Symbol Sequence

Table of contents

1 Overview..2
2 Matching..2

1. Overview

A sequence of symbol is used when each symbol have to map another object type.

Here are a few type of sequence's symbols used in the JPL:

- Sequence of Amino Acids
- Sequence of Ribo/Deoxy Ribonucleotides

We use *Decorators* like `JPLBioSeq` or `JPLPeptide` to wrap the main used symbol sequence. *Decorators* enable to compose method variations and add extension to basic objects.

2. Matching

We propose simple matcher over symbol sequence in util package.

The symbols in the sequence and the type of matcher have to be of the same type. There are many ways to get matches:

```
        JPLMotifMatcher<JPLAminoAcid> matcher;
        JPLSymbolSequence<JPLAminoAcid> sequence;

        sequence =
            new
JPLSymbolSequence.Builder<JPLAminoAcid>("MPEPTMIDEPEPMTIDEMM",
              JPLAASymbol.getSymbolType()).build();

        matcher = JPLMotifMatcher.newInstance("M");
```

There are two possible ways to get positions on the sequence matching the pattern, here is the old way:

```
        matcher.setSequence(sequence);
        List<Integer> list = matcher.iterator().nextToList();
```

or the less error prone way:

```
        List<Integer> list = matcher.iterator(sequence).nextToList();
```

Some more specific parsers exists build on the above models. They are kind of syntaxic sugars ;-)

```
        // the peptide to search motif on
        JPLPeptide peptide =
            new JPLPeptideBuilder("MARVALMMKHGTREEDFFMM").build());

        // the matchers
        JPLAAMatcher simpleMatcher =
            JPLAAMatcher.newInstance(JPLAminoAcid.M);

        List<Integer> indicesFromSimpleMatcher =
            simpleMatcher.iterator(peptide).nextToList();

        List<Integer> indicesFromSimpleMatcher =
            modifMatcher.iterator(peptide).nextToList();
```