1. Main requirements

- We made a clear separation between the following components.
  - Clone detection algorithm
  - Normalization for type 2
  - Visualisation of the detected clones
- Using variable and small functions for crucial parts of the clone detection algorithm. This way we can easily tweak the performance and thresholds of the tool
- We make use of comments and name conventions. This way it is easy to understand the contents of the code.

2. Types of clones

Our clone detection tool can detect clones of type 1 and 2.
For type 1 we make use of basic AST comparison to detect clones. We are using the paper ‘Clone Detection Using Abstract Syntax Trees’ from ‘Ira D. Baxter’.

For clone detection of type 2, we first normalize the AST. We are doing this using the visit statement from rascal. Below are the changes we make to the AST to make it work for clone detection of type 2:

```
case \method(x, _, y, z, q) => \method(x, "method", y, z, q)
case \method(x, _, y, z) => \method(x, "method", y, z)
case \characterLiteral(_) => characterLiteral("characterLiteral")
case \number(_) => \number("1")
case \booleanLiteral(_) => \booleanLiteral(true)
case \stringLiteral(_) => \stringLiteral("stringLiteral")
case \variable(_, x) => \variable("variable", x)
case \variable(_, x, y) => \variable("variable", x, y)
case \simpleName(_) => \simpleName("simpleName")
```

3. Core algorithm

The paper of Ira D. Baxter on clone detection (Clone Detection Using Abstract Syntax Trees) describes how the algorithm works. The following algorithm is the core of our tool:

```
1. Clones=∅
2. For each subtree i:
      If mass(i)>=MassThreshold
      Then hash i to bucket
3. For each subtree i and j in the same bucket
      If CompareTree(i,j) > SimilarityThreshold
         Then { For each subtree s of i
                  If IsMember(Clones,s)
                  Then RemoveClonePair(Clones,s)
                For each subtree s of j
                  If IsMember(Clones,s)
                  Then RemoveClonePair(Clones,s)
                AddClonePair(Clones,i,j)
              }
```

4. Vizualization

The paper ‘Identifying and Removing software clones’ by ‘Rainer Koschke’ describes a view on showing clones in software (see image below). We really liked this view, and attempted to create this in our own tool.
The tool shows an overview of all the classes, and their methods. You get an quick overview of all the clones in the classes.