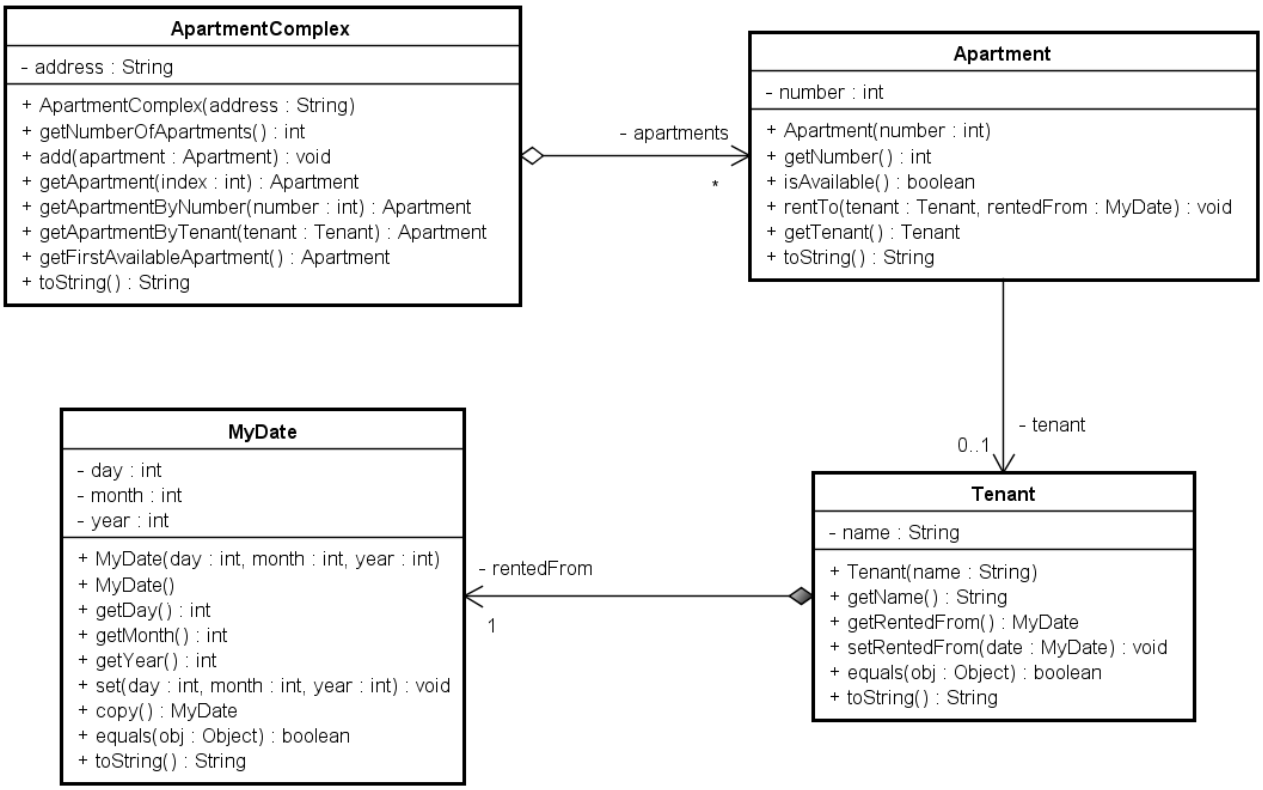
Exercises, SDJ1

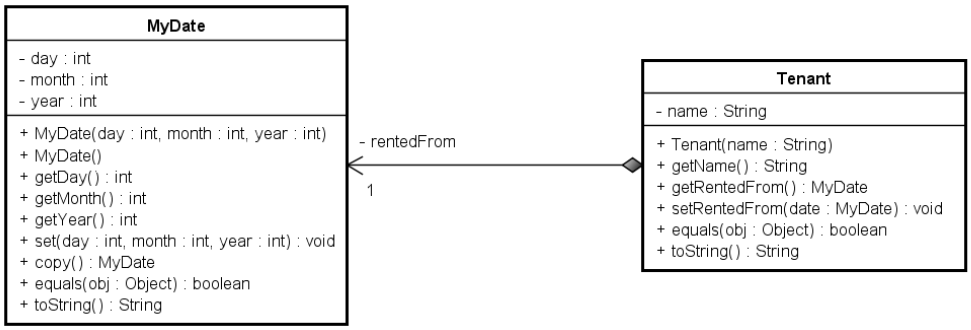
Exercise: Apartment Complex

Implement all classes – and a test class (The following sections present the exercise in smaller parts)



Part 1

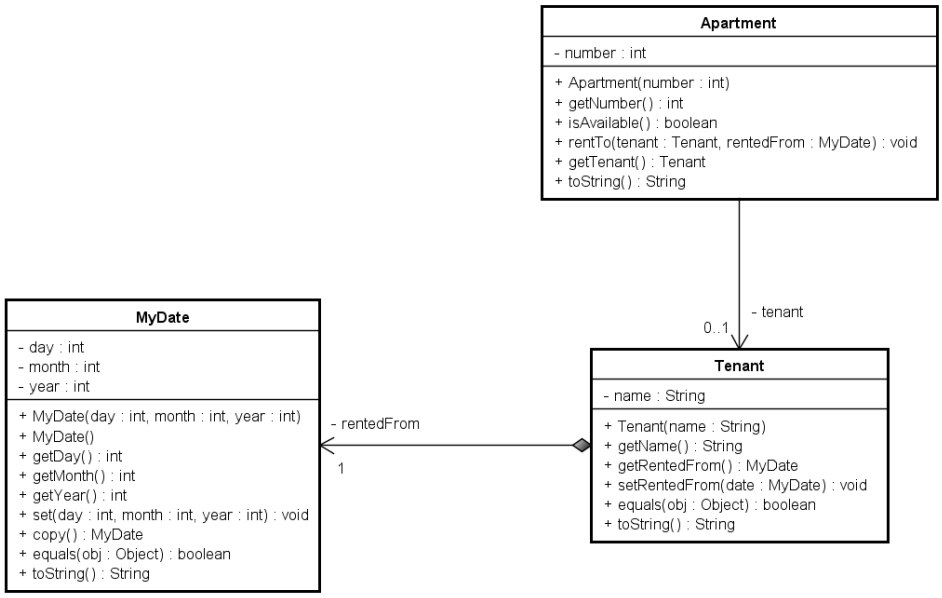
The natural step is to start with the following classes:



- Class `MyDate` is the class you already implemented (could have more methods but at least methods `copy`, `equals`, `toString` and a constructor for today's date)
- Class `Tenant` has one of its two instance variable; a variable of type `MyDate` (named `rentedFrom`)
- Note that because of the multiplicity 1 you have to have exactly one `MyDate` object in `Tenant` and thus, you have to create it in the constructor of `Tenant`
- Note that because of *composition* you have to copy the `MyDate` object in methods taking a `MyDate` as argument or returning a `MyDate` object (`getRentedFrom` and `setRentedFrom`)

Part 2

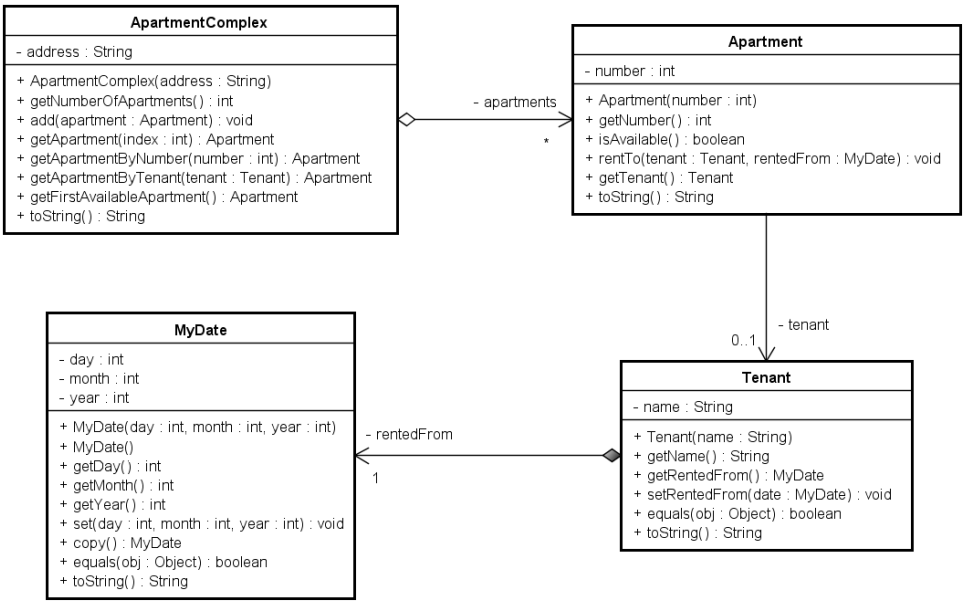
The next step is to implement class `Apartment`:



- Two instance variables, `number` and `tenant`.
- Note that because of the multiplicity `0..1` you could have 0 or 1 one `Tenant` object in `Apartment` and thus, it is legal to set `tenant` (the `Tenant` instance variable) to `null` in the constructor of `Apartment`. A `null` tenant indicates no tenant.
- Method `isAvailable` returns `true` or `false` depending on the `tenant` instance variable.
- Method `rentTo` update the tenant instance variable and call method `setRentedFrom` on it.

Part 3

The next step is to implement the class `ApartmentComplex`:



- The aggregation to `Residence` has the multiplicity `*` and thus, should be implemented as a collection (in our case an array or an `ArrayList`). A closer look shows us that the one making the diagram had an `ArrayList` in mind – how can we see this?

- If we use an `ArrayList` then methods `getNumberOfApartments()`, `add()` and `getApartment(index)` is implemented simply by using delegation to the `ArrayList`.
- The two remaining methods needs a loop.

Part 4

The last step is to implement a test class with a main method.

- In principle, all methods and all branches should be tested for each class during implementation, i.e. Implement `MyDate`, test it, implement `Tenant`, test it, and so forth.
- The final test, testing `ApartmentComplex` you have to create one or more apartment complexes, add some Apartments (including Tenant's) and call other methods in the `ApartmentComplex` class.
- Note: if you declare local variable like `MyDate`-objects, `Tenant`-objects, `Apartment`-objects as part of creating objects to add to the `ApartmentComplex`-object then these objects are unimportant in the sense that they should not be printed out – the only interesting objects would be objects taken from `ApartmentComplex`.