Version 1.0 September 30, 2013

Imports in ProcessJ

by Matt B. Pedersen

1 Introduction

This document describes the semantics and implementation of the **import** statement.

2 The Import Hierarchy

For a series of imports like

import *std.math*;
import *io.files*;
import *std.random*;

```
.. code ..
```

the import hierarchy (and thus the order in which name usage is resolved) is as follows:

The file that contains the `.. code ..` has the final symbol table in the hierarch, and is thus searched first. If nothing is found there, the symbol table representing the import *std.random* will be searched and so forth until finally the *std.math* symbol table is searched.

If an import using a * is used, the files in the specified package are imported in reverse alphabetical order.

A symbol table has two links: *parent* and *importParent*.
importParent points to the symbol table of the first imported file *parent* point to the next scope which could be another import.

The way name resolution is done for import hierarchies is slightly different than the regular static scoping rules; the reason is that the look up must be *shallow*: if the main file imports a file A which imports a file B which

declares a type T, then this type T should not be available to the main file without an import of B.

Therefore the resolution goes as follows:

when looking for a type or a constant look in the symbol table associated with file; if nothing is found then follow the link to the *importParent*, but from now on follow the *parent* links up the chain to avoid doing any deep searches in the symbol tables for the imported files of other imports.

Such is a search is realized quite easily like this:

```
    // searches the parent chain        public Object get(String name)
{
        Object result = entries.get(name);
        if (result != null)
            return result;
        if (parent == null
            return null;
        return parent.get(name);
    }

    // searches locally then call get with the importParent
    // and get then continues up the parent chain.
    public Object getIncludeImports(String name) {
        Object result = entries.get(name);
        if (result != null)
            return result;
        if (importParent == null)
            return null;
        return importParent.get(name);
    }
```

3 Different Import Types

ProcessJ has two different kinds of import which each come in two different flavors:

1. Single file imports like

(a) **import** f;

(b) **import** $p.f$;

(c) **import** $p_1.p_2.f$;

...

(d) **import** $p_1.p_2.....p_n.f$;

The file import is either a file name by it self, in which case the file must be in the same directory as the file that imports it (and in the same package!), or a number of package names eventually terminated by a file name. the file $p_1.p_2.f$ must be in the directory `p1/p2` and called `f.pj`.

If no local file (relative to where the main file being compiled is), an import from the include directory is attempted.

2. Wild card imports like:

   (a) **import** $p.*$;

   (b) **import** $p_1.p_2.*$;

   ...

   (c) **import** $p_1.p_2.....p_n.*$;

   Again, for say $p_1.p_2.*$, if `p1/p2` does not exist the include directory is searched.

   Wildcard files are imported in reverse alphabetical order, and a wild card import is a deep import. That is, if the directory `p1/p2` contains other directories, these are also visited and any `.pj` file will be imported.

3.1 Example

(The filenames followed by nothing have no imports)
`Main.pj`
import Import.A;
import Import.M;
import Import.N;

`A.pj`
import Import.B;

import Import.C;
import Import.D;

B.pj
import Import.E;
import Import.F;

C.pj

D.pj

E.pj
import Import.Q;
import Import.R;

F.pj
import Import.T;
import Import.S;
import Import.U;

M.pj

N.pj

Q.pj

R.pj

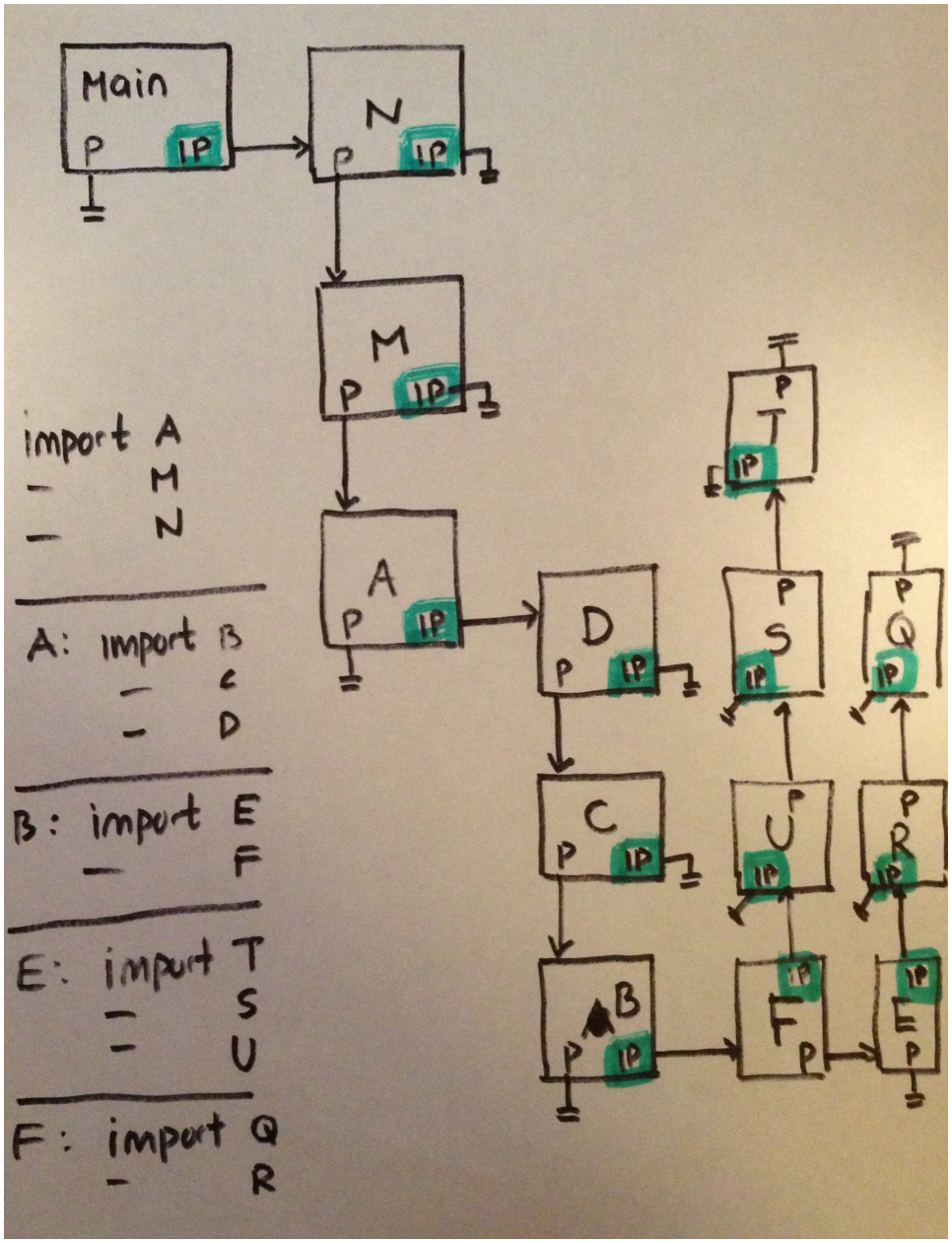
S.pj

T.pj

U.pj

With the switch `-sts` on the compiler we get the following import tree:

```
name.........: Global Type Table
parent.......: --//
importParent.:
|  name.........: Import: /Volumes/Data/Dropbox/ProcessJ/Import/N.pj
|  parent.......:
|  |  name.........: Import: /Volumes/Data/Dropbox/ProcessJ/Import/M.pj
|  |  parent.......:
|  |  |  name.........: Import: /Volumes/Data/Dropbox/ProcessJ/Import/A.pj
|  |  |  parent.......: --//
|  |  |  importParent.:
|  |  |  |  name.........: Import: /Volumes/Data/Dropbox/ProcessJ/Import/D.pj
|  |  |  |  parent.......:
|  |  |  |  |  name.........: Import: /Volumes/Data/Dropbox/ProcessJ/Import/C.pj
|  |  |  |  |  parent.......:
|  |  |  |  |  |  name.........: Import: /Volumes/Data/Dropbox/ProcessJ/Import/B.pj
|  |  |  |  |  |  parent.......: --//
|  |  |  |  |  |  importParent.:
|  |  |  |  |  |  |  name.........: Import: /Volumes/Data/Dropbox/ProcessJ/Import/F.pj
|  |  |  |  |  |  |  parent.......:
|  |  |  |  |  |  |  |  name.........: Import: /Volumes/Data/Dropbox/ProcessJ/Import/E.pj
|  |  |  |  |  |  |  |  parent.......: --//
|  |  |  |  |  |  |  |  importParent.:
|  |  |  |  |  |  |  |  |  name.........: Import: /Volumes/Data/Dropbox/ProcessJ/Import/R.pj
|  |  |  |  |  |  |  |  |  parent.......:
|  |  |  |  |  |  |  |  |  |  name.........: Import: /Volumes/Data/Dropbox/ProcessJ/Import/Q.pj
|  |  |  |  |  |  |  |  |  |  parent.......: --//
|  |  |  |  |  |  |  |  |  |  importParent.: --//
|  |  |  |  |  |  |  |  |  importParent.: --//
|  |  |  |  |  |  |  importParent.:
|  |  |  |  |  |  |  |  name.........: Import: /Volumes/Data/Dropbox/ProcessJ/Import/U.pj
|  |  |  |  |  |  |  |  parent.......:
|  |  |  |  |  |  |  |  |  name.........: Import: /Volumes/Data/Dropbox/ProcessJ/Import/S.pj
|  |  |  |  |  |  |  |  |  parent.......:
|  |  |  |  |  |  |  |  |  |  name.........: Import: /Volumes/Data/Dropbox/ProcessJ/Import/T.pj
|  |  |  |  |  |  |  |  |  |  parent.......: --//
|  |  |  |  |  |  |  |  |  |  importParent.: --//
|  |  |  |  |  |  |  |  |  importParent.: --//
|  |  |  |  |  |  |  importParent.: --//
|  |  |  |  |  importParent.: --//
|  |  |  |  importParent.: --//
|  |  importParent.: --//
|  importParent.: --//
```



Graph version of the import hierarchy.