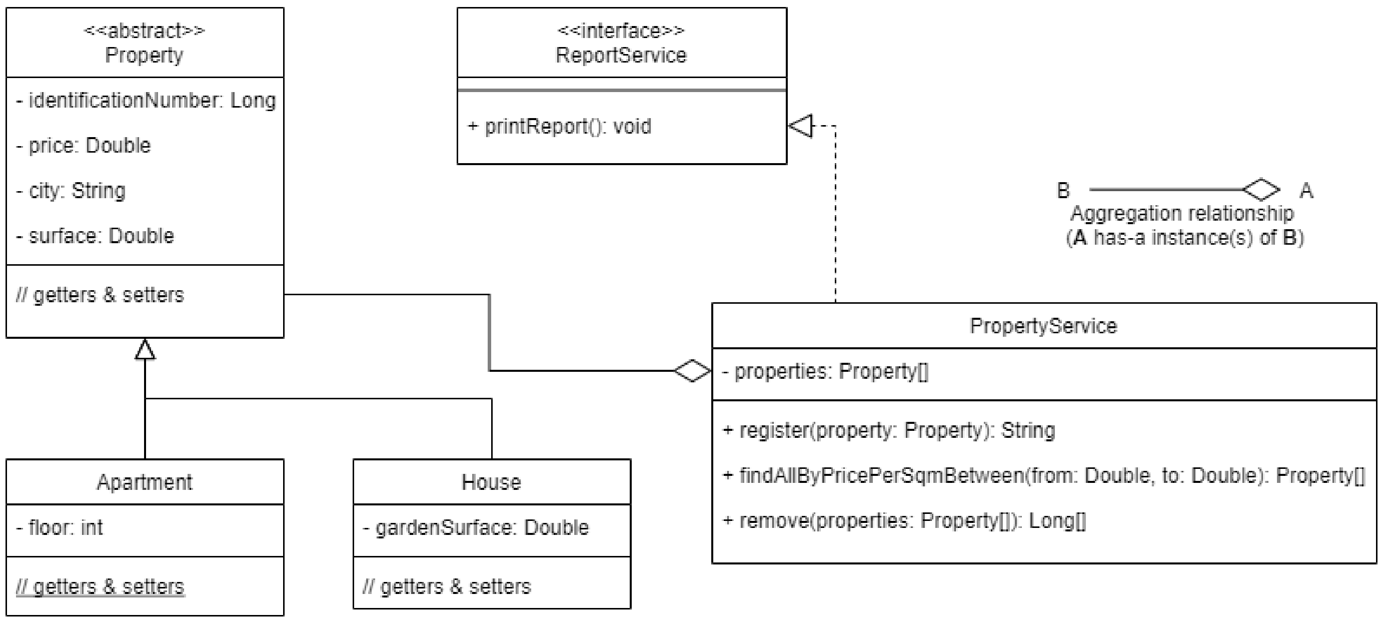
B. Properties

The application you'll have to build today represents a management application to be used inside an estate agency for properties management. There will be two types of properties to work with: `Apartment` and `House`. The functionalities to be built includes property registration, property removal, filtering properties by surface and printing an overall report.

Figure 1: UML diagram



Requirements

1. Read and understand the UML diagram and then create the classes structure as presented in the diagram.
2. Implement the PropertyService register & remove methods by taking into account the following guidelines:

- The `register` method must assign a unique `identificationNumber` long to each property before saving it to the `properties` array. The same string will be returned on successful processing. The `properties` must be always sorted ascending by `city` and then by `price` (if 2 properties have the same `city`);
- If the property has an `identificationNumber` already set on registering then return the `'Identification number already assigned'` message. If the package doesn't have a `price`, `city` or `surface` set then return the `'Invalid data'` message;
- If there is no space left to store another property in the `properties` array then remove the property with the lowest price from the properties array and save the new property afterwards. (preserve the array sorted as mentioned on the first guideline)
- The `remove` method must remove all the properties that are found (search by `identificationNumber`) from the `properties` class attribute;
- The `remove` method must return an array of the properties that were not found to be removed or an empty array otherwise.

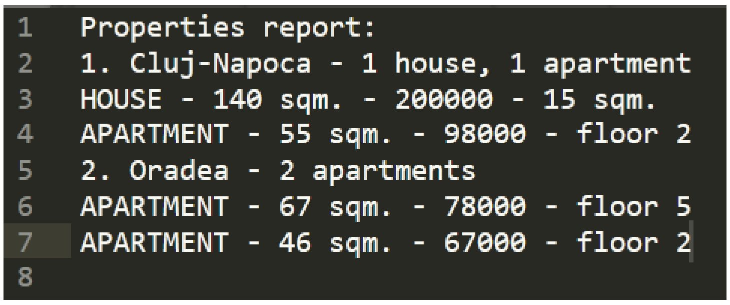
3. Create a class called `Main` where you should provide the `main` static method. Create an instance of `PropertyService` class and then create some instances of the other classes as follows:

- Create an instance of a `House` and two instances of `Apartment`;
- Register them by using the `PropertService register` method;
- Use the `PropertService remove` method on one of the `Apartment` instances.

4. Implement the `findAlByPricePerSqmBetween` method so that it returns an array of `Property` objects that have the **price per square meter** value in between the two values of the `from and to` prices parameters. If either of the method parameters are `null` then use the min/max values for them.

5. Implement the method `printReport()` inside your `PropertyService` class so that it displays the contents of the `properties` vector as shown in the next picture (or similar). Order report ascending by city and for every property display the type (`HOUSE or APARTMENT`) `surface` and `price` plus the extra info for each of the types. (use method overriding for this)

Figure 2: Property report example



```
Properties report:
1. Cluj-Napoca - 1 house, 1 apartment
HOUSE - 140 sqm. - 200000 - 15 sqm.
APARTMENT - 55 sqm. - 98000 - floor 2
2. Oradea - 2 apartments
APARTMENT - 67 sqm. - 78000 - floor 5
APARTMENT - 46 sqm. - 67000 - floor 2
```