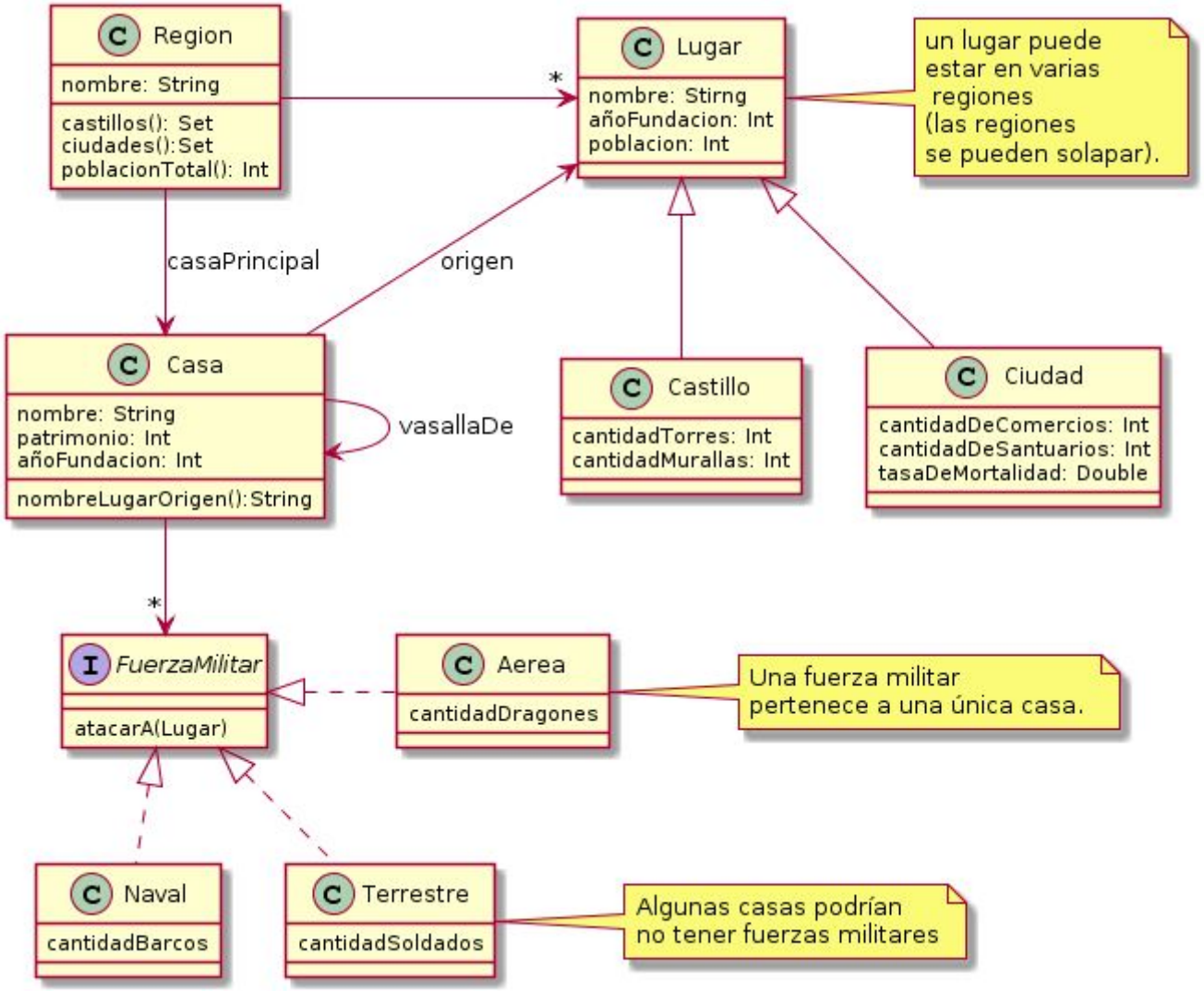
Segundo Parcial - Juego de Tronos

Contexto

Nos encontramos trabajando en el diseño de un juego para celulares basado en el mundo de *Game Of Thrones*, el cual requiere un módulo de estadísticas sobre las regiones geográficas de esta saga, los lugares, y las casas (familias) que allí habitan.

Dicho módulo utilizará una base relacional embebida en el dispositivo, que será exclusivamente accedida desde la aplicación.

Actualmente contamos con un modelo de objetos ya diseñado, del cual tenemos el siguiente diagrama de clases y algunas aclaraciones:



- La aplicación **no** necesita consultar por todos los lugares de una región. En su lugar se piden normalmente o bien todos los castillos de una región, o bien por todas sus ciudades, y la población total de la región

- La aplicación **no** necesita consultar por el lugar origen de una casa, sino tan sólo por el nombre de éste.
- Tanto las casas como las regiones implementan una interfaz `Fundable`, que permite tratarlas polimórficamente según el mensaje `añoFundacion()`
- En un lugar pueden originarse varias casas, y éstas pueden ser la casa principal de varias regiones

A - ORM

Plantear un modelo de datos relacional que permita persistir al modelo de objetos dado, detallar cómo realizaría el mapeo, y justificar las decisiones tomadas sobre los siguientes aspectos:

- Si hubo que hacer cambios al modelo de objetos
- Si se utilizó alguna estrategia de mapeo de herencia, o alguna `@MappedSuperclass`
- Si se embebió (`@Embeddeded`) o enumeró alguna entidad (`@Enumerated`)
- Si se tuvo alguna consideración particular para mantener el orden o duplicados en colecciones
- Si se empleó algún objeto de base de datos como constraints, triggers, vistas o procedimientos

B - Desnormalización

Detectamos que el método `casasImportantes` representa un cuello de botella en nuestra aplicación:

```
clase RepositorioRegiones
  //una casa es importante si es la principal de una región
  // muy poblada y además es rica
  metodo casasImportantes()
      em.createQuery("from Region")
           .list()
           .filter(region => region.poblacionTotal > 5000)
           .map(region => region.casaPrincipal)
           .filter(casa => casa.esRica)
```

```
clase Casa
 metodo esRica
  patrimonio > 10000
```

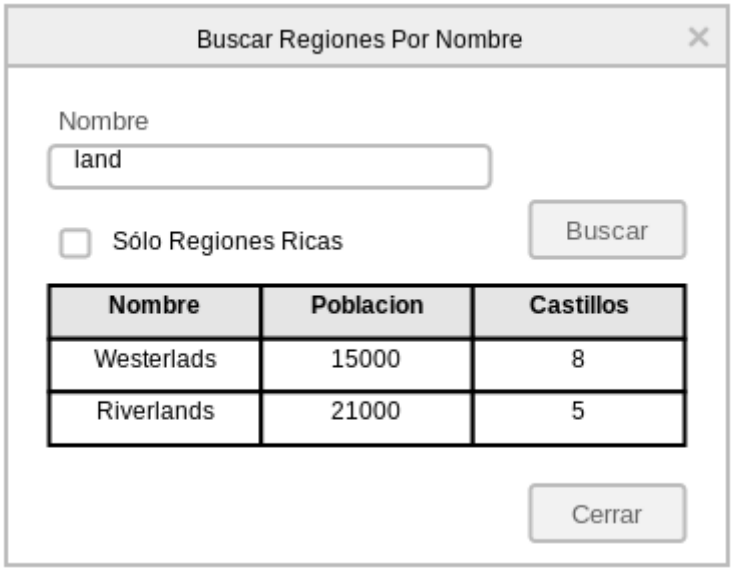
```
clase Region
  metodo poblacionTotal
    em.createQuery(..obtener las poblaciones de cada lugar...)
      .list()
      .sum()
```

Explicar en qué puntos tendría sentido aplicar desnormalización para paliar el problema y qué estrategias se podrían utilizar para mantener la consistencia de los datos.

Considerar que el patrimonio de la casa, la población de los lugares y la casa principal de una región pueden cambiar.

C - MVVM

Necesitamos un formulario que nos permita buscar regiones por nombre:



Para desarrollarlo utilizaremos un framework MVVM. Indicar:

1. cuáles widgets (componentes visuales) deberemos utilizar
2. los eventos involucrados
3. la interfaz del *view model*
4. los bindings involucrados