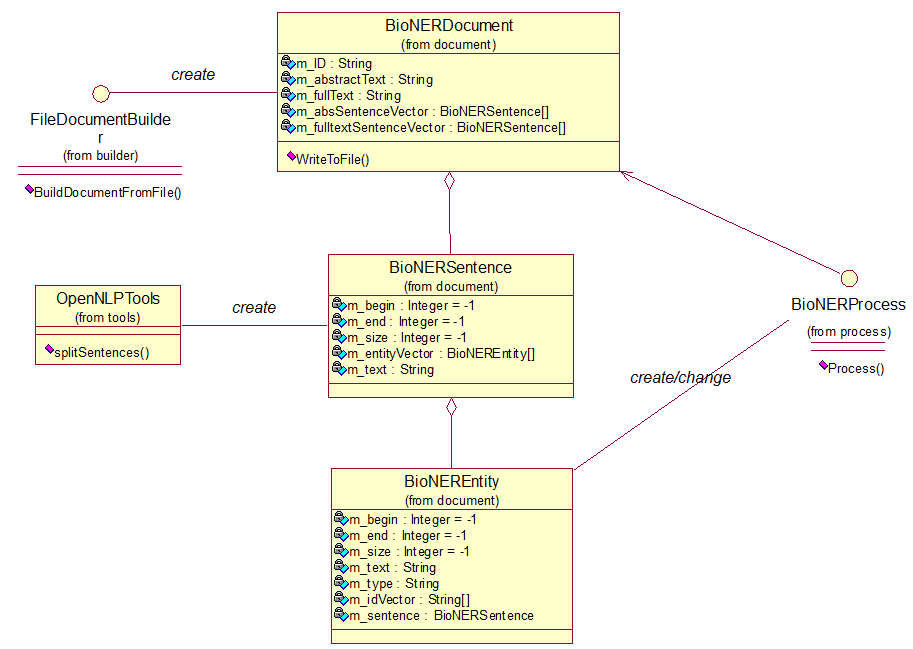
**详细设计说明书**

1. 总体设计

如下图所示：



整个程序围绕三个实体类展开：

BioNERDocument：存储一篇文献的信息。最重要的部分是存储着多个BioNERSentence，代表着文档中的每一个句子。

BioNERSentence：代表文章中的句子。同时其中存储着该句子的位置信息。

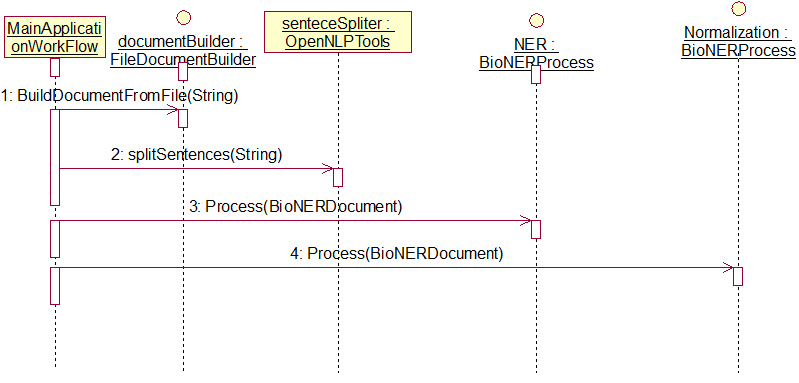
BioNEREntity：识别出来的实体。也存着位置信息，但是这个位置是在所在句子中的。这些对象被存在自己所在的句子（BioNERSentence）中。

这三个类一个包含一个，代表了输出时需要的全部信息。而系统中NER、Normalization等步骤的工作就是在这三个类的对象上操作，填充或修改。这样就将NER、Normalization这些主要步骤抽象成一个统一的接口：BioNERProcess，输入为BioNERDocument对象（BioNERSentence和BioNEREntity也就一起给进去了），在运行结束后其中的信息应该得到相应的修改。

基本流程为：

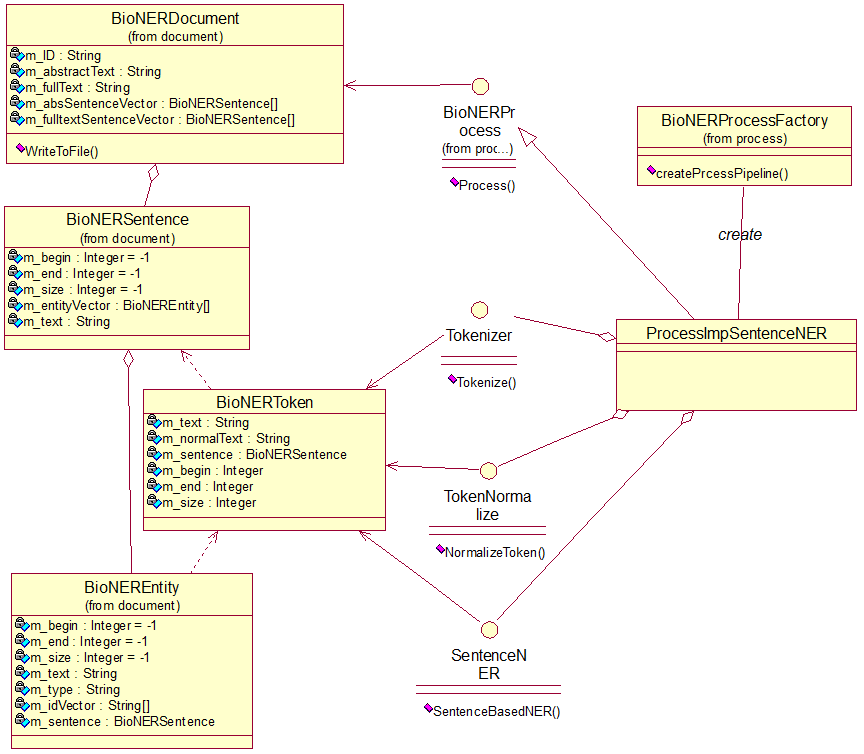
1. 从输入数据建立BioNERDocument、BioNERSentence对象。
2. 得到几个BioNERProcess的实例对象。
3. 将BioNERDocument对象在BioNERProcess的对象中按顺序过一遍，在这个过程中BioNEREntity应该被正确的创建、存储、赋值。
4. 将BioNERDocument输出。

流程图：



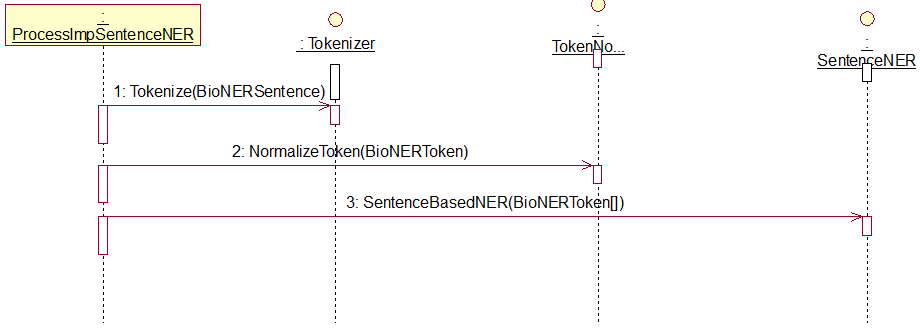
1. NER部分设计

目前实现的是以句子为单位的NER。类图如下：



如图所示，类ProcessImpSentenceNER实现了接口BioNERProcess，提供一个以句子为单位进行NER的功能。拿到一个BioNERDocument对象时，将它其中的每一个句子拿出来依次处理。

这里需要引入一个新的类：BioNERToken，顾名思义就是一个token的相关信息。对于一个句子BioNERSentence，首先通过Tokenizer得到一串BioNERToken，之后通过TokenNormalizer对其标准化（转成小写、处理特殊字符等），最后交给SentenceNER进行NER，得到一串BioNEREntity并将其存入BioNERSentence中。如下图：

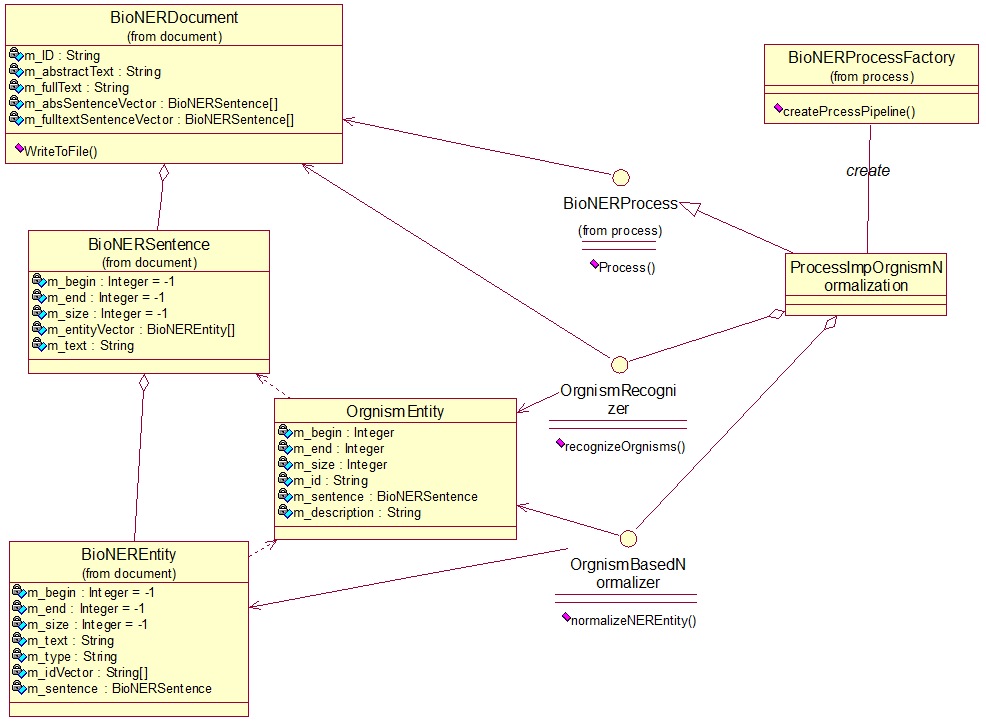


这个部分最重要的是三个接口的实现：Tokenizer、TokenNormalizer和SentenceNER，而其中最重要的就是SentenceNER，也就是NER的方法。如果以后有新的NER方法，只要替换掉这个SentenceNER接口的实现类就可以了。

1. Tokenizer的实现
2. TokenNormalizer的实现
3. SentenceNER的实现
4. Normalization部分设计

这个部分要实现一个基于文章中出现的物种信息对一些entity进行Normalization的功能。也就是说有一些entity经过NER后可能有多个ID，这里要通过文章中出现的一些物种的词汇信息从中挑出一个正确的ID，至少也要尽量减少一个entity中ID的数量。

这个部分的类图如下：



如图所示，这里将Normalization的逻辑封装到两个接口中：OrgnismRecognizer和OrgnismBasedNormalizer，并引入一个新类OrgnismEntity。OrgnismRecognizer用来识别出一篇文章中的所有物种，输入一个BioNERDocument，返回一串OrgnismEntity。OrgnismBasedNormalizer则根据输入的一串OrgnismEntity，对ID多于一个的entity进行ID的消减。

