts):

"El rock clásico es influenciado por tres corrientes musicales distintas. La primera es el Blues. Esta corriente musical aporta la progresión de 12 compases. La segunda es el Jazz, que añade la improvisación. La tercera es el Country, que aporte el clásico patrón de dos tiempos por compás. Si bien todas estas corrientes hacen uso de guitarras eléctricas, la característica principal del rock clásico es su sonido potente logrado a través de la distorsión de las notas de la guitarra eléctrica."

