


[illegible]

 GeoZone
-latitud: double -longitud: double -largo: double -ancho: double  -area: double
+create(_latitud: double, _longitud:double, _largo:double, _ancho: double): GeoZone +create(z: Geozone): GeoZone  +area(): double

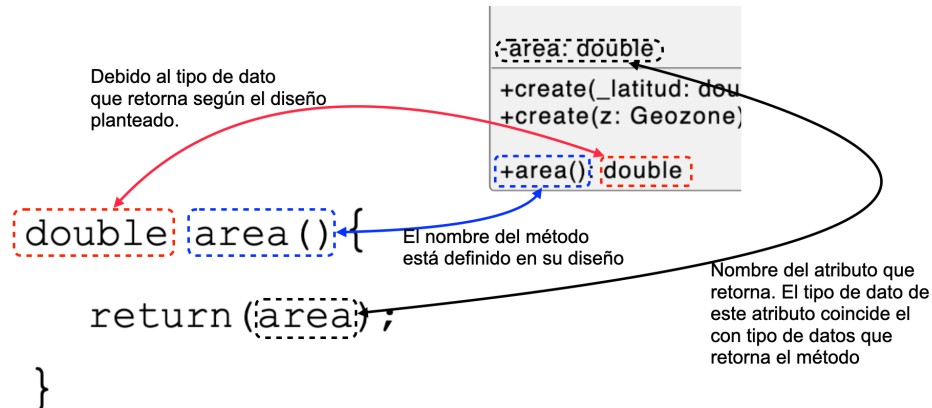
El primer constructor de la clase está implementado de la siguiente forma:

a) Determine, sólo a través de la declaración, si el método `area()` es un setter o un getter. (2pts)

b) Implemente en C++ el método `area()` de la clase `GeoZone`. (2pts)

*Observación: En una implementación real, va a existir un conflicto entre el nombre del método y el nombre de atributo.*

Justificación:



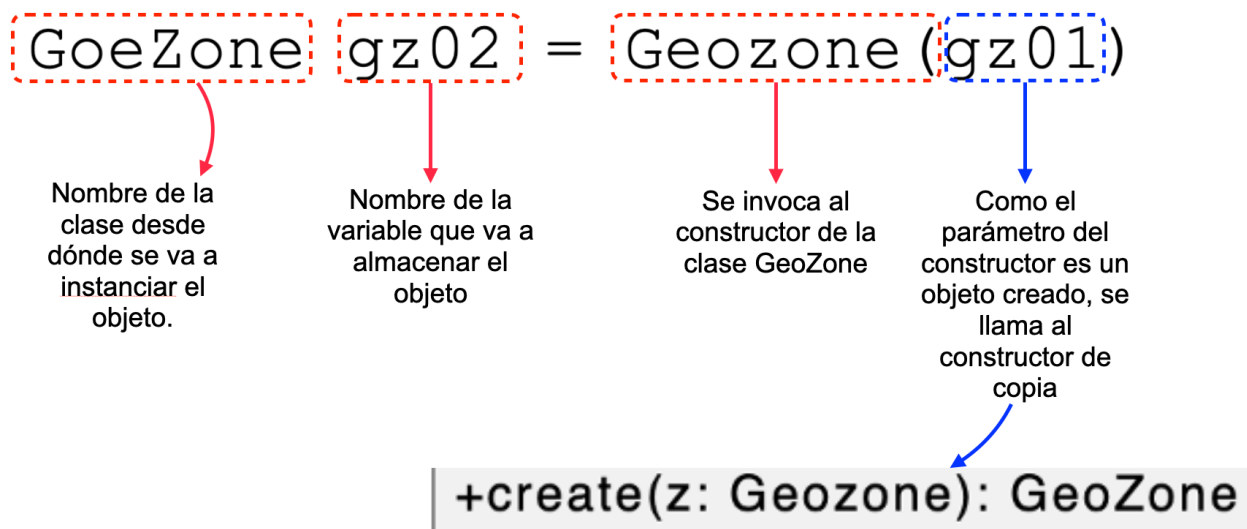
c) Suponga que el constructor de copia está implementado. Además, existe el objeto gz01 instanciado de la siguiente forma:

```
GeoZone gz01 = GeoZone(-33.0458105, -71.6131598, 38.6, 42.85)
```

Implemente la creación de un nuevo objeto, gz02, a partir del objeto gz01; utilizando el constructor de copia. (3pts)

```
GeoZone gz02 = Geozone(gz01)
```

Justificación



d) Implemente en C++ el constructor de copia de la clase. (3pts)

```
GeoZone(const GeoZone& g) {  
    latitud    = g.latitud;  
    longitud   = g.longitud;  
    largo      = g.largo;  
    ancho      = g.ancho;  
    area       = g.area;  
}
```

Justificación:

El constructor siempre tiene el mismo nombre que la clase

Debido a que se pasa por referencia, se informa al compilador que el objeto no debe ser modificado dentro del método.

Lo que recibe el constructor es un objeto GeoZone. Se pasa por referencia para disminuir el tiempo de llamado al método

```
GeoZone(const GeoZone& g) {  
    latitud    = g.latitud;  
    longitud   = g.longitud;  
    largo      = g.largo;  
    ancho      = g.ancho;  
    area       = g.area;  
}
```

Se copian los atributos del objeto origen al nuevo objeto que se está instanciando.