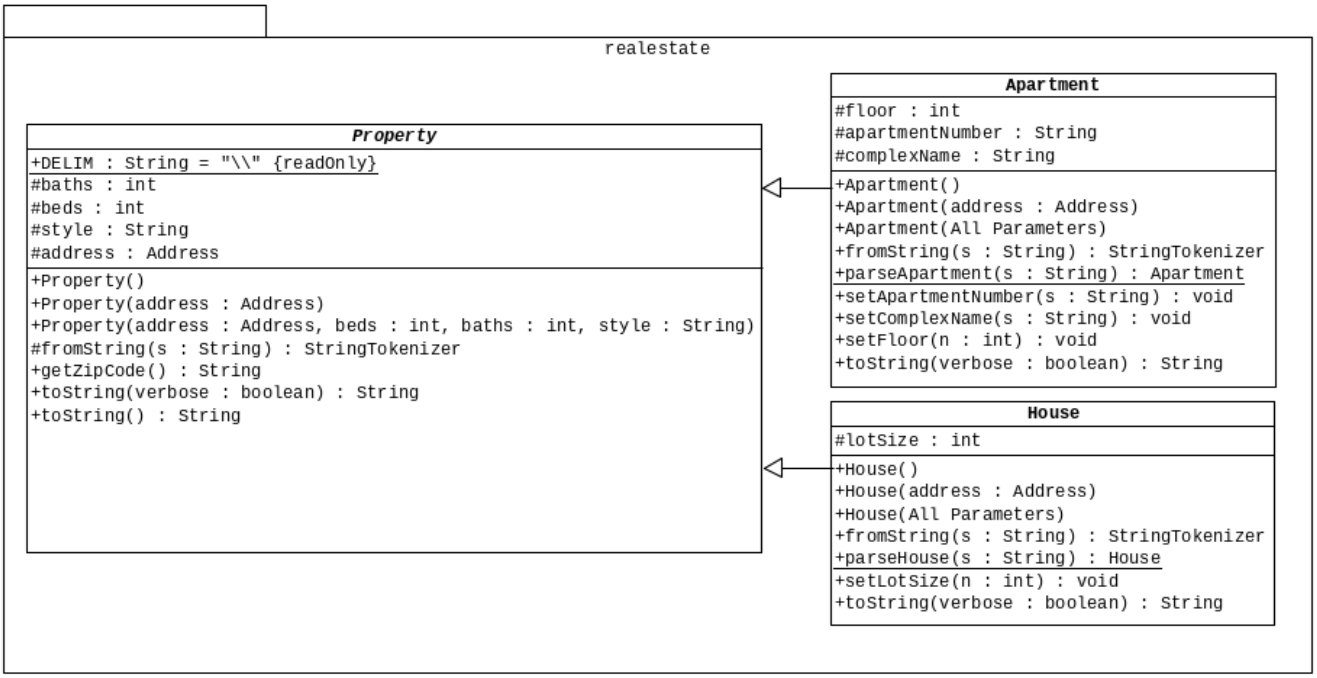
HomeBase

realestate Package v1.0

Class Diagram

The relationships between the various classes and interfaces in this package are illustrated in the following UML class diagram.



In addition to the specifications that are contained in this class diagram, the implementation must comply with the following specifications.

The `Property` Class

Purpose

An encapsulation of an abstract property in a real estate information system.

Constructors

```
Property(Address address)
```

Must construct an instance using `-1` for the `int` parameters, and `"Unknown"` for the `String` parameters.

```
    Property()
```
Must construct an instance using `-1` for the `int` parameters, `"Unknown"` for the `String` parameters, and `null` for the `Address`.

```
    StringTokenizer fromString(String s)
```
Must tokenize the given `String` representation and set the attributes of the owning object accordingly. The `String` representation is delimited using `DELIM` and contains the following fields (in order):

1. Address (a `String` representation)
2. Bedrooms
3. Bathrooms
4. Style

This method must process as much of the `String` as possible until a problem is encountered (so that it can be used with "partial" `String` representations).

```
    getZipCode()
```
Must return the Zip code of the `Address` attribute (if it is non-`null`) and must return `"Unknown"` otherwise.

```
    String toString(boolean verbose)
```
Must return either a verbose or terse `String` representation of the owning `Property`. The verbose representation must be in a format that can be processed by the `fromString()` method (including the `Address`). The terse representation must include the number of beds and baths formatted using `"Bed: %d Bath: %d"` as the format `String`.

```
    String toString()
```
Must return a verbose `String` representation.

The `Apartment` Class

Purpose

An encapsulation of an apartment (in a real estate information system).

Constructors

```
    Apartment(Address address)
```
Must construct an instance using `-1` for the `int` parameters, and `"Unknown"` for the `String` parameters.

```
Apartment()
```

Must construct an instance using `-1` for the `int` parameters, `"Unknown"` for the `String` parameters, and `null` for the `Address`.

```
Apartment(All Parameters)
```

Must have the following parameters:

```
Address address, int beds, int baths, String style, int floor, String apartmentNumber, String complexName
```

Methods

```
StringTokenizer fromString(String s)
```

Must set the attributes of the owning `Apartment` based on the `String` representation it is passed. It must return the `StringTokenizer` that was used to tokenize the `String`. It must call the `fromString()` method in the parent class to parse the "common" portions of the `String` and uses the `StringTokenizer` that is returned to parse the remainder. The remainder of the `String` representation is `DELIM`-delimited and contains the following fields (in order): Floor, Apartment Number, and Apartment Complex Name.

```
void parseApartment(String s)
```

Must construct an `Apartment` object from a `String` representation. It must call the `fromString()` method.

```
String toString(boolean verbose)
```

Must return either a verbose or terse `String` representation of the owning `Apartment`. The verbose representation must be in a format that can be processed by the `fromString()` method. The terse representation must consist of the complex, a space,and the terse representation of a `Property`.

The `House` Class

Purpose

An encapsulation of a house (in a real estate information system).

Constructors

```
House(Address address)
```

Must construct an instance using `-1` for the `int` parameters, and `"Unknown"` for the `String` parameters.

```
House()
```

Must construct an instance using `-1` for the `int` parameters, `"Unknown"` for the `String` parameters, and `null` for the `Address`.

```
House(All Parameters)
```

Must have the following parameters:

`Address address, int beds, int baths, String style, int lotSize`

Methods

```
StringTokenizer fromString(String s)
```

Must set the attributes of the owning `House` based on the `String` representation it is passed. It must return the `StringTokenizer` that was used to tokenize the `String`. It must call the `fromString()` method in the parent class to parse the "common" portions of the `String` and uses the `StringTokenizer` that is returned to parse the remainder. The remainder of the `String` representation is `DELIM`-delimited and contains the following fields (in order): Lot Size.

```
void parseHouse(String s)
```

Must construct an `House` object from a `String` representation. It must call the `fromString()` method.

```
String toString(boolean verbose)
```

Must return either a verbose or terse `String` representation of the owning `House`. The verbose representation must be in a format that can be processed by the `fromString()` method. The terse representation must consist of the terse representation of a `Property` followed by the lot size formatted using `" (%d sq ft)"` as the format `String`.