

# Parser

Address parser, which will be responsible for parsing an input address into a standardize format. This is necessary because the address database stores address data in a certain format. In order to geocode an address, the input address must be turned into a recognizable format first.

## Simple Parsing

```
String addr1 = "Mozilla Corporation, 1981  
second street building K Mountain View CA  
94043-0801";  
System.out.println(AddressParser.parseAddress  
s(addr1));
```

produces

```
{STREET=second, ZIP=94043-0801, TYPE=street,  
NUMBER=1981, STATE=CA, CITY=Mountain View,  
LINE2=building K, NAME=Mozilla Corporation}
```

The parser parses the input address string by recognizing certain pattern of U.S. addresses. It expects the input to be a reasonably formatted address in the format of

<name> <number> <predir> <street> <designator> <postdir> <line 2> <city> <state> <zip>

Almost all of these components are optional, for example, the following are all valid inputs

- 123 main st, philadelphia pa
- 123 main st, apt 20, 19148
- 123 main st 19148
- 123 west main st pa 19148
- 123 main st south philadelphia 19148
- Attn: John Smith, 123 main st apt 2 philadelphia pa
- philadelphia pa
- philadelphia pa 19148

## State Names Spelling Correction

The parser parses the input address by looking for certain separator keywords. For example, the number token is considered as separator between the <name> component and the <street> component. Street designators such as 'st', 'ave' and 'blvd' signals the end of the <street> component. The states are also considered keywords. Keywords are required to spell correctly otherwise the parser won't be able to identify them. For instance, the mis-spelled

'street' in '123 main street' will not be recognized as the street designator because it's mis-spelled. State abbreviations (NJ, PA, DC, etc) are required to have correctly spelling; this requirement is less strict for full state names (new jersey, California, etc), the parser is able to automatically correct these spelling errors.

The parser is able to performs spelling correction on state names (only on spell-out states names, state abbreviations will not be auto corrected). For example,

```
System.out.println(AddressParser.parseAddress("philadelphia PENNSYLVANA"));
```

outputs (mis-spelled 'PENNSYLVANA' is automatically corrected)

```
{CITY=philadelphia, STATE=PENNSYLVANIA}
```

## Disambiguation of Inputs

In most of the normal cases, the parser performs parsing at the syntactical level without considering the semantics of the input. That means that the parser cannot resolve ambiguous inputs such as '123 center lane st valley pa'. Without knowing whether 'valley' or 'st valley' are valid city names in PA, the parser won't be able to make a decision of whether to parse 'st valley' as the city and leave '123 center lane' as the street address or to parse 'valley' as the city and leave '123 center lane st' as the street address.

Many other geocoders have problems dealing with these ambiguous addresses input because they also have a parser that works in a similar fashion. Such geocoders required you to use comma to separate the street address from the city because of this. The parser in JGeocoder will have the same problem if there is no comma in the input that separates the street address form the city but to a much less extent. For instance, the parser is able to correctly parse the inputs

```
123 street st st louis MO  
123 cape may ave cape may court house NJ
```

```
{CITY=st louis, STATE=MO, NUMBER=123,  
TYPE=st, STREET=street}  
{CITY=cape may court house, STATE=NJ,  
NUMBER=123, TYPE=ave, STREET=cape may}
```

## Parsing Address Name and Line 2

Different from many geocoders such as [google map](#) and [geocoder.us](#), JGeocoder's parser is designed to be business address friendly. In business addresses, people often put the name of the business before the first line of the address and also similarly often, they might put the business name after the street line. Here are two examples of such business addresses:

```
1500 desire ave, district court,  
feasterville pa 19053  
district court, 1500 desire ave,  
feasterville pa 19053
```

Try these 2 addresses on google map and you will find that google map is not able to understand them.

JGeocoder's parser on the other hand, is capable of handling such business addresses. With the JGeocoder's parser, you will get the following outputs given the above 2 addresses.

```
{NUMBER=1500, LINE2=district court,  
STATE=pa, CITY=feasterville, TYPE=ave,  
ZIP=19053, STREET=desire}  
{NUMBER=1500, STATE=pa, CITY=feasterville,  
TYPE=ave, ZIP=19053, NAME=district court,  
STREET=desire}
```