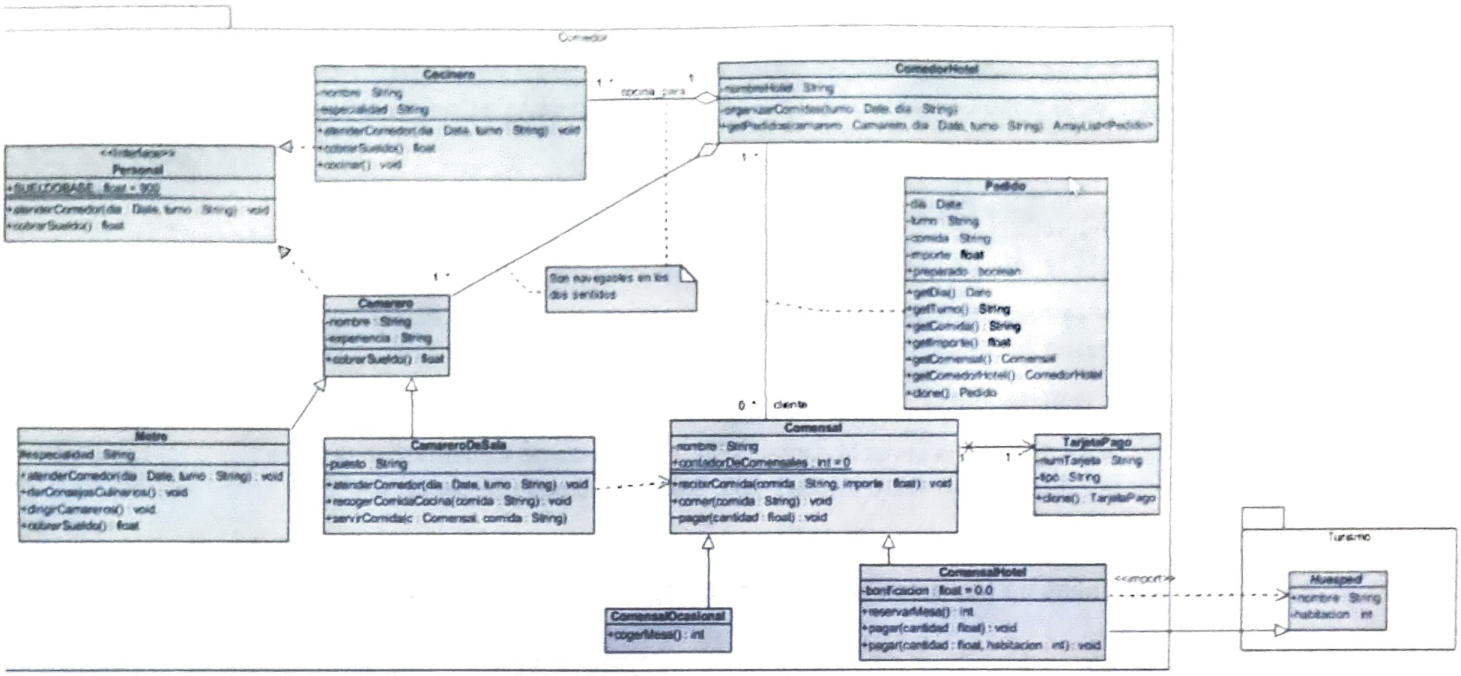

1. Indica verdadero o falso:

- Un objeto de la clase metre tiene tres atributos de instancia.

  Falso, son cuatro en total: especialidad, nombre, experiencia, ComedorHotel (de referencia).

- Se puede declarar una variable de tipo Personal.

  Verdadero

- Se puede crear un objeto de tipo Personal.

  Falso, no se pueden crear objetos de interfaces, ya que una interfaz no es una clase.

- Camarero es una clase abstracta.

  Verdadero, ya que está en cursiva, y además debe implementar todos los métodos de Personal al haberlo implementado

- Un objeto de la clase Comensal tiene 2 atributos de instancia.

  Falso, son tres en total: nombre, un ArrayList de TarjetaPago y un ArrayList de Personal

- Un objeto de la clase ComensalHotel tiene 4 atributos de instancia.

  Verdadero

- En Ruby se puede sólo acceder a atributos privados dentro de la propia clase.

  Falso, cada objeto puede acceder solo a sus atributos, ya sean de la propia clase o heredados

- En Java cada objeto puede acceder solo a sus atributos.

- En Java, para acceder al atributo nombre de un objeto de tipo Metre hay que usar consultores y modificadores.

  Verdadero

- Al declarar una variable de tipo Camarero, es posible asignarle un objeto de tipo metre.

  Verdadero, ya que cumple la regla de compatibilidad de tipos de la programación orientada a objetos

- Un objeto de la clase ComensalHotel puede hacerle un casting a ComensalOcasional

  Falso, no puede hacerse un casting entre hermanos

- Un objeto de la clase Comensal puede hacerle un casting a ComensalHotel

  Verdadero, y al revés también sería posible.

- En Ruby, si se redefine la variable contadorDeComensales de Comensal en la clase ComensalHotel, el cambio de valor solo afecta a ComensalHotel.

  Falso, el cambio afecta también a Comensal y a ComensalOcasional

- En Java, si se redefine la variable contadorDeComensales de Comensal en la clase ComensalOcasional, el cambio de valor solo afecta a ComensalOcasional.

  Verdadero

2. Indica en qué líneas no hay error, hay error de compilación o hay error de ejecución.

```
Personal p = new Metre();
```

No hay error, ya que metre implementa de manera indirecta la interfaz Personal a través de camarero

```
p = new Personal(...);
```

Error, no se puede instanciar una interfaz ya que no es una clase ⟶ Error de compilación

```
Camarero c1,c2;
```

No hay problema, aunque Camarero sea abstracta solo se está declarando

```
c1 = new Metre(...);
```

Correcto, metre hereda de Camarero

```
c2 = new CamareroDeSala(...);
```

Correcto, CamareroDeSala hereda de Camarero

```
c1.cobrarSueldo();
```

Correcto, está en el tipo estático (Camarero), por lo que no hay error de compilación, y en el dinámico (metre), por lo que tampoco de ejecución

```
c1.atenderComedor(...);
```

Correcto, está en el tipo estático (Comedor, aunque declarado en Personal) y en el dinámico (Metre).

```
c1.dirigirCamarero();
```

Error de compilación. Se puede arreglar mediante casting ⟶ ((Metre) c1).dirigirCamarero();

```
c2.servirComida(...);
```

Error de compilación. Se puede arreglar mediante casting ⟶ ((CamareroDeSala) c2).servirComida(..);

```
((Metre) c2).dirigirCamarero();
```

Error de ejecución. CamareroDeSala no tiene ese método.

```
CamareroDeSala c3 = new CamareroDeSala();
```

Correcto

```
((Metre) c3).dirigirCamarero();
```

Error de compilación. No se puede hacer casting entre hermanos

3. Implementa la cabecera de la clase Camarero.

En Java:

```
Package Comedor

abstract class Camarero implements Personal {
    ...
}
```

En Ruby:

```
module Comedor
class Camarero
    ...
end
```

→ método sin implementar

```
def atenderComedor (...)
    raise NotImplementedError.new
end
private_class_method :new → no permite crear objetos
```

4. Implementa en Ruby la clase Metre con su constructor correspondiente y la declaración de sus métodos.

```
module Comedor

class Metre < Camarero
    @especialidad
    def initialize (es, n, exp, comedores)
        super (n, exp, comedores)
        @especialidad = es
    end

    def atenderComedor (dia, turno)
        ...
    end

    def darConsejosCulinarios
        ...
    end

    def dirigirCamareros
        ...
    end

    def cobrarSueldo
        puts "Tienes un plus por responsabilidad"
    end
end
```