F:\git\java\mar3\filemonitor\target\jcseg-core\jcseg-core-0.doc

0:F:\git\java\search\jcseg-1.9.5\jcseg-core\pom.xml

```
<?xml version="1.0"?>
<project xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd" xmlns="http://maven.apache.org/POM/4.0.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <modelVersion>4.0.0</modelVersion>

  <parent>
    <groupId>org.lionsoul.jcseg</groupId>
    <artifactId>jcseg</artifactId>
    <version>${jcseg.version}</version>
  </parent>

  <artifactId>jcseg-core</artifactId>
  <name>jcseg-core</name>
  <url>http://code.google.com/p/jcseg</url>

  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <jcseg.version>1.9.5</jcseg.version>
    <maven.test.skip>true</maven.test.skip>
<maven.javadoc.skip>true</maven.javadoc.skip>
  </properties>




  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>3.8.1</version>
      <scope>test</scope>
    </dependency>
  </dependencies>

  <build>
<plugins>
<plugin>
```

```
<artifactId>maven-resources-plugin</artifactId>
<version>2.5</version>
<executions>
<execution>
  <id>properties-file-copy</id>
  <phase>generate-resources</phase>
  <goals>
<goal>copy-resources</goal>
  </goals>
  <configuration>
  <outputDirectory>${basedir}/target/classes</outputDirectory>
  <resources>
 <resource>
<directory>../</directory>
<includes>
<include>jcseg.properties</include>
</includes>
<filtering>true</filtering>
 </resource>
   </resources>
 </configuration>
</execution>
</executions>
</plugin>

<plugin>
<groupId>org.apache.maven.plugins</groupId>
<artifactId>maven-source-plugin</artifactId>
<version>2.1.2</version>
<executions>
  <execution>
<id>attach-sources</id>
<phase>package</phase>
<goals>
  <goal>jar</goal>
</goals>
  </execution>
</executions>
  </plugin>

  <plugin>
<groupId>org.apache.maven.plugins</groupId>
```

```
<artifactId>maven-javadoc-plugin</artifactId>
<version>2.7</version>
<executions>
  <execution>
<id>attach-javadocs</id>
  <goals>
<goal>jar</goal>
  </goals>
  </execution>
</executions>
   </plugin>

  <plugin>
<groupId>org.apache.maven.plugins</groupId>
<artifactId>maven-shade-plugin</artifactId>
<version>1.4</version>
<executions>
  <execution>
<phase>package</phase>
<goals>
  <goal>shade</goal>
</goals>
<configuration>
  <transformers>
<transformer
implementation="org.apache.maven.plugins.shade.resource.ManifestResourceTransformer">
  <mainClass>org.lionsoul.jcseg.test.JcsegTest</mainClass>
</transformer>
  </transformers>
</configuration>
  </execution>
</executions>
  </plugin>
</plugins>
  </build>
</project>
```

1:F:\git\java\search\jcseg-1.9.5\jcseg-core\src\main\java\org\lionsoul\jcseg\ASegment.java

```
package org.lionsoul.jcseg;

import java.io.BufferedReader;
import java.io.IOException;
```

```
import java.io.Reader;
import java.util.ArrayList;

import org.lionsoul.jcseg.core.ADictionary;
import org.lionsoul.jcseg.core.IChunk;
import org.lionsoul.jcseg.core.ILexicon;
import org.lionsoul.jcseg.core.ISegment;
import org.lionsoul.jcseg.core.IWord;
import org.lionsoul.jcseg.core.JcsegTaskConfig;
import org.lionsoul.jcseg.filter.CNNMFilter;
import org.lionsoul.jcseg.filter.ENSCFilter;
import org.lionsoul.jcseg.filter.PPTFilter;
import org.lionsoul.jcseg.util.IHashQueue;
import org.lionsoul.jcseg.util.IPushbackReader;
import org.lionsoul.jcseg.util.IStringBuffer;
import org.lionsoul.jcseg.util.IntArrayList;


/**
 * abstract segment class, implemented ISegment interface
 * implemented all the common method that
 * simple segment and Complex segment algorithm both share.
 *
 * @authorchenxin
 */
public abstract class ASegment implements ISegment
{

/*current position for the given stream.*/
protected int idx;

//protected PushbackReader reader = null;
protected IPushbackReader reader = null;

/*CJK word cache poll*/
//protected LinkedList<IWord> wordPool = null;
protected IHashQueue<IWord> wordPool = null;
protected IStringBuffer isb;
protected IntArrayList ialist;

//Segmentation function control mask
protected int ctrlMask = 0;
```

```
/*the dictionary and task config*/
protected ADictionary dic;
protected JcsegTaskConfig config;

public ASegment( JcsegTaskConfig config,
ADictionary dic ) throws IOException
{
this(null, config, dic);
}

public ASegment( Reader input,
JcsegTaskConfig config,
ADictionary dic ) throws IOException
{
this.config = config;
this.dic = dic;
wordPool = new IHashQueue<IWord>();
isb = new IStringBuffer(64);
ialist = new IntArrayList(15);
reset(input);
}

/**
 * stream/reader reset.
 *
 * @param input
 * @throws IOException
 */
public void reset( Reader input ) throws IOException
{
if ( input != null )
reader = new IPushbackReader(new BufferedReader(input));
idx = -1;
}

/**
 * read the next char from the current position
 * @throws IOException
 */
protected int readNext()
throws IOException
```

```
{
int c = reader.read();
if ( c != -1 ) idx++;
return c;
}

/**
 * push back the data to the stream.
 *
 * @param data
 * @throws IOException
 */
protected void pushBack( int data )
throws IOException
{
reader.unread(data);
idx--;
}

@Override
public int getStreamPosition()
{
return idx + 1;
}

/**
 * set the dictionary of the current segmentor.
 *
 * @paramdic
 */
public void setDict( ADictionary dic )
{
this.dic = dic;
}

/**
 * get the current dictionary instance .
 *
 * @returnADictionary
 */
public ADictionary getDict()
{
```

```
return dic;
}

/**
 * set the current task config .
 *
 * @paramconfig
 */
public void setConfig( JcsegTaskConfig config )
{
this.config = config;
}

/**
 * get the current task config instance.
 *
 * @paramJcsegTaskConfig
 */
public JcsegTaskConfig getConfig()
{
return config;
}

/**
 * @see ISegment#next()
 */
@Override
public IWord next() throws IOException
{
/*
 * @Note: check and get the token directly from the word pool
 * if word pool is available.
 * changed wordPool to IHashQueue for the same Word in wordPool
 * the start position of the word will be the last one
 *
 * @added: 2014-04-11
 */
if ( wordPool.size() > 0 ) return wordPool.remove();

int c, pos;

while ( (c = readNext()) != -1 )
```

```
{
if ( ENSCFilter.isWhitespace(c) ) continue;
pos = idx;

/* CJK string.
 * */
if ( isCJKChar( c ) )
{
char[] chars = nextCJKSentence(c);
int cjkidx = 0;
IWord w = null;
while ( cjkidx < chars.length )
{
/*
 * find the next CJK word.
 * the process will be different with the different algorithm
 * @see getBestCJKChunk() from SimpleSeg or ComplexSeg.
 */
w = null;


//------------------------------------------------------------

/*
 * @istep 1:
 *
 * check if there is chinese numeric.
 * make sure chars[cjkidx] is a chinese numeric
 * andit is not the last word.
 */
if ( CNNMFilter.isCNNumeric(chars[cjkidx]) > -1
&& cjkidx + 1 < chars.length )
{
//get the chinese numeric chars
String num = nextCNNumeric( chars, cjkidx );
int NUMLEN= num.length();

/*
 * check the chinese fraction.
 * old logic: {{{
 * cjkidx + 3 < chars.length && chars[cjkidx+1] == ''
 * && chars[cjkidx+2] == ''
```

```
 * && CNNMFilter.isCNNumeric(chars[cjkidx+3]) > -1.
 * }}}
 *
 * checkCF will be reset to be 'TRUE' it num is a chinese fraction.
 * @added 2013-12-14.
 * */
if ( (ctrlMask & ISegment.CHECK_CF_MASK) != 0  )
{
//get the chinese fraction.
w = new Word(num, IWord.T_CN_NUMERIC);
w.setPosition(pos+cjkidx);
w.setPartSpeech(IWord.NUMERIC_POSPEECH);
wordPool.add(w);

/* Here:
 * Convert the chinese fraction to arabic fraction,
 * if the Config.CNFRA_TO_ARABIC is true.
 */
if ( config.CNFRA_TO_ARABIC )
{
String[] split = num.split("");
IWord wd = new Word(CNNMFilter.cnNumericToArabic(split[1], true)
+"/"+CNNMFilter.cnNumericToArabic(split[0], true),
IWord.T_CN_NUMERIC);
wd.setPosition(w.getPosition());
wd.setPartSpeech(IWord.NUMERIC_POSPEECH);
wordPool.add(wd);
}
}
/*
 * check the chinese numeric and single units.
 * type to find chinese and unit composed word.
 * */
else if ( CNNMFilter.isCNNumeric(chars[cjkidx+1]) > -1
|| dic.match(ILexicon.CJK_UNITS, chars[cjkidx+1]+""))
{
StringBuilder sb = new StringBuilder();
String temp = null;
String ONUM= num;//backup the old num
sb.append(num);
boolean matched = false;
int j;
```

```
//find the word that made up with the numeric
//like:
for ( j = num.length();
(cjkidx + j) < chars.length
&& j < config.MAX_LENGTH; j++ )
{
sb.append(chars[cjkidx + j]);
temp = sb.toString();
if ( dic.match(ILexicon.CJK_WORD, temp) )
{
w = dic.get(ILexicon.CJK_WORD, temp);
num = temp;
matched = true;
}
}

/*
 * @Note: when mached is true, to avoid the start position problem
 * we have to check the word pool the same word is exists or not
 *  if it exist the same word to have to clone the word
 *
 * @added: 2014-04-11
 */
//if ( matched && wordPool.contains(w) )w = w.clone();

IWord wd = null;

/*
 * @Note: when matched is true, num maybe a word like '',
 * yat, this will make it skip the chinese numeric to arabic logic
 * so find the matched word that it maybe a single chinese untis word
 *
 * @added: 2014-06-06
 */
if ( matched == true && num.length() - NUMLEN == 1
&& dic.match(ILexicon.CJK_UNITS, num.substring(NUMLEN)) )
{
num= ONUM;
matched = false;//reset the matched
}
```

```
//find the numeric units
if ( matched == false && config.CNNUM_TO_ARABIC )
{
//get the numeric'a arabic
String arabic = CNNMFilter.cnNumericToArabic(num, true)+"";

if ( (cjkidx + num.length()) < chars.length
&& dic.match(ILexicon.CJK_UNITS,
chars[cjkidx + num.length()]+"" ) )
{
char units = chars[ cjkidx + num.length() ];
num += units; arabic += units;
}

wd = new Word( arabic, IWord.T_CN_NUMERIC);
wd.setPartSpeech(IWord.NUMERIC_POSPEECH);
wd.setPosition(pos+cjkidx);
}

//clear the stop words as need
if ( dic.match(ILexicon.STOP_WORD, num) )
{
cjkidx += num.length();
continue;
}

if ( w == null )
{
w = new Word( num, IWord.T_CN_NUMERIC );
w.setPartSpeech(IWord.NUMERIC_POSPEECH);
}

w.setPosition(pos + cjkidx);
wordPool.add(w);
if ( wd != null ) wordPool.add(wd);

}//end chinese numeric



if ( w != null )
{
cjkidx += w.getLength();
```

```
//add the pinyin to the pool
if ( config.APPEND_CJK_PINYIN
&& config.LOAD_CJK_PINYIN && w.getPinyin() != null )
{
IWord wd = new Word(w.getPinyin(), IWord.T_CJK_PINYIN);
wd.setPosition(w.getPosition());
wordPool.add(wd);
}
//add the syn words to the poll
if ( config.APPEND_CJK_SYN
&& config.LOAD_CJK_SYN && w.getSyn() != null )
{
IWord wd;
for ( int j = 0; j < w.getSyn().length; j++ )
{
wd = new Word(w.getSyn()[j], w.getType());
wd.setPartSpeech(w.getPartSpeech());
wd.setPosition(w.getPosition());
wordPool.add(wd);
}
}

continue;
}

}


//-------------------------------------------------------------
IChunk chunk = getBestCJKChunk(chars, cjkidx);
//System.out.println(chunk+"\n");
//w = new Word(chunk.getWords()[0].getValue(), IWord.T_CJK_WORD);
w = chunk.getWords()[0];

/*
 * @istep 2:
 *
 * find the chinese name.
 */
int T = -1;
if ( config.I_CN_NAME
&& w.getLength() <= 2 && chunk.getWords().length > 1  )
```

```
{
StringBuilder sb = new StringBuilder();
sb.append(w.getValue());
String str = null;

//the w is a Chinese last name.
if ( dic.match(ILexicon.CN_LNAME, w.getValue())
&& (str = findCHName(chars, 0, chunk)) != null)
{
T = IWord.T_CN_NAME;
sb.append(str);
}
//the w is Chinese last name adorn
else if ( dic.match(ILexicon.CN_LNAME_ADORN, w.getValue())
&& chunk.getWords()[1].getLength() <= 2
&& dic.match(ILexicon.CN_LNAME,
chunk.getWords()[1].getValue()))
{
T = IWord.T_CN_NICKNAME;
sb.append(chunk.getWords()[1].getValue());
}
/*
 * the length of the w is 2:
 * the last name and the first char make up a word
 * for the double name.
 */
/*else if ( w.getLength() > 1
&& findCHName( w, chunk ))  {
T = IWord.T_CN_NAME;
sb.append(chunk.getWords()[1].getValue().charAt(0));
}*/

if ( T != -1 )
{
w = new Word(sb.toString(), T);
//if ( config.APPEND_PART_OF_SPEECH )
//w.setPosition(pos+cjkidx);
w.setPartSpeech(IWord.NAME_POSPEECH);
}
}

//check the stopwords(clear it when Config.CLEAR_STOPWORD is true)
```

```
if ( T == -1 && config.CLEAR_STOPWORD
&& dic.match(ILexicon.STOP_WORD, w.getValue()) )
{
cjkidx += w.getLength();
continue;
}


//-------------------------------------------------------

/*
 * @istep 3:
 *
 * reach the end of the chars - the last word.
 * check the existence of the chinese and english mixed word
 */
IWord enAfter = null, ce = null;
if ( ( ctrlMask & ISegment.CHECK_CE_MASk ) != 0
&& (cjkidx + w.getLength() >= chars.length) )
{
//System.out.println("CE-Word"+w.getValue());
enAfter = nextBasicLatin(readNext());
//if ( enAfter.getType() == IWord.T_BASIC_LATIN ) {
String cestr = w.getValue()+enAfter.getValue();

/*
 * here: (2013-08-31 added)
 * also check the stopwords, and make sure
 * the CE word is not a stop words.
 */
if ( ! (config.CLEAR_STOPWORD
&& dic.match(ILexicon.STOP_WORD, cestr))
&& dic.match(ILexicon.CE_MIXED_WORD, cestr) )
{
ce = dic.get(ILexicon.CE_MIXED_WORD, cestr);
//@see comments of ASegment#next
//if ( wordPool.contains(ce) ) ce = ce.clone();

ce.setPosition(pos+cjkidx);
wordPool.add(ce);
cjkidx += w.getLength();
enAfter = null;
```

```
}
//}
}

/*
 * no ce word found, store the english word.
 *
 * @reader: (2013-08-31 added)
 * the newly found letter or digit word "enAfter" token
 * will be handled at last cause we have to handle
 * the pinyin and the syn words first.
 */
if ( ce == null )
{
//@see comment of ASegment#next()
//You may uncomment the following code
//if ( wordPool.contains(w) ) w = w.clone();
w.setPosition(pos+cjkidx);
wordPool.add(w);
cjkidx += w.getLength();
} else {
w = ce;
}


//------------------------------------------------------

/*
 * @istep 4:
 *
 * check and append the pinyin and the syn words.
 */
//add the pinyin to the pool
if ( T == -1 && config.APPEND_CJK_PINYIN
&& config.LOAD_CJK_PINYIN && w.getPinyin() != null )
{
IWord wd = new Word(w.getPinyin(), IWord.T_CJK_PINYIN);
wd.setPosition(w.getPosition());
wordPool.add(wd);
}

//add the syn words to the pool
```

```
String[] syns = null;
if ( T == -1 && config.APPEND_CJK_SYN
&& config.LOAD_CJK_SYN && ( syns = w.getSyn() ) != null )
{
IWord wd;
for ( int j = 0; j < syns.length; j++ )
{
wd = new Word(syns[j], w.getType());
wd.setPartSpeech(w.getPartSpeech());
wd.setPosition(w.getPosition());
wordPool.add(wd);
}
}

//handle the after english word
//generated at the above chinese and english mix word
if ( enAfter != null && ! ( config.CLEAR_STOPWORD
&& dic.match(ILexicon.STOP_WORD,
enAfter.getValue()) ) )
{
//@Note: bug fixed for the position (2014-07-23)
//changed chars.length to pos+chars.length
enAfter.setPosition(pos+chars.length);
//check and to the secondary split.
if ( config.EN_SECOND_SEG
&& ( ctrlMask & ISegment.START_SS_MASK ) != 0 )
enSecondSeg(enAfter, false);
wordPool.add(enAfter);
//append the synoyms words.
if ( config.APPEND_CJK_SYN ) appendLatinSyn(enAfter);
}
}

if ( wordPool.size() == 0 ) continue;
return wordPool.remove();
}
/* english/latin char.
 * */
else if ( isEnChar(c) )
{
IWord w, sword = null;
if ( ENSCFilter.isEnPunctuation( c ) )
```

```
{
String str = ((char)c)+"";
if ( config.CLEAR_STOPWORD
&& dic.match(ILexicon.STOP_WORD, str) ) continue;
w = new Word(str, IWord.T_PUNCTUATION);
w.setPosition(pos);
w.setPartSpeech(IWord.PUNCTUATION);
}
else
{
//get the next basic latin token.
w = nextBasicLatin(c);
w.setPosition(pos);

/* @added: 2013-12-16
 * check and do the seocndary segmentation work.
 * This will split 'qq2013' to 'qq, 2013'.
 * */
if ( config.EN_SECOND_SEG
&& ( ctrlMask & ISegment.START_SS_MASK ) != 0 )
sword = enSecondSeg(w, true);

//clear the stopwords
if ( config.CLEAR_STOPWORD
&& dic.match( ILexicon.STOP_WORD, w.getValue()) )
{
w = null;//Let gc do its work
if ( sword == null ) continue;
}
else
{
/* @added: 2013-12-23.
 * for jcseg-1.9.3 to switch the sub token
 * ahead of the origin one.
 * */
if ( sword != null ) wordPool.add(w);

/* @added: 2013-09-25
 * append the english synoyms words.
 * */
if ( config.APPEND_CJK_SYN ) appendLatinSyn(w);
}
```

```
}

return (sword == null) ? w : sword;
}
/* find a content around with pair punctuations.
 * set the pptmaxlen to 0 to close it
 * */
else if ( config.PPT_MAX_LENGTH > 0
&& PPTFilter.isPairPunctuation( (char) c ) )
{
IWord w = null, w2 = null;
String text = getPairPunctuationText(c);

//handle the punctuation.
String str = ((char)c)+"";
if ( ! ( config.CLEAR_STOPWORD
&& dic.match(ILexicon.STOP_WORD, str) ) )
{
w = new Word(str, IWord.T_PUNCTUATION);
w.setPartSpeech(IWord.PUNCTUATION);
w.setPosition(pos);
}

//handle the pair text.
if ( text != null && ! ( config.CLEAR_STOPWORD
&& dic.match(ILexicon.STOP_WORD, text) ) )
{
w2 = new Word( text, ILexicon.CJK_WORD );
w2.setPartSpeech(IWord.PPT_POSPEECH);
w2.setPosition(pos+1);

if ( w == null ) w = w2;
else wordPool.add(w2);
}

/* here:
 * 1. the punctuation is clear.
 * 2. the pair text is null or being cleared.
 * @date 2013-09-06
 */
if ( w == null && w2 == null ) continue;
```

```
return w;
}
/* letter number like '';
 * */
else if ( isLetterNumber(c) )
{
IWord w = new Word(nextLetterNumber(c), IWord.T_OTHER_NUMBER);
//clear the stopwords
if ( config.CLEAR_STOPWORD
&& dic.match(ILexicon.STOP_WORD, w.getValue()) ) continue;
w.setPartSpeech(IWord.NUMERIC_POSPEECH);
w.setPosition(pos);
return w;
}
/* other number like '';
 * */
else if ( isOtherNumber(c) )
{
IWord w = new Word(nextOtherNumber(c), IWord.T_OTHER_NUMBER);
//clear the stopwords
if ( config.CLEAR_STOPWORD
&& dic.match(ILexicon.STOP_WORD, w.getValue()) ) continue;
w.setPartSpeech(IWord.NUMERIC_POSPEECH);
w.setPosition(pos);
return w;
}
/* chinse punctuation.
 * */
else if ( ENSCFilter.isCnPunctuation( c ) )
{
String str = ((char)c)+"";
if ( config.CLEAR_STOPWORD
&& dic.match(ILexicon.STOP_WORD, str)) continue;
IWord w = new Word(str, IWord.T_PUNCTUATION);
w.setPartSpeech(IWord.PUNCTUATION);
w.setPosition(pos);
return w;
}

/* @reader: (2013-09-25)
 * unrecognized char will cause unknow problem for different system.
 * keep it or clear it ?
```

```
 * if you use jcseg for search, better shut it down.
 * */
else if ( config.KEEP_UNREG_WORDS )
{
String str = ((char)c)+"";
if ( config.CLEAR_STOPWORD
&& dic.match(ILexicon.STOP_WORD, str)) continue;
IWord w = new Word(str, IWord.T_UNRECOGNIZE_WORD);
w.setPartSpeech(IWord.UNRECOGNIZE);
w.setPosition(pos);
return w;
}
}

return null;
}

/**
 * Check and append the synoyms words of
 * specified word included the CJK and basic latin words.
 *
 * All the synoyms words share the same position,
 * part of speech, word type with the primitive word.
 *
 * @paramw
 */
private void appendLatinSyn( IWord w )
{
IWord ew;

/*
 * @added 2014-07-07
 * w maybe EC_MIX_WORD, so check its syn first
 * and make sure it is not a EC_MIX_WORD then check the EN_WORD
 */
if ( w.getSyn() == null )
ew = dic.get(ILexicon.EN_WORD, w.getValue());
else
ew = w;

if (  ew != null && ew.getSyn() != null )
{
```

```
IWord sw = null;
String[] syns = ew.getSyn();
for ( int j = 0; j < syns.length; j++ )
{
sw = new Word(syns[j], w.getType());
sw.setPartSpeech(w.getPartSpeech());
sw.setPosition(w.getPosition());
wordPool.add(sw);
}
}

}

/**
 * Do the secondary split for the specified complex latin word.
 * This will split a complex english, arabic, punctuation
 * compose word to multiple simple parts.
 * Like 'qq2013' will split to 'qq' and '2013' .
 *
 * And all the sub words share the same
 * type and part of speech with the primitive word.
 *
 * You should check the config.EN_SECOND_SEG before invoke this method.
 *
 * @paramw
 * @paramretfwWether to return the fword.
 * @paramIWord - the first sub token for the secondary segment.
 */
public IWord enSecondSeg( IWord w, boolean retfw )
{
//System.out.println("second: "+w.getValue());
isb.clear();
char[] chars = w.getValue().toCharArray();
int _TYPE = ENSCFilter.getEnCharType(chars[0]);
int _ctype, start = 0, j, p = 0;

isb.append(chars[0]);
IWord sword = null, fword = null;//first word
String _str = null;


for ( j = 1; j < chars.length; j++ )
```

```
{
/* get the char type.
 * It could only be one of
 * EN_LETTER, EN_NUMERIC, EN_PUNCTUATION.
 * */
_ctype = ENSCFilter.getEnCharType(chars[j]);
if ( _ctype == ENSCFilter.EN_PUNCTUATION )
{
_TYPE = ENSCFilter.EN_PUNCTUATION;
p++;
continue;
}

if ( _ctype == _TYPE ) isb.append(chars[j]);
else
{
start = j - isb.length() - p;

/* If the number of chars is larger than
 * config.EN_SSEG_LESSLEN we create a new IWord
 * and add to the wordPool.
 * */
if ( isb.length() >= config.STOKEN_MIN_LEN )
{
_str = isb.toString();
//check and clear the stopwords
if ( ! ( config.CLEAR_STOPWORD
&& dic.match(ILexicon.STOP_WORD, _str) ) )
{
sword = new Word(_str, w.getType());
sword.setPartSpeech(w.getPartSpeech());
sword.setPosition(w.getPosition() + start);
if ( retfw && fword == null ) fword = sword;
else wordPool.add(sword);
}
}

isb.clear();
isb.append(chars[j]);
p = 0;
_TYPE = _ctype;
}
```

```
}

//Continue to check the last item.
if ( isb.length() >= config.STOKEN_MIN_LEN )
{
start = j - isb.length() - p;
_str = isb.toString();

if ( ! ( config.CLEAR_STOPWORD
&& dic.match(ILexicon.STOP_WORD, _str) ) )
{
sword = new Word(_str, w.getType());
sword.setPartSpeech(w.getPartSpeech());
sword.setPosition(w.getPosition() + start);
if ( retfw && fword == null ) fword = sword;
else wordPool.add(sword);
}
}

chars = null;//Let gc do its work.

return fword;
}

/**
 * check the specified char is CJK, Thai... char
 * true will be return if it is or return false.
 *
 * @param c
 * @return boolean
 */
static boolean isCJKChar( int c )
{
if ( Character.getType(c) == Character.OTHER_LETTER )
return true;
return false;
}

/**
 * check the specified char is a basic latin and russia and
 * greece letter true will be return if it is or return false.
```

```
 *
 * this method can recognize full-width char and letter.
 *
 * @param c
 * @return boolean
 */
static boolean isEnChar( int c )
{
/*int type = Character.getType(c);
Character.UnicodeBlock cu = Character.UnicodeBlock.of(c);
if ( ! Character.isWhitespace(c) &&
(cu == Character.UnicodeBlock.BASIC_LATIN
|| type == Character.DECIMAL_DIGIT_NUMBER
|| type == Character.LOWERCASE_LETTER
|| type == Character.UPPERCASE_LETTER
|| type == Character.TITLECASE_LETTER
|| type == Character.MODIFIER_LETTER))
return true;
return false;*/
return ( ENSCFilter.isHWEnChar(c) || ENSCFilter.isFWEnChar(c) );
}

/**
 * check the specified char is Letter number like ''
 * true will be return if it is,
 * or return false.
 *
 * @param c
 * @return boolean
 */
static boolean isLetterNumber( int c )
{
if ( Character.getType(c) == Character.LETTER_NUMBER )
return true;
return false;
}

/**
 * check the specified char is other number like ''
 * true will be return if it is,
 * or return false.
 *
```

```
 * @param c
 * @return boolean
 */
static boolean isOtherNumber( int c )
{
if ( Character.getType(c) == Character.OTHER_NUMBER )
return true;
return false;
}

/**
 * match the next CJK word in the dictionary.
 *
 * @param chars
 * @param index
 * @return IWord[]
 */
protected IWord[] getNextMatch(char[] chars, int index)
{

ArrayList<IWord> mList = new ArrayList<IWord>(8);
//StringBuilder isb = new StringBuilder();
isb.clear();

char c = chars[index];
isb.append(c);
String temp = isb.toString();
if ( dic.match(ILexicon.CJK_WORD, temp) ) {
mList.add(dic.get(ILexicon.CJK_WORD, temp));
}

String _key = null;
for ( int j = 1;
j < config.MAX_LENGTH && ((j+index) < chars.length); j++ )
{
isb.append(chars[j+index]);
_key = isb.toString();
if ( dic.match(ILexicon.CJK_WORD, _key) ) {
mList.add(dic.get(ILexicon.CJK_WORD, _key));
}
}
```

```
/*
 * if match no words from the current position
 * to idx+Config.MAX_LENGTH, just return the Word with
 * a value of temp as a unrecognited word.
 */
if ( mList.isEmpty() ) {
mList.add(new Word(temp, ILexicon.UNMATCH_CJK_WORD));
}

/*for ( int j = 0; j < mList.size(); j++ ) {
System.out.println(mList.get(j));
}*/

IWord[] words = new IWord[mList.size()];
mList.toArray(words);
mList.clear();

return words;
}

/**
 * find the chinese name from the position of the given word.
 *
 * @param chars
 * @param index
 * @param chunk
 * @return IWord
 */
protected String findCHName( char[] chars, int index, IChunk chunk )
{
StringBuilder isb = new StringBuilder();
//isb.clear();
/*there is only two IWords in the chunk. */
if ( chunk.getWords().length == 2 )
{
IWord w = chunk.getWords()[1];
switch ( w.getLength() ) {
case 1:
if ( dic.match(ILexicon.CN_SNAME, w.getValue()) )
{
isb.append(w.getValue());
return isb.toString();
```

```
}
return null;
case 2:
case 3:
/*
 * there is only two IWords in the chunk.
 * case 2:
 * like: , chunk: _
 * more: ,chunk: _ (1.6.8)
 * case 3:
 * 1.double name: the two chars and char after it make up a word.
 * like: , chunk: _
 * 2.single name: the char and the two chars after it make up a word. -ignore
 */
String d1 = new String(w.getValue().charAt(0)+"");
String d2 = new String(w.getValue().charAt(1)+"");
if ( dic.match(ILexicon.CN_DNAME_1, d1)
&& dic.match(ILexicon.CN_DNAME_2, d2))
{
isb.append(d1);
isb.append(d2);
return isb.toString();
}
/*
 * the name char of the single name and the char after it
 * make up a word.
 */
else if ( dic.match(ILexicon.CN_SNAME, d1) )
{
IWord iw = dic.get(ILexicon.CJK_WORD, d2);
if ( iw != null && iw.getFrequency()
>= config.NAME_SINGLE_THRESHOLD )
{
isb.append(d1);
return isb.toString();
}
}
return null;
}
}
/*three IWords in the chunk */
else
```

```
{
IWord w1 = chunk.getWords()[1];
IWord w2 = chunk.getWords()[2];
switch ( w1.getLength() ) {
case 1:
/*check if it is a double name first.*/
if ( dic.match(ILexicon.CN_DNAME_1, w1.getValue()) )
{
if ( w2.getLength() == 1 )
{
/*real double name?*/
if ( dic.match(ILexicon.CN_DNAME_2, w2.getValue()) )
{
isb.append(w1.getValue());
isb.append(w2.getValue());
return isb.toString();
}
/*not a real double name, check if it is a single name.*/
else if ( dic.match(ILexicon.CN_SNAME, w1.getValue()) )
{
isb.append(w1.getValue());
return isb.toString();
}
}
/*
 * double name:
 * char 2 and the char after it make up a word.
 * like: , chunk:__ ()
 * like: , chunk:__ (""Config.SINGLE_THRESHOLD)
 * like: , chunk:__ (single name)
 */
else
{
String d1 = new String(w2.getValue().charAt(0)+"");
int index_ = index + chunk.getWords()[0].getLength() + 2;
IWord[] ws = getNextMatch(chars, index_);
//System.out.println("index:"+index+":"+chars[index]+", "+ws[0]);
/*is it a double name?*/
if ( dic.match(ILexicon.CN_DNAME_2, d1) &&
(ws.length > 1 || ws[0].getFrequency()
>= config.NAME_SINGLE_THRESHOLD))
{
```

```
isb.append(w1.getValue());
isb.append(d1);
return isb.toString();
}
/*check if it is a single name*/
else if ( dic.match(ILexicon.CN_SNAME, w1.getValue()) )
{
isb.append(w1.getValue());
return isb.toString();
}
}
}
/*check if it is a single name.*/
else if ( dic.match(ILexicon.CN_SNAME, w1.getValue()) )
{
isb.append(w1.getValue());
return isb.toString();
}
return null;
case 2:
String d1 = new String(w1.getValue().charAt(0)+"");
String d2 = new String(w1.getValue().charAt(1)+"");
/*
 * it is a double name and char 1, char 2 make up a word.
 * like: , chunk: __
 * more: , chunk:__ (1.6.8)
 */
if ( dic.match(ILexicon.CN_DNAME_1, d1)
&& dic.match(ILexicon.CN_DNAME_2, d2))
{
isb.append(w1.getValue());
return isb.toString();
}
/*
 * it is a single name, char 1 and the char after it make up a word.
 */
else if ( dic.match(ILexicon.CN_SNAME, d1) )
{
IWord iw = dic.get(ILexicon.CJK_WORD, d2);
if ( iw != null && iw.getFrequency()
>= config.NAME_SINGLE_THRESHOLD )
{
```

```
isb.append(d1);
return isb.toString();
}
}
return null;
case 3:
/*
 * singe name:  - ignore
 *  mean the char and the two chars after it make up a word.
 *
 * it is a double name.
 * like:  chunk: __
 */
String c1 = new String(w1.getValue().charAt(0)+"");
String c2 = new String(w1.getValue().charAt(1)+"");
IWord w3 = dic.get(ILexicon.CJK_WORD, w1.getValue().charAt(2)+"");
if ( dic.match(ILexicon.CN_DNAME_1, c1)
&& dic.match(ILexicon.CN_DNAME_2, c2)
&& (w3 == null || w3.getFrequency()
>= config.NAME_SINGLE_THRESHOLD))
{
isb.append(c1);
isb.append(c2);
return isb.toString();
}
return null;
}
}

return null;
}

/**
 * find the Chinese double name:
 * when the last name and the first char of the name make up a word.
 *
 * @param chunk the best chunk.
 * @return boolean
 */
@Deprecated
public boolean findCHName( IWord w, IChunk chunk )
{
```

```
String s1 = new String(w.getValue().charAt(0)+"");
String s2 = new String(w.getValue().charAt(1)+"");

if ( dic.match(ILexicon.CN_LNAME, s1)
&& dic.match(ILexicon.CN_DNAME_1, s2))
{
IWord sec = chunk.getWords()[1];
switch ( sec.getLength() )
{
case 1:
if ( dic.match(ILexicon.CN_DNAME_2, sec.getValue()) )
return true;
case 2:
String d1 = new String(sec.getValue().charAt(0)+"");
IWord _w = dic.get(ILexicon.CJK_WORD, sec.getValue().charAt(1)+"");
//System.out.println(_w);
if ( dic.match(ILexicon.CN_DNAME_2, d1)
&& (_w == null
|| _w.getFrequency() >= config.NAME_SINGLE_THRESHOLD ) )
return true;
}
}

return false;
}

/**
 * load a CJK char list from the stream start from the current position.
 * till the char is not a CJK char.
 *
 * @param c
 * @return char[]
 * @throws IOException
 */
protected char[] nextCJKSentence( int c ) throws IOException
{
//StringBuilder isb = new StringBuilder();
isb.clear();
int ch;
isb.append((char)c);

//reset the CE check mask.
```

```
ctrlMask &= ~ISegment.CHECK_CE_MASk;

while ( (ch = readNext()) != -1 )
{
if ( ENSCFilter.isWhitespace(ch) )
{
pushBack(ch);
break;
}

if ( ! isCJKChar(ch) )
{
pushBack(ch);
/*check chinese english mixed word*/
if ( ENSCFilter.isEnLetter(ch) || ENSCFilter.isEnNumeric(ch) )
ctrlMask |= ISegment.CHECK_CE_MASk;
break;
}
isb.append((char)ch);
}

return isb.toString().toCharArray();
}

/**
 * find the letter or digit word from the current position.
 * count until the char is whitespace or not letter_digit.
 *
 * @param c
 * @return IWord
 * @throws IOException
 */
protected IWord nextBasicLatin( int c ) throws IOException
{

isb.clear();
if ( c > 65280 ) c -= 65248;
if ( c >= 65 && c <= 90 ) c += 32;
isb.append((char)c);

int ch;
//EC word, single units control variables.
```

```
boolean _check  = false;
boolean _wspace = false;

//Secondary segmantation
int _ctype = 0;
int tcount = 1;//number of different char type.
int _TYPE  = ENSCFilter.getEnCharType(c);//current char type.
ctrlMask &= ~ISegment.START_SS_MASK;//reset the secondary segment mask.

while ( ( ch = readNext() ) != -1 )
{
//Covert the full-width char to half-width char.
if ( ch > 65280 ) ch -= 65248;
_ctype = ENSCFilter.getEnCharType(ch);

//Whitespace check.
if ( _ctype == ENSCFilter.EN_WHITESPACE )
{
_wspace = true;
break;
}

//English punctuation check.
if ( _ctype == ENSCFilter.EN_PUNCTUATION )
{
if ( ! config.isKeepPunctuation((char)ch) )
{
pushBack(ch);
break;
}
}

//Not EN_KNOW, and it could be letter, numeric.
if ( _ctype == ENSCFilter.EN_UNKNOW )
{
pushBack(ch);
if ( isCJKChar( ch ) ) _check = true;
break;
}

//covert the lower case letter to upper case.
if ( ch >= 65 && ch <= 90 ) ch += 32;
```

```
//append the char to the buffer.
isb.append((char)ch);

/* Char type counter.
 * condition to start the secondary segmentation.
 * @reader: we could do better.
 *
 * @added 2013-12-16
 * */
if ( _ctype != _TYPE )
{
tcount++;
_TYPE = _ctype;
}

}

String __str = isb.toString();
IWord w = null;
boolean chkunits = true;

/*
 * @step 2:
 * 1. clear the useless english punctuations from the end.
 * 2. try to find the english and punctuation mixed word.
 *
 * set _ctype as the status for the existence of punctuation
 * at the end of the isb cause we need to plus the tcount
 * to avoid the secondary check for words like chenxin+, c+.
 */
_ctype = 0;
for ( int t = isb.length() - 1; t > 0
&& isb.charAt(t) != '%'
&& ENSCFilter.isEnPunctuation(isb.charAt(t)); t-- )
{
/*
 * try to find a english and punctuation mixed word.
 * this will clear all the punctuation until a mixed word is found.
 * like "i love c++.", c++ will be found from token "c++.".
 * @date 2013-08-31
 */
```

```
if ( dic.match(ILexicon.EN_PUN_WORD, __str) )
{
w = dic.get(ILexicon.EN_PUN_WORD, __str);
w.setPartSpeech(IWord.EN_POSPEECH);
chkunits = false;
//return w;
break;
}

/*
 * keep the en punctuation.
 * @date 2013-09-06
 */
pushBack(isb.charAt(t));
isb.deleteCharAt(t);
__str = isb.toString();

/*check and plus the tcount.*/
if ( _ctype == 0 )
{
tcount--;
_ctype = 1;
}
}

//condition to start the secondary segmentation.
boolean ssseg = (tcount > 1) && chkunits;

/*@step 3: check the end condition.
 * and the check if the token loop was break by whitespace
 * cause there is no need to continue all the following work if it is.
 *
 * @added 2013-11-19
 * */
if ( ch == -1 || _wspace )
{
w = new Word(__str, IWord.T_BASIC_LATIN);
w.setPartSpeech(IWord.EN_POSPEECH);
if ( ssseg ) ctrlMask |= ISegment.START_SS_MASK;
return w;
}
```

```
if ( ! _check )
{
/* @reader: (2013-09-25)
 * we check the units here, so we can recognize
 * many other units that is not chinese like ',' eg..
 * */
if ( chkunits && ( ENSCFilter.isDigit(__str)
|| ENSCFilter.isDecimal(__str) ) )
{
ch = readNext();
if ( dic.match(ILexicon.CJK_UNITS, ((char)ch)+"") ) {
w = new Word(new String(__str+((char)ch)), IWord.T_MIXED_WORD);
w.setPartSpeech(IWord.NUMERIC_POSPEECH);
}
else pushBack(ch);
}

if ( w == null )
{
w = new Word(__str, IWord.T_BASIC_LATIN);
w.setPartSpeech(IWord.EN_POSPEECH);
if ( ssseg ) ctrlMask |= ISegment.START_SS_MASK;
}

return w;
}


//@step 4: check and get english and chinese mix word like 'B'.
IStringBuffer ibuffer = new IStringBuffer();
ibuffer.append(__str);
String _temp = null;
int mc = 0, j = 0;//the number of char that readed from the stream.

//replace width IntArrayList at 2013-09-08
//ArrayList<Integer> chArr = new ArrayList<Integer>(config.MIX_CN_LENGTH);
ialist.clear();

/* Attension:
 * make sure that (ch = readNext()) is after j < Config.MIX_CN_LENGTH.
 * or it cause the miss of the next char.
 *
```

```
 * @reader: (2013-09-25)
 * we do not check the type of the char readed next.
 * so, words started with english and its length except the start english part
 * less than config.MIX_CN_LENGTH in the EC dictionary could be recongnized.
 */
for ( ; j < config.MIX_CN_LENGTH
&& (ch = readNext()) != -1; j++ )
{
/* Attension:
 *  it is a accident that jcseg works find for
 *  we break the loop directly when we meet a whitespace.
 *  1. if a EC word is found, unit check process will be ignore.
 *  2. if matches no EC word, certianly return of readNext()
 *  will make sure the units check process works find.
 */
if ( ENSCFilter.isWhitespace(ch) )
{
pushBack(ch);
break;
}

ibuffer.append((char)ch);
//System.out.print((char)ch+",");
ialist.add(ch);
_temp = ibuffer.toString();
//System.out.println((j+1)+": "+_temp);
if ( dic.match(ILexicon.EC_MIXED_WORD, _temp) ) {
w = dic.get(ILexicon.EC_MIXED_WORD, _temp);
mc = j + 1;
}
}
ibuffer = null;//Let gc do it's work.

//push back the readed chars.
for ( int i = j - 1; i >= mc; i-- ) pushBack(ialist.get(i));
//chArr.clear();chArr = null;

/* @step 5: check if there is a units for the digit.
 * @reader: (2013-09-25)
 * now we check the units before the step 4, so we can recognize
 * many other units that is not chinese like ','
 * */
```

```
if ( chkunits && mc == 0 )
{
if ( ENSCFilter.isDigit(__str)
|| ENSCFilter.isDecimal(__str) )
{
ch = readNext();
if ( dic.match(ILexicon.CJK_UNITS, ((char)ch)+"") ) {
w = new Word(new String(__str+((char)ch)), IWord.T_MIXED_WORD);
w.setPartSpeech(IWord.NUMERIC_POSPEECH);
} else pushBack(ch);
}
}

/* simply return the combination of english char, arabic
 * numeric, english punctuaton if matches no single units or EC word.
 * */
if ( w == null )
{
w = new Word(__str, IWord.T_BASIC_LATIN);
w.setPartSpeech(IWord.EN_POSPEECH);
if ( ssseg ) ctrlMask |= ISegment.START_SS_MASK;
}

return w;
}

/**
 * find the next other letter from the current position.
 * find the letter number from the current position.
 * count until the char in the specified position is not
 * a letter number or whitespace.
 *
 * @param c
 * @return String
 * @throws IOException
 */
protected String nextLetterNumber( int c ) throws IOException
{
//StringBuilder isb = new StringBuilder();
isb.clear();
isb.append((char)c);
int ch;
```

```
while ( (ch = readNext()) != -1 )
{
if ( ENSCFilter.isWhitespace(ch) )
{
pushBack(ch);
break;
}

if ( ! isLetterNumber( ch ) )
{
pushBack(ch);
break;
}
isb.append((char)ch);
}

return isb.toString();
}

/**
 * find the other number from the current position.
 * count until the char in the specified position is not
 * a orther number or whitespace.
 *
 * @param c
 * @return String
 * @throws IOException
 */
protected String nextOtherNumber( int c ) throws IOException
{
//StringBuilder isb = new StringBuilder();
isb.clear();
isb.append((char)c);
int ch;
while ( (ch = readNext()) != -1 )
{
if ( ENSCFilter.isWhitespace(ch) )
{
pushBack(ch);
break;
}
```

```
if ( ! isOtherNumber(ch) )
{
pushBack(ch);
break;
}
isb.append((char)ch);
}

return isb.toString();
}

/**
 * find the chinese number from the current position.
 * count until the char in the specified position is not
 * a orther number or whitespace.
 *
 * @param chars char array of CJK items.
 * @param index
 * @return String[]
 */
protected String nextCNNumeric(
char[] chars, int index ) throws IOException
{
//StringBuilder isb = new StringBuilder();
isb.clear();
isb.append( chars[ index ]);
ctrlMask &= ~ISegment.CHECK_CF_MASK;//reset the fraction check mask.

for ( int j = index + 1;
j < chars.length; j++ )
{
/* check and deal with '' if the
 * current char is not a chinese numeric.
 * (try to recognize a chinese fraction)
 * @added 2013-12-14
 * */
if ( CNNMFilter.isCNNumeric(chars[j]) == -1 )
{
if ( j + 2 < chars.length
&& chars[j  ] == ''
&& chars[j+1] == ''
/*check and make sure chars[j+2] is a chinese numeric.
```

```
 *  or error will happen on situation like '' .
 *  @added 2013-12-14 */
&& CNNMFilter.isCNNumeric(chars[j+2]) != -1 )
{
isb.append(chars[j++]);
isb.append(chars[j++]);
isb.append(chars[j  ]);
//set the chinese fraction check mask.
ctrlMask |= ISegment.CHECK_CF_MASK;
continue;
}
else
break;
}

//append the buffer.
isb.append( chars[j] );
}

return isb.toString();
}

/**
 * find pair punctuation of the given punctuation char.
 * the purpose is to get the text bettween them.
 *
 * @param c
 * @throws IOException
 */
protected String getPairPunctuationText( int c ) throws IOException
{
//StringBuilder isb = new StringBuilder();
isb.clear();
char echar = PPTFilter.getPunctuationPair( (char) c);
boolean matched = false;
int j, ch;

//replaced with IntArrayList at 2013-09-08
//ArrayList<Integer> chArr = new ArrayList<Integer>(config.PPT_MAX_LENGTH);
ialist.clear();

for ( j = 0; j < config.PPT_MAX_LENGTH; j++ )
```

```
{
ch = readNext();
if ( ch == -1 ) break;
if ( ch == echar )
{
matched = true;
pushBack(ch);//push the pair punc back.
break;
}
isb.append( (char) ch );
ialist.add(ch);
}

if ( matched == false )
{
for ( int i = j - 1; i >= 0; i-- )
pushBack( ialist.get(i) );
return null;
}

return isb.toString();
}

/**
 * an abstract method to gain a CJK word from the
 * current position.
 * simpleSeg and ComplexSeg is different to deal this,
 * so make it a abstract method here.
 *
 * @param  chars
 * @param  index
 * @return IChunk
 * @throws IOException
 */
protected abstract IChunk getBestCJKChunk(char chars[], int index) throws IOException;

}
```

2:F:\git\java\search\jcseg-1.9.5\jcseg-core\src\main\java\org\lionsoul\jcseg\Chunk.java

```
package org.lionsoul.jcseg;

import org.lionsoul.jcseg.core.IChunk;
```

```
import org.lionsoul.jcseg.core.IWord;

/**
 * chunk concept for the mmseg chinese word segment algorithm.
 * has implemented IChunk interface.
 *
 * @authorchenxin
 */
public class Chunk implements IChunk {

/**
 * the word array
 */
private IWord[] words;

/**
 * the average words length
 */
private double averageWordsLength = -1D;

/**
 * the words variance
 */
private double wordsVariance = -1D;

/**
 * single word degree of morphemic freedom
 */
private double singleWordMorphemicFreedom = -1D;

/**
 * words length
 */
private int length = -1;


public Chunk( IWord[] words ) {
this.words = words;
}

/**
 * @see IChunk#getWords()
```

```
 */
@Override
public IWord[] getWords() {
return words;
}

/**
 * @see IChunk#getAverageWordsLength()
 */
@Override
public double getAverageWordsLength() {
if ( averageWordsLength == -1D ) {
averageWordsLength = (double) getLength() / (double) words.length;
}
return averageWordsLength;
}

/**
 * @see IChunk#getWordsVariance()
 */
@Override
public double getWordsVariance() {
if ( wordsVariance == -1D ) {
double variance = 0D, temp;
for ( int j = 0; j < words.length; j++ ) {
temp = (double) words[j].getLength() - getAverageWordsLength();
variance = variance + temp * temp;
}
//wordsVariance = Math.sqrt( variance / (double) words.length );
wordsVariance = variance / words.length;
}
return wordsVariance;
}

/**
 * @see IChunk#getSingleWordsMorphemicFreedom()
 */
@Override
public double getSingleWordsMorphemicFreedom() {
if ( singleWordMorphemicFreedom == -1D ) {
singleWordMorphemicFreedom = 0;
for ( int j = 0; j < words.length; j++ ) {
```

```
//one-character word
if ( words[j].getLength() == 1 ) {
singleWordMorphemicFreedom = singleWordMorphemicFreedom
//+ words[j].getFrequency();
+ Math.log((double) words[j].getFrequency());
}
}
}
return singleWordMorphemicFreedom;
}

/**
 * @see IChunk#getLength()
 */
@Override
public int getLength() {
if ( length == -1 ) {
length = 0;
for ( int j = 0; j < words.length; j++ ) {
length = length + words[j].getLength();
}
}
return length;
}

/**
 * @see Object#toString()
 */
public String toString() {
StringBuilder sb = new StringBuilder();
sb.append("chunk: ");
for ( int j = 0; j < words.length; j++ ) {
sb.append(words[j]+"/");
}
return sb.toString();
}

}
```

3:F:\git\java\search\jcseg-1.9.5\jcseg-core\src\main\java\org\lionsoul\jcseg\ComplexSeg.java

```
package org.lionsoul.jcseg;
```

```
import java.io.IOException;
import java.io.Reader;
import java.util.ArrayList;

import org.lionsoul.jcseg.core.ADictionary;
import org.lionsoul.jcseg.core.IChunk;
import org.lionsoul.jcseg.core.ILexicon;
import org.lionsoul.jcseg.core.IWord;
import org.lionsoul.jcseg.core.JcsegTaskConfig;
//import java.util.Iterator;


/**
 * Complex segment for JCSeg, has implements the ASegment class.
 * this will need the filter work of four filter rule:
 *
 * 1.maximum match chunk.
 * 2.largest average word length.
 * 3.smallest variance of words length.
 * 4.largest sum of degree of morphemic freedom of one-character words.
 *
 *
 * @authorchenxin
 */
public class ComplexSeg extends ASegment {

public ComplexSeg( JcsegTaskConfig config, ADictionary dic ) throws IOException {
super(config, dic);
}

public ComplexSeg( Reader input,
JcsegTaskConfig config, ADictionary dic ) throws IOException {
super(input, config, dic);
}

/**
 * @see ASegment#getBestCJKChunk(char[], int)
 */
@Override
public IChunk getBestCJKChunk(char chars[], int index)
{
```

```
IWord[] mwords = getNextMatch(chars, index), mword2, mword3;
if ( mwords.length == 1
&& mwords[0].getType() == ILexicon.UNMATCH_CJK_WORD ) {
return new Chunk(new IWord[]{mwords[0]});
}

int idx_2, idx_3;
ArrayList<IChunk> chunkArr = new ArrayList<IChunk>();

for ( int x = 0; x < mwords.length; x++ )
{
//the second layer
idx_2 = index + mwords[x].getLength();
if ( idx_2 < chars.length ) {
mword2 = getNextMatch(chars, idx_2);
/*
 * the first try for the second layer
 * returned a UNMATCH_CJK_WORD
 * here, just return the largest length word in
 * the first layer.
 */
if ( mword2.length == 1
&& mword2[0].getType() == ILexicon.UNMATCH_CJK_WORD) {
return new Chunk(new IWord[]{mwords[mwords.length - 1]});
}
for ( int y = 0; y < mword2.length; y++ ) {
//the third layer
idx_3 = idx_2 + mword2[y].getLength();
if ( idx_3 < chars.length ) {
mword3 = getNextMatch(chars, idx_3);
for ( int z = 0; z < mword3.length; z++ ) {
ArrayList<IWord> wArr = new ArrayList<IWord>(3);
wArr.add(mwords[x]);
wArr.add(mword2[y]);
if ( mword3[z].getType() != ILexicon.UNMATCH_CJK_WORD )
wArr.add(mword3[z]);

IWord[] words = new IWord[wArr.size()];
wArr.toArray(words);
wArr.clear();

chunkArr.add(new Chunk(words));
```

```
}
} else {
chunkArr.add(new Chunk(new IWord[]{mwords[x], mword2[y]}));
}
}
} else {
chunkArr.add(new Chunk(new IWord[]{mwords[x]}));
}
}

if ( chunkArr.size() == 1 )
return chunkArr.get(0);

/*Iterator<IChunk> it = chunkArr.iterator();
while ( it.hasNext() ) {
System.out.println(it.next());
}
System.out.println("-+----------------------+-");*/

IChunk[] chunks = new IChunk[chunkArr.size()];
chunkArr.toArray(chunks);
chunkArr.clear();

mwords = null;
mword2 = null;
mword3 = null;

return filterChunks(chunks);
}

/**
 * filter the chunks with the four rule.
 *
 * @param chunks
 * @return IWord
 */
private IChunk filterChunks(IChunk[] chunks) {
//call the maximum match rule.
IChunk[] afterChunks = MMRule.createRule().call(chunks);
if ( afterChunks.length >= 2 ) {
//call the largest average rule.
afterChunks = LAWLRule.createRule().call(afterChunks);
```

```
if ( afterChunks.length >= 2 ) {
//call the smallest variance rule.
afterChunks = SVWLRule.createRule().call(afterChunks);
if ( afterChunks.length >= 2 ) {
//call the largest sum of degree of morphemic freedom rule.
afterChunks = LSWMFRule.createRule().call(afterChunks);
if ( afterChunks.length >= 2 ) {
/*
 * Attention:
 * there is chance for length of the chunks over 2
 * even after the four rules.
 * we use the LASTRule to clear the Ambiguity.
 */
//return LASTRule.createRule().call(afterChunks).getWords()[0];
afterChunks = new IChunk[]{LASTRule.createRule().call(afterChunks)};
}
}
}
}
return afterChunks[0];
}

}
```

4:F:\git\java\search\jcseg-1.9.5\jcseg-core\src\main\java\org\lionsoul\jcseg\core\ADictionary.java

```
package org.lionsoul.jcseg.core;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.FilenameFilter;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.UnsupportedEncodingException;

import org.lionsoul.jcseg.filter.ENSCFilter;



public abstract class ADictionary {
```

```
public static final String AL_TODO_FILE = "lex-autoload.todo";

protected JcsegTaskConfig config;
protected boolean sync;

/**autoload thread */
private Thread autoloadThread = null;

public ADictionary( JcsegTaskConfig config, Boolean sync ) {
this.config = config;
this.sync = sync.booleanValue();
}

public JcsegTaskConfig getConfig() {
return config;
}

public void setConfig( JcsegTaskConfig config ) {
this.config = config;
}

/**
 * load all the words from a specified lexicon file .
 *
 * @paramconfig
 * @paramfile
 */
public void loadFromLexiconFile( File file ) {
loadWordsFromFile(config, this, file, "UTF-8");
}

public void loadFromLexiconFile( String file ) {
loadWordsFromFile(config, this, new File(file), "UTF-8");
}

/**
 * load the all the words form all the files
 * under a specified lexicon directionry .
 *
 * @paramlexDir
 *
```

```
 * @throws IOException
 */
public void loadFromLexiconDirectory( String lexDir ) throws IOException {

File[] files = getLexiconFiles(lexDir,
config.getLexiconFilePrefix(), config.getLexiconFileSuffix());
if ( files == null ) return;

for ( int j = 0; j < files.length; j++ ) {
loadWordsFromFile(config, this, files[j], "UTF-8");
}
}

/**start the lexicon autoload thread .*/
public void startAutoload() {
if ( autoloadThread != null ) return;
autoloadThread = new Thread(new Runnable(){
@Override
public void run() {
File todo = new File(config.getLexiconPath()+"/"+AL_TODO_FILE);
long lastModified = todo.lastModified();
while ( true ) {
//sleep for some time (seconds)
try {
Thread.sleep(config.getPollTime() * 1000);
} catch (InterruptedException e) {break;}

//check the update of the lex-autoload.todo
if ( todo.lastModified() <= lastModified ) continue;

//load words form the lexicon files
try {
BufferedReader reader = new BufferedReader(new FileReader(todo));
String line = null;
while ( ( line = reader.readLine() ) != null ) {
line = line.trim();
if ( line.indexOf('#') != -1 ) continue;
if ( "".equals(line) ) continue;
loadFromLexiconFile(config.getLexiconPath()+"/"+line);
}
reader.close();
FileWriter fw = new FileWriter(todo);
```

```
fw.write("");
fw.close();

lastModified = todo.lastModified();
//System.out.println("newly added words loaded.");
} catch (IOException e) {
break;
}
}
autoloadThread = null;
}
});
autoloadThread.setDaemon(true);
autoloadThread.start();
//System.out.println("lexicon autoload thread started!!!");
}

public void stopAutoload() {
if ( autoloadThread != null ) {
autoloadThread.interrupt();
autoloadThread = null;
}
}

public boolean isSync() {
return sync;
}

/**
 * loop up the dictionary,
 * check the given key is in the dictionary or not.
 *
 * @param t
 * @param key
 * @return true for matched,
 * false for not match.
 */
public abstract boolean match( int t, String key );

/**
 * add a new word to the dictionary.
 *
```

```
     * @param t
     * @param key
     * @param type
     */
    public abstract void add( int t, String key, int type );

    /**
     * add a new word to the dictionary with
     * its statistics frequency.
     *
     * @param t
     * @param key
     * @param fre
     * @param type
     */
    public abstract void add( int t, String key, int fre, int type );

    /**
     * return the IWord asscociate with the given key.
     * if there is not mapping for the key null will be return.
     *
     * @param t
     * @param key
     */
    public abstract IWord get( int t, String key );

    /**
     * remove the mapping associate with the given key.
     *
     * @param t
     * @param key
     */
    public abstract void remove( int t, String key );

    /**
     * return the size of the dictionary
     *
     * @paramt
     * @return int
     */
    public abstract int size(int t);
```

```
/**
 * get the key's type index located in ILexicon interface.
 *
 * @param key
 * @return int
 */
public static int getIndex( String key ) {
if ( key == null )
return -1;
key = key.toUpperCase();

if ( key.equals("CJK_WORDS") )
return ILexicon.CJK_WORD;
else if ( key.equals("CJK_UNITS") )
return ILexicon.CJK_UNITS;
else if ( key.equals("EC_MIXED_WORD") )
return ILexicon.EC_MIXED_WORD;
else if ( key.equals("CE_MIXED_WORD") )
return ILexicon.CE_MIXED_WORD;
else if ( key.equals("CN_LNAME") )
return ILexicon.CN_LNAME;
else if ( key.equals("CN_SNAME") )
return ILexicon.CN_SNAME;
else if ( key.equals("CN_DNAME_1") )
return ILexicon.CN_DNAME_1;
else if ( key.equals("CN_DNAME_2") )
return ILexicon.CN_DNAME_2;
else if ( key.equals("CN_LNAME_ADORN") )
return ILexicon.CN_LNAME_ADORN;
else if ( key.equals("EN_PUN_WORDS") )
return ILexicon.EN_PUN_WORD;
else if ( key.equals("STOP_WORDS") )
return ILexicon.STOP_WORD;
else if ( key.equals("EN_WORD") )
return ILexicon.EN_WORD;

return ILexicon.CJK_WORD;
}

/**
 * get all the lexicon file under the specified path
```

```
 * and meet the specified conditions .
 *
 * @throws IOException
 */
public static File[] getLexiconFiles( String lexDir,
String prefix, String suffix ) throws IOException {

File path = new File(lexDir);
if ( path.exists() == false )
throw new IOException("Lexicon directory ["+lexDir+"] does'n exists.");

/*
 * load all the lexicon file under the lexicon path
 * that start with __prefix and end with __suffix.
 */
final String __suffix = suffix;
final String __prefix = prefix;
File[] files = path.listFiles(new FilenameFilter(){
@Override
public boolean accept(File dir, String name) {
return ( name.startsWith(__prefix)
&& name.endsWith(__suffix));
}
});

return files;
}

/**
 * load all the words in the
 * specified lexicon file into the dictionary.
 *
 * @paramconfig
 * @paramdic
 * @paramfile
 * @paramcharset
 */
public static void loadWordsFromFile(
JcsegTaskConfig config,
ADictionary dic, File file, String charset ) {

InputStreamReader ir = null;
```

```
BufferedReader br = null;

try {
ir = new InputStreamReader(
new FileInputStream( file ), charset);
br = new BufferedReader(ir);

String line = null;
boolean isFirstLine = true;
int t = -1;
while ( (line = br.readLine()) != null ) {
line = line.trim();
if ( "".equals(line) ) continue;
//swept the notes
if ( line.charAt(0) == '#' && line.length() > 1 ) continue;
//the first line fo the lexicon file.
if ( isFirstLine == true ) {
t = ADictionary.getIndex(line);
//System.out.println(line+", "+t);
isFirstLine = false;
if ( t >= 0 ) continue;
}

//handle the stopwords
if ( t == ILexicon.STOP_WORD )
{
if ( line.charAt(0) <= 127 || ( line.charAt(0) > 127
&& line.length() <= config.MAX_LENGTH) )
{
dic.add(ILexicon.STOP_WORD, line, IWord.T_CJK_WORD);
}
continue;
}

//special lexicon
if ( line.indexOf('/') == -1 )
{
/*
 * Here:
 *  1. english and chinese mixed words,
 *  2. chinese and english mixed words,
 *  3. english punctuation words,
```

```
 * don't have to limit its length.
 */
boolean olen = (t == ILexicon.EC_MIXED_WORD);
olen = olen || (t == ILexicon.CE_MIXED_WORD);
olen = olen || (t == ILexicon.EN_PUN_WORD);
if ( olen || line.length() <= config.MAX_LENGTH ) {
dic.add(t, line, IWord.T_CJK_WORD);
}
}
//normal words lexicon file
else
{
String[] wd = line.split("/");

if ( wd.length < 4 ) {//format check
System.out.println("Lexicon File: " + file.getAbsolutePath()
+ "#" + wd[0] + " format error. -ignored");
continue;
}
if ( wd.length == 5 ) {//single word degree check.
if ( ! ENSCFilter.isDigit(wd[4]) ) {
System.out.println("Lexicon File: " + file.getAbsolutePath()
+ "#" + wd[0] + " format error(single word " +
"degree should be an integer). -ignored");
continue;
}
}

//length limit(CJK_WORD only)
if ( ( t == ILexicon.CJK_WORD )
&& wd[0].length() > config.MAX_LENGTH ) continue;

if ( dic.get(t, wd[0]) == null ) {
if ( wd.length > 4 )
dic.add(t, wd[0], Integer.parseInt(wd[4]), IWord.T_CJK_WORD);
else
dic.add(t, wd[0], IWord.T_CJK_WORD);
}

IWord w = dic.get(t, wd[0]);
//System.out.println(wd.length);
//set the pinying of the word.
```

```
if ( config.LOAD_CJK_PINYIN && ! "null".equals(wd[2]) ) {
w.setPinyin(wd[2]);
}

boolean li = ( t == ILexicon.CJK_WORD );

//set the syn words.
String[] arr = w.getSyn();
if ( config.LOAD_CJK_SYN && ! "null".equals(wd[3]) ) {
String[] syns = wd[3].split(",");
for ( int j = 0; j < syns.length; j++ ) {
syns[j] = syns[j].trim();
/* Here:
 * filter the syn words that its length
 * is greater than Config.MAX_LENGTH
 */
if ( li && syns[j].length() > config.MAX_LENGTH ) continue;
/* Here:
 * check the syn word is not exists, make sure
 * the same syn word won't appended. (dictionary reload)
 *
 * @date 2013-09-02
 */
if ( arr != null ) {
int length = arr.length;
boolean add = true;
for ( int i = 0; i < length; i++ )  {
if ( syns[j].equals(arr[i]) ) {
add = false;
break;
}
}
if ( ! add ) continue;
}
w.addSyn(syns[j]);
}
}

//set the word's part of speech
arr = w.getPartSpeech();
if ( config.LOAD_CJK_POS && ! "null".equals(wd[1]) ) {
String[] pos = wd[1].split(",");
```

```
for ( int j = 0; j < pos.length; j++ ) {
pos[j] = pos[j].trim();
/* Here:
 * check the part of speech is not exists, make sure
 * the same part of speech won't appended.(dictionary reload)
 *
 * @date 2013-09-02
 */
if ( arr != null ) {
int length = arr.length;
boolean add = true;
for ( int i = 0; i < length; i++ )  {
if ( pos[j].equals(arr[i]) ) {
add = false;
break;
}
}
if ( ! add ) continue;
}
w.addPartSpeech(pos[j].trim());
}
}
}
}
} catch (UnsupportedEncodingException e) {
e.printStackTrace();
} catch (IOException e) {
e.printStackTrace();
} finally {
try {
if ( ir != null ) ir.close();
if ( br != null ) br.close();
} catch (IOException e) {}
}
}
}
```

5:F:\git\java\search\jcseg-1.9.5\jcseg-core\src\main\java\org\lionsoul\jcseg\core\DictionaryFactory.java

```
package org.lionsoul.jcseg.core;

import java.io.IOException;
```

```
import java.lang.reflect.Constructor;

/**
 * Dictionary Factory to create Dictionary instance .
 * a path of the class that has extends the ADictionary
 * class must be given first.
 *
 * @authorchenxin
 */
public class DictionaryFactory {

private DictionaryFactory() {}

/**
 * create a new ADictionary instance .
 *
 * @param __dicClass
 * @returnADictionary
 */
public static ADictionary createDictionary(
String __dicClass, Class<?>[] paramType, Object[] args) {
try {
Class<?> _class = Class.forName(__dicClass);
Constructor<?> cons = _class.getConstructor(paramType);
return ( ( ADictionary ) cons.newInstance(args) );
} catch ( Exception e ) {
System.err.println("can't create the ADictionary instance " +
"with classpath ["+__dicClass+"]");
e.printStackTrace();
}
return null;
}

/**
 * create a default ADictionary instance of class
 * com.webssky.jcseg.Dictionary .
 *
 * @seeDictionary
 * @returnADictionary
 */
public static ADictionary createDefaultDictionary( JcsegTaskConfig config, boolean sync ) {
ADictionary dic = createDictionary("org.lionsoul.jcseg.Dictionary",
```

```
new Class[]{JcsegTaskConfig.class, Boolean.class},
new Object[]{config, sync});
try {
//load lexicon from more than one path.
String[] lexpath = config.getLexiconPath();
if ( lexpath == null )
throw new IOException("Invalid lexicon path, " +
"make sure the JcsegTaskConfig is initialized.");

//load word item from all the directories.
for ( String lpath : lexpath )
dic.loadFromLexiconDirectory(lpath);
if ( dic.getConfig().isAutoload() ) dic.startAutoload();
} catch (IOException e) {
e.printStackTrace();
}
return dic;
}

public static ADictionary createDefaultDictionary( JcsegTaskConfig config ) {
return createDefaultDictionary(config, config.isAutoload());
}
}
```

6:F:\git\java\search\jcseg-1.9.5\jcseg-core\src\main\java\org\lionsoul\jcseg\core\IChunk.java

```
package org.lionsoul.jcseg.core;

/**
 * chunk interface for JCSeg.
 * the most important concept for the mmseg chinese segment alogorithm.
 *
 * @authorchenxin
 */
public interface IChunk {
/**
 * get the all the words in the chunk.
 *
 * @return IWord[]
 */
public IWord[] getWords();

/**
```

```
 * return the average word length for all the chunks.
 *
 * @return double
 */
public double getAverageWordsLength();

/**
 * return the variance of all the words in all
 * the chunks.
 *
 * @return double
 */
public double getWordsVariance();

/**
 * return the degree of morphemic freedom for all
 * the single words.
 *
 * @return double
 */
public double getSingleWordsMorphemicFreedom();

/**
 * return the length of the chunk(the number of the word)
 *
 * @return int
 */
public int getLength();
}
```

7:F:\git\java\search\jcseg-1.9.5\jcseg-core\src\main\java\org\lionsoul\jcseg\core\ILastRule.java

```
package org.lionsoul.jcseg.core;

/**
 * JCSeg rule.
 * after the filter of the four rule,
 * if there is still more than one rule, JSCSegRule will
 * return the specified chunk.
 *
 * @authorchenxin
 */
public interface ILastRule {
```

```
/**
 * filter the chunks
 *
 * @return IChunk.
 */
public IChunk call(IChunk[] chunks);
}

8:F:\git\java\search\jcseg-1.9.5\jcseg-core\src\main\java\org\lionsoul\jcseg\core\ILexicon.java
package org.lionsoul.jcseg.core;

/**
 * lexicon configuration class.
 *
 * @authorchenxin
 */
public interface ILexicon {

public static final int T_LEN = 12;

/**
 * China,JPanese,Korean words
 */
public static final int CJK_WORD = 0;

/**
 * chinese single units
 */
public static final int CJK_UNITS = 1;

/**
 * chinese and english mix word.
 * like B,SIM.
 */
public static final int EC_MIXED_WORD = 2;

/**
 * chinese last name.
 */
public static final int CN_LNAME = 3;

/**
```

```
 * chinese single name.
 */
public static final int CN_SNAME = 4;

/**
 * fisrt word of chinese double name.
 */
public static final int CN_DNAME_1 = 5;

/**
 * sencond word of chinese double name.
 */
public static final int CN_DNAME_2 = 6;

/**
 * the adorn() char before the last name.
 */
public static final int CN_LNAME_ADORN = 7;
public static final int EN_PUN_WORD = 8;
public static final int STOP_WORD = 9;
public static final int CE_MIXED_WORD = 10;
public static final int EN_WORD = 11;

/**
 * unmatched word
 */
public static final int UNMATCH_CJK_WORD = 15;
}
```

9:F:\git\java\search\jcseg-1.9.5\jcseg-core\src\main\java\org\lionsoul\jcseg\core\IRule.java

```
package org.lionsoul.jcseg.core;

/**
 * filter rule interface.
 * the most important concept for mmseg chinese
 * segment algorithm.
 *
 * @authorchenxin
 */
public interface IRule {
/**
 * do the filter work
```

```
 *
 * @param chunks
 * @return IChunk[]
 */
public IChunk[] call( IChunk[] chunks );
}

10:F:\git\java\search\jcseg-1.9.5\jcseg-core\src\main\java\org\lionsoul\jcseg\core\ISegment.java
package org.lionsoul.jcseg.core;

import java.io.IOException;
import java.io.Reader;

/**
 * Jcseg segment interface
 *
 * @authorchenxin
 */
public interface ISegment
{
//Wether to check the chinese and english mixed word.
public static final int CHECK_CE_MASk = 1 << 0;
//Wether to check the chinese fraction.
public static final int CHECK_CF_MASK = 1 << 1;
//Wether to start the latin secondary segmentation.
public static final int START_SS_MASK = 1 << 2;

/**
 * reset the reader
 *
 * @param input
 */
public void reset( Reader input ) throws IOException;

/**
 * get the current length of the stream
 *
 * @return int
 */
public int getStreamPosition();

/**
```

```
 * segment a word from a char array
 * from a specified position.
 *
 * @return IWord
 */
public IWord next() throws IOException;
}

11:F:\git\java\search\jcseg-1.9.5\jcseg-core\src\main\java\org\lionsoul\jcseg\core\IWord.java
package org.lionsoul.jcseg.core;

/**
 * Word interface
 *
 * @authorchenxin
 */
public interface IWord extends Cloneable {

public static final String[] NAME_POSPEECH = {"nr"};
public static final String[] NUMERIC_POSPEECH = {"m"};
public static final String[] EN_POSPEECH = {"en"};
public static final String[] MIX_POSPEECH = {"mix"};
public static final String[] PPT_POSPEECH = {"nz"};
public static final String[] PUNCTUATION = {"w"};
public static final String[] UNRECOGNIZE = {"urg"};

/**
 * China,JPanese,Korean words
 */
public static final int T_CJK_WORD = 1;

/**
 * chinese and english mix word.
 * like B,SIM.
 */
public static final int T_MIXED_WORD = 2;

/**
 * chinese last name.
 */
public static final int T_CN_NAME = 3;
```

```
/**
 * chinese nickname.
 * like:
 */
public static final int T_CN_NICKNAME = 4;

/**
 * latain series.
 * including the arabic numbers.
 */
public static final int T_BASIC_LATIN = 5;

/**
 * letter number like ''
 */
public static final int T_LETTER_NUMBER = 6;

/**
 * other number like ''
 */
public static final int T_OTHER_NUMBER = 7;

/**
 * pinyin
 */
public static final int T_CJK_PINYIN = 8;

/**
 * Chinese numeric */
public static final int T_CN_NUMERIC = 9;

public static final int T_PUNCTUATION = 10;

/**
 * useless chars like the CJK punctuation
 */
public static final int T_UNRECOGNIZE_WORD = 11;


/**
 * return the value of the word
 *
```

```
     * @return String
     */
    String getValue();

    /**
     * return the length of the word
     *
     * @return int
     */
    int getLength();

    /**
     * return the frequency of the word,
     * use only when the word's length is one.
     *
     * @return int
     */
    int getFrequency();

    /**
     * return the type of the word
     *
     * @return int
     */
    int getType();

    /**
     * set the position of the word
     *
     * @param pos
     */
    void setPosition( int pos );

    /**
     * return the start position of the word.
     *
     * @return int
     */
    int getPosition();

    /**
     * return the pinying of the word
```

```
 */
public String getPinyin();

/**
 * return the syn words of the word.
 *
 * @return String[]
 */
public String[] getSyn();

public void setSyn( String[] syn );

/**
 * return the part of speech of the word.
 *
 * @return String[]
 */
public String[] getPartSpeech();

public void setPartSpeech( String[] ps );

/**
 * set the pinying of the word
 *
 * @param py
 */
public void setPinyin( String py );

/**
 * add a new part to speech to the word.
 *
 * @param ps
 */
public void addPartSpeech( String ps );

/**
 * add a new syn word to the word.
 *
 * @param s
 */
public void addSyn( String s );
```

```
/**
 * I mean: you have to rewrite the equals method
 * cause the jcseg require it
 */
@Override
public boolean equals( Object o );

/**
 * make clone available
 */
public IWord clone();
}
```

12:F:\git\java\search\jcseg-1.9.5\jcseg-core\src\main\java\org\lionsoul\jcseg\core\JcsegException.java

```
package org.lionsoul.jcseg.core;

/**
 * JCSeg exception class
 *
 * @authorchenxin
 */
public class JcsegException extends Exception {

private static final long serialVersionUID = 4495714680349884838L;

public JcsegException( String info ) {
super(info);
}

public JcsegException( Throwable res ) {
super(res);
}

public JcsegException( String info, Throwable res ) {
super(info, res);
}

}
```

13:F:\git\java\search\jcseg-1.9.5\jcseg-core\src\main\java\org\lionsoul\jcseg\core\JcsegTaskConfig.java

```
package org.lionsoul.jcseg.core;

import java.io.BufferedInputStream;
import java.io.File;
import java.io.FileReader;
import java.io.IOException;
import java.io.InputStream;
import java.util.Properties;

import org.lionsoul.jcseg.util.Util;


/**
 * Jcseg segmentation task config class .
 *
 *
 */
public class JcsegTaskConfig
{

/**jar home directory.*/
public static String JAR_HOME = null;
/**default lexicon property file name*/
public static final String LEX_PROPERTY_FILE = "jcseg.properties";
/**simple algorithm or complex algorithm */
public static final int SIMPLE_MODE = 1;
public static final int COMPLEX_MODE = 2;
public static final int DETECT_MODE= 3;

/**maximum length for maximum match(5-7)*/
public int MAX_LENGTH = 5;

/**
 * maximum length for the chinese words after the LATIN word.
 * use to match chinese and english mix word, like 'B,AA...'
 */
public int MIX_CN_LENGTH = 2;

/**identify the chinese name? */
public boolean I_CN_NAME = false;

/**the max length for the adron of the chinese last name.like  “”*/
```

```
public int MAX_CN_LNADRON = 1;

/**wether to load the pinying of the CJK_WORDS*/
public boolean LOAD_CJK_PINYIN = false;

/**append the pinying to the splited IWord*/
public boolean APPEND_CJK_PINYIN = false;

/**append the part of speech.*/
public boolean APPEND_PART_OF_SPEECH = false;

/**wether to load the syn word of the CJK_WORDS.*/
public boolean LOAD_CJK_SYN = false;

/**append the syn word to the splited IWord.*/
public boolean APPEND_CJK_SYN = true;

/**wether to load the word's part of speech*/
public boolean LOAD_CJK_POS = false;

/**
 * the threshold of the single word that is a single word
 * when it and the last char of the name make up a word.
 */
public int NAME_SINGLE_THRESHOLD = 1000000;

/**the maxinum length for the text bettween the pair punctution.*/
public int PPT_MAX_LENGTH = 15;

/**clear away the stopword.*/
public boolean CLEAR_STOPWORD = false;

/**chinese numeric to Arabic .*/
public boolean CNNUM_TO_ARABIC = true;

/**chinese fraction to arabic fraction .*/
public boolean CNFRA_TO_ARABIC = true;

/*Wether to do the secondary split for complex latin compose*/
public boolean EN_SECOND_SEG = true;
/*Less length for the second split to make up a word*/
public int STOKEN_MIN_LEN = 1;
```

```
/*keep puncutations*/
private String KEEP_PUNCTUATIONS = "@%&.'#+";

public boolean KEEP_UNREG_WORDS = false;

private String prefix = "lex";
private String suffix = "lex";
private String[] lexPath = null;/*lexicon direcotry path array.*/
private boolean lexAutoload = false;
private int polltime = 10;

//the currently used lexicon properties file
private String pfile = null;

public JcsegTaskConfig()
{
this(null);
}

public JcsegTaskConfig( String proFile )
{
JAR_HOME = Util.getJarHome(this);
try {
resetFromPropertyFile(proFile);
} catch (IOException e) {
e.printStackTrace();
}
}

/**
 * reset the value of its options from a propertie file .
 *
 * @paramproFilepath of jcseg.properties file.
 * when null is givend, jcseg will look up the
 * default jcseg.properties file.
 *
 * @throws IOException
 */
public void resetFromPropertyFile( String proFile ) throws IOException
{
Properties lexPro = new Properties();
```

```
/*load the mapping from the default property file.*/
if ( proFile == null )
{
/*
 * 1.load the the jcseg.properties located with the jar file.
 * 2.load the jcseg.propertiess from the classpath.
 * 3.load the jcseg.properties from the user.home
 */
boolean jcseg_properties = false;
File pro_file = new File(JAR_HOME+"/"+LEX_PROPERTY_FILE);
if ( pro_file.exists() )
{
lexPro.load(new FileReader(pro_file));
pfile = JAR_HOME+"/"+LEX_PROPERTY_FILE;
jcseg_properties = true;
}

if ( ! jcseg_properties )
{
InputStream is = DictionaryFactory.class.getResourceAsStream("/"+LEX_PROPERTY_FILE);
if ( is != null ) {
lexPro.load(new BufferedInputStream( is ));
pfile = "classpath/jcseg.properties";
jcseg_properties = true;
}
}

if ( ! jcseg_properties )
{
pro_file = new File(System.getProperty("user.home")+"/"+LEX_PROPERTY_FILE);
if ( pro_file.exists() ) {
lexPro.load(new FileReader(pro_file));
pfile = pro_file.getAbsolutePath();
jcseg_properties = true;
}
}

/*
 * jcseg properties file loading status report,
 * show the crorrent properties file location information .
 *
 * @date2013-07-06
```

```
 */
if ( ! jcseg_properties )
{
String _report = "jcseg properties[jcseg.properties] file loading error: \n";
_report += "try the follwing ways to solve the problem: \n";
_report += "1. put jcseg.properties into the classpath.\n";
_report += "2. put jcseg.properties together with the jcseg-core-{version}.jar file.\n";
_report += "3. put jcseg.properties in directory "+System.getProperty("user.home")+"\n\n";
throw new IOException(_report);
}
}
/*load the  mapping from the specified property file.*/
else
{
File pro_file = new File(proFile);
if ( ! pro_file.exists() )
throw new IOException("property file ["+proFile+"] not found!");
lexPro.load(new FileReader(pro_file));
}

/*about the lexicon*/
//the lexicon path
String lexDirs = lexPro.getProperty("lexicon.path");
if ( lexDirs == null )
throw new IOException("lexicon.path property not find in jcseg.properties file!!!");
if ( lexDirs.indexOf("{jar.dir}") > -1 )
lexDirs = lexDirs.replace("{jar.dir}", JAR_HOME);
//System.out.println("path: "+lexPath);

//Multiple path for lexicon.path.
lexPath = lexDirs.split(";");
File f = null;
for ( int i = 0; i < lexPath.length; i++ )
{
lexPath[i] = java.net.URLDecoder.decode(lexPath[i], "UTF-8");
f = new File(lexPath[i]);
if ( ! f.exists() )
throw new IOException("Invalid sub lexicon path " + lexPath[i]
+ " for lexicon.path in jcseg.properties");
f = null;//Let gc do its work.
}
```

```
//the lexicon file prefix and suffix
if ( lexPro.getProperty("lexicon.suffix") != null )
suffix = lexPro.getProperty("lexicon.suffix");
if ( lexPro.getProperty("lexicon.prefix") != null )
prefix = lexPro.getProperty("lexicon.prefix");

//reset all the options
if ( lexPro.getProperty("jcseg.maxlen") != null )
MAX_LENGTH = Integer.parseInt(lexPro.getProperty("jcseg.maxlen"));
if ( lexPro.getProperty("jcseg.mixcnlen") != null )
MIX_CN_LENGTH = Integer.parseInt(lexPro.getProperty("jcseg.mixcnlen"));
if ( lexPro.getProperty("jcseg.icnname") != null
&& lexPro.getProperty("jcseg.icnname").equals("1"))
I_CN_NAME = true;
if ( lexPro.getProperty("jcseg.cnmaxlnadron") != null )
MAX_CN_LNADRON = Integer.parseInt(lexPro.getProperty("jcseg.cnmaxlnadron"));
if ( lexPro.getProperty("jcseg.nsthreshold") != null )
NAME_SINGLE_THRESHOLD = Integer.parseInt(lexPro.getProperty("jcseg.nsthreshold"));
if ( lexPro.getProperty("jcseg.pptmaxlen") != null )
PPT_MAX_LENGTH = Integer.parseInt(lexPro.getProperty("jcseg.pptmaxlen"));
if ( lexPro.getProperty("jcseg.loadpinyin") != null
&& lexPro.getProperty("jcseg.loadpinyin").equals("1"))
LOAD_CJK_PINYIN = true;
if ( lexPro.getProperty("jcseg.loadsyn") != null
&& lexPro.getProperty("jcseg.loadsyn").equals("1") )
LOAD_CJK_SYN = true;
if ( lexPro.getProperty("jcseg.loadpos") != null
&& lexPro.getProperty("jcseg.loadpos").equals("1"))
LOAD_CJK_POS = true;
if ( lexPro.getProperty("jcseg.clearstopword") != null
&& lexPro.getProperty("jcseg.clearstopword").equals("1"))
CLEAR_STOPWORD = true;
if ( lexPro.getProperty("jcseg.cnnumtoarabic") != null
&& lexPro.getProperty("jcseg.cnnumtoarabic").equals("0"))
CNNUM_TO_ARABIC = false;
if ( lexPro.getProperty("jcseg.cnfratoarabic") != null
&& lexPro.getProperty("jcseg.cnfratoarabic").equals("0"))
CNFRA_TO_ARABIC = false;
if ( lexPro.getProperty("jcseg.keepunregword") != null
&& lexPro.getProperty("jcseg.keepunregword").equals("1"))
KEEP_UNREG_WORDS = true;
if ( lexPro.getProperty("lexicon.autoload") != null
```

```
&& lexPro.getProperty("lexicon.autoload").equals("1"))
lexAutoload = true;
if ( lexPro.getProperty("lexicon.polltime") != null )
polltime = Integer.parseInt(lexPro.getProperty("lexicon.polltime"));

//secondary split
if ( lexPro.getProperty("jcseg.ensencondseg") != null
&& lexPro.getProperty("jcseg.ensencondseg").equals("0"))
EN_SECOND_SEG = false;
if ( lexPro.getProperty("jcseg.stokenminlen") != null )
STOKEN_MIN_LEN = Integer.parseInt(lexPro.getProperty("jcseg.stokenminlen"));

//load the keep punctuations.
if ( lexPro.getProperty("jcseg.keeppunctuations") != null )
KEEP_PUNCTUATIONS = lexPro.getProperty("jcseg.keeppunctuations");
}

/**property about lexicon file.*/
public String getLexiconFilePrefix() {
return prefix;
}

public String getLexiconFileSuffix() {
return suffix;
}
/**return the lexicon directory path*/
public String[] getLexiconPath() {
return lexPath;
}

public void setLexiconPath( String[] lexPath ) {
this.lexPath = lexPath;
}

/**about lexicon autoload*/
public boolean isAutoload() {
return lexAutoload;
}

public void setAutoload( boolean autoload ) {
lexAutoload = autoload;
}
```

```
public int getPollTime() {
return polltime;
}

public void setPollTime( int polltime ) {
this.polltime = polltime;
}

public int getMaxLength() {
return MAX_LENGTH;
}

public void setMaxLength( int maxLength ) {
MAX_LENGTH = maxLength;
}

public int getMixCnLength() {
return MIX_CN_LENGTH;
}

public void setMixCnLength( int mixCnLength ) {
MIX_CN_LENGTH = mixCnLength;
}

public boolean identifyCnName() {
return I_CN_NAME;
}

public void setICnName( boolean iCnName ) {
I_CN_NAME = iCnName;
}

public int getMaxCnLnadron() {
return MAX_CN_LNADRON;
}

public void setMaxCnLnadron( int maxCnLnadron ) {
MAX_CN_LNADRON = maxCnLnadron;
}

public boolean loadCJKPinyin() {
```

```
return LOAD_CJK_PINYIN;
}

public void setLoadCJKPinyin( boolean loadCJKPinyin ) {
LOAD_CJK_PINYIN = loadCJKPinyin;
}

public void setAppendPartOfSpeech( boolean partOfSpeech ) {
APPEND_PART_OF_SPEECH = partOfSpeech;
}

public boolean appendCJKPinyin() {
return APPEND_CJK_PINYIN;
}

public void setAppendCJKPinyin( boolean appendCJKPinyin ) {
APPEND_CJK_PINYIN = appendCJKPinyin;
}

public boolean loadCJKSyn() {
return LOAD_CJK_SYN;
}

public void setLoadCJKSyn( boolean loadCJKSyn ) {
LOAD_CJK_SYN = loadCJKSyn;
}

public boolean appendCJKSyn() {
return APPEND_CJK_SYN;
}

public void setAppendCJKSyn( boolean appendCJKPinyin ) {
APPEND_CJK_SYN = appendCJKPinyin;
}

public boolean ladCJKPos() {
return LOAD_CJK_POS;
}

public void setLoadCJKPos( boolean loadCJKPos ) {
LOAD_CJK_POS = loadCJKPos;
}
```

```
public int getNameSingleThreshold() {
return NAME_SINGLE_THRESHOLD;
}

public void setNameSingleThreshold( int thresold ) {
NAME_SINGLE_THRESHOLD = thresold;
}

public int getPPTMaxLength() {
return PPT_MAX_LENGTH;
}

public void setPPT_MAX_LENGTH( int pptMaxLength ) {
PPT_MAX_LENGTH = pptMaxLength;
}

public boolean clearStopwords() {
return CLEAR_STOPWORD;
}

public void setClearStopwords( boolean clearstopwords ) {
CLEAR_STOPWORD = clearstopwords;
}

public boolean cnNumToArabic() {
return CNNUM_TO_ARABIC;
}

public void setCnNumToArabic( boolean cnNumToArabic ) {
CNNUM_TO_ARABIC = cnNumToArabic;
}

public boolean cnFractionToArabic() {
return CNFRA_TO_ARABIC;
}

public void setCnFactionToArabic( boolean cnFractionToArabic ) {
CNFRA_TO_ARABIC = cnFractionToArabic;
}

public boolean getEnSecondSeg() {
```

```
return EN_SECOND_SEG;
}

public void setEnSecondSeg( boolean enSecondSeg ) {
this.EN_SECOND_SEG = enSecondSeg;
}

public int getSTokenMinLen() {
return STOKEN_MIN_LEN;
}

public void setSTokenMinLen( int len ) {
STOKEN_MIN_LEN = len;
}

public void setKeepPunctuations( String keepPunctuations ) {
KEEP_PUNCTUATIONS = keepPunctuations;
}

public boolean isKeepPunctuation( char c ) {
return (KEEP_PUNCTUATIONS.indexOf(c) > -1);
}

public boolean keepUnregWords() {
return KEEP_UNREG_WORDS;
}

public void setKeepUnregWords( boolean keepUnregWords ) {
KEEP_UNREG_WORDS = keepUnregWords;
}

//return the currently use properties file
public String getPropertieFile()
{
return pfile;
}
}
```

14:F:\git\java\search\jcseg-1.9.5\jcseg-core\src\main\java\org\lionsoul\jcseg\core\JHashMap.java

```
package org.lionsoul.jcseg.core;
```

```
public class JHashMap {

/**
 * default size for hash table
 */
public static final int DEFAULT_TABLE_SIZE = 31;

/**
 * the default filling factor
 */
public static final float DEFAULT_FILL_FACTOR = 0.75f;

/**
 * size of the current hash table
 */
private int size;

/**
 * current filling factor
 */
private float fillFactor = DEFAULT_FILL_FACTOR;

/**
 * The next size value at which to resize (capacity * load factor).
 */
transient int threshold;

/**
 * hash table block
 */
public Entry [] table = null;


public JHashMap() {
this( DEFAULT_TABLE_SIZE );
}

public JHashMap( int _size ) {
this( _size, DEFAULT_FILL_FACTOR );
}

public JHashMap( int _size, float factor ) {
```

```
fillFactor = factor;
int opacity = nextPrime( _size );
table = new Entry[opacity];
size = 0;
threshold = (int)( opacity * fillFactor );
}

/**
 * check the hash table is empty or not
 * @return true for empty and false for not
 */
public boolean isEmpty() {
return size == 0;
}

/**
 * make the whole table empty
 */
public void makeEmpty() {
for ( int j = 0; j < table.length; j++ )
table[ j ] = null;
size = 0;
}

/**
 * find an item in a hash table
 * @param key
 * @return true for found and false for not
 */
public boolean containsKey( String key ) {
return getEntry( key ) != null;
}

/**
 * find a value in a hash table
 * @param val
 * @return true for found and false for not
 */
public boolean containsValue( IWord val ) {
Entry e = null;
for ( int i = 0; i < table.length; i++ ) {
for ( e = table[i];
```

```
e != null;
e = e._next) {
if ( val.equals( e.val ) || val == e.val )
return true;
}
}
return false;
}

/**
 * get the value associate withe the specified key
 * @param key
 * @return if there is a mapping for the key return
 * the value of the Entry
 * or return null
 */
public IWord get( String key ) {
Entry e = getEntry( key );
if ( e != null )
return e.val;
return null;
}

/**
    * Returns the entry associated with the specified key in the
    * hash table.
    * @returns null if the HashMap contains no mapping for the key.
    * else the associated Entry
    */
   final Entry getEntry( String key ) {
       int hash = (key == null) ? 0 : hash( key );
       String k;
       for ( Entry e = table[hash];
            e != null;
            e = e._next ) {
           if ( e.hash == hash &&
               ((k = e.key) == key || (key != null && key.equals(k))))
               return e;
       }
       return null;
   }
```

```
/**
 * Associates the specified value with the specified key in the table
 * if the map contains the mapping for the key, the old value is replaced
 * @param key
 * @param val
 * @return the oldValue if the same key is exists or the item just add
 */
public IWord put( String key,  IWord val ) {
int hash = (key == null) ? 0 : hash( key );
String k;
for ( Entry e = table[hash];
e != null;
e = e._next ) {
if ( e.hash == hash &&
((k = e.key) == key || ( key != null && key.equals(k)))) {
IWord oVal = e.getValue();
e.val = val;
return oVal;
}
}

addEntry( key, val, hash );
return null;
}

/**
 * Add a new entry with the specified key, value and hash code to
     * the specified bucket.
     * It is the responsibility of this
     * method to resize the table if appropriate.
 */
void addEntry( String key, IWord val, int hash ) {
Entry e = table[hash];
table[hash] = new Entry( key, val, hash, e );
if ( size++ >= threshold )
reHash();
/*
 * Attention:
 * do not use the following condition check
 * it will cause memory use up error.
 * here use threshold instead
 */
```

```
//if ( table.length * fillFactor >= size++ ) {
//reHash();
//}
}


/**
 * remove a item from a hash table
 * @param key the item to remove
 * @return if there is a mapping for the key
 * return the old value
 * else return null
 */
public IWord remove( String key ) {
int hash = ( key == null ) ? 0 : hash( key );
String k;
Entry eb = null;
for ( Entry e = table[hash];
e != null;
e = e._next ) {
Entry next = e._next;
/*
 * the first Entry of the LinkedList
 */
if ( eb == null ) {
table[hash] = next;
if ( next == null ) {
IWord eVal = e.val;
e = null;
return eVal;
}
eb = e;
continue;
}
else if ( e.hash == hash &&
((k=e.key) == key || (key != null && key.equals(k)))) {
eb._next = e._next;
IWord eVal = e.val;
e = null;
size--;
return eVal;
}
```

```
eb = eb._next;
}
return null;
}

/**
 * if table.length times fillFactor is larger than
 * the size, we need to reload the hash table
 */
private void reHash() {
Entry[] _src = table;
//create a new double-sized table
//then copy the table
//nextPrime( 2 * table.length )
int opacity = nextPrime(2 * table.length);
table = new Entry[opacity];
Entry e;
for ( int j = 0; j < _src.length; j++ ) {
e = _src[j];
if ( e != null ) {
_src[j] = null;
do {
Entry next = e._next;
int hash = hash( e.key );
e.hash = hash;
e._next = table[hash];
table[hash] = e;
e = next;
} while ( e != null );
}
}
threshold = (int)(opacity * fillFactor);
}

/**
 * Entry class
 */
public static class Entry {

String key;
IWord val;
int hash;
```

```
Entry _next;

public Entry( String k, IWord v, int h, Entry next ) {
key = k;
val = v;
hash = h;
_next = next;
}

public String getKey() {
return key;
}

public IWord getValue() {
return val;
}
}

/**
 * a hash routine for String Object
 * @param key
 * @return the hashcode
 */
public int hash( String key ) {

int factor = 131;
int hashVal = 0;

for ( int j = 0; j < key.length(); j++ )
hashVal = hashVal * factor + key.charAt(j);

hashVal = hashVal % size;
if ( hashVal < 0 )
hashVal = hashVal + size;

return (hashVal & 0x7FFFFFFF);
}

/**
 * internal method to general prime number after the given number
 * @param n the base number
 * @return the next prime number
```

```
 */
private static int nextPrime( int n ) {

//make sure n is an odd number
if ( n % 2 == 0 )
n++;

for ( ; ! isPrime( n ); n = n + 2 ) ;

return n;
}

/**
 * internal method to test a given number is a prime or not
 * @param n the number to test
 * @return the result of the test
 */
private static boolean isPrime( int n ) {

if ( n == 2 || n == 3 )
return true;

if ( n == 1 || n % 2 == 0 )
return false;

for ( int j = 3; j * j < n; j++)
if ( n % j == 0 )
return false;

return true;
}

}
```

15:F:\git\java\search\jcseg-1.9.5\jcseg-core\src\main\java\org\lionsoul\jcseg\core\LexiconException.java

```
package org.lionsoul.jcseg.core;


public class LexiconException extends Exception {

private static final long serialVersionUID = 3794928123652720865L;
```

```
public LexiconException( String info ) {
super(info);
}

public LexiconException( Throwable res ) {
super(res);
}

public LexiconException( String info, Throwable res ) {
super(info, res);
}

}

16:F:\git\java\search\jcseg-1.9.5\jcseg-
core\src\main\java\org\lionsoul\jcseg\core\SegmentFactory.java
package org.lionsoul.jcseg.core;

import java.io.Reader;
import java.lang.reflect.Constructor;


public class SegmentFactory
{

//current jcseg version.
public static final String version = "1.9.4";

/**
 * load the ISegment class with the given path
 *
 * @param __segClass
 * @return ISegment
 */
public static ISegment createSegment( String __segClass,
Class<?> paramtypes[], Object args[] )
{
ISegment seg = null;
try {
Class<?> _class = Class.forName(__segClass);
Constructor<?> cons = _class.getConstructor(paramtypes);
```

```
seg = ( ISegment ) cons.newInstance(args);
} catch (Exception e) {
e.printStackTrace();
System.out.println("can't load the ISegment implements class " +
"with path ["+__segClass+"] ");
}

return seg;
}

/**
 * create the specified mode jcseg instance .
 *
 * @parammode
 * @returnISegment
 * @throws JcsegException
 */
public static ISegment createJcseg( int mode, Object...args ) throws JcsegException
{
String __segClass;
if ( mode == JcsegTaskConfig.SIMPLE_MODE )
__segClass = "org.lionsoul.jcseg.SimpleSeg";
else if ( mode == JcsegTaskConfig.COMPLEX_MODE )
__segClass = "org.lionsoul.jcseg.ComplexSeg";
else if ( mode == JcsegTaskConfig.DETECT_MODE )
__segClass = "org.lionsoul.jcseg.DetectSeg";
else
throw new JcsegException("No Such Algorithm Excpetion");

Class<?>[] _paramtype = null;
if ( args.length == 2 ) {
_paramtype = new Class[]{JcsegTaskConfig.class, ADictionary.class};
} else if ( args.length == 3 ) {
_paramtype = new Class[]{Reader.class, JcsegTaskConfig.class, ADictionary.class};
} else {
throw new JcsegException("length of the arguments should be 2 or 3");
}

return createSegment(__segClass, _paramtype, args);
}

}
```

17:F:\git\java\search\jcseg-1.9.5\jcseg-core\src\main\java\org\lionsoul\jcseg\DetectSeg.java
```
package org.lionsoul.jcseg;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.Reader;

import org.lionsoul.jcseg.core.ILexicon;
import org.lionsoul.jcseg.core.ISegment;
import org.lionsoul.jcseg.core.ADictionary;
import org.lionsoul.jcseg.core.IWord;
import org.lionsoul.jcseg.core.JcsegTaskConfig;
import org.lionsoul.jcseg.filter.ENSCFilter;
import org.lionsoul.jcseg.util.IPushbackReader;
import org.lionsoul.jcseg.util.IStringBuffer;

/**
 * Detect Segmentation mode
 * return words only in the loaded dictionary
 *
 * yat, when matched a word and return it
 * or continue to find the next word in the dictionary
 *
 * @author chenxin
 * @since1.9.4
 */

public class DetectSeg implements ISegment
{
/*current position for the given stream.*/
private int idx;

//protected PushbackReader reader = null;
private IPushbackReader reader = null;
privateIStringBufferisb= null;

/*the dictionary and task config*/
private ADictionary dic;
private JcsegTaskConfig config;

/**
```

```
 * method to create the new ISegment
 *
 * @paramconfig
 * @paramdic
 * @throwsIOException
 */
public DetectSeg(JcsegTaskConfig config, ADictionary dic)
throws IOException
{
this(null, config, dic);
}

/**
 * method to create a new ISegment
 *
 * @paraminput
 * @paramconfig
 * @paramdic
 * @throwsIOException
 */
public DetectSeg(Reader input, JcsegTaskConfig config, ADictionary dic)
throws IOException
{
this.config= config;
this.dic= dic;

isb= new IStringBuffer(64);
reset(input);//reset the stream
}

/**
 * @seeISegment#reset(Reader)
 */
@Override
public void reset(Reader input) throws IOException
{
if ( input != null )
reader = new IPushbackReader(new BufferedReader(input));
idx = -1;
}

/**
```

```
 * @seeISegment#getStreamPosition()
 */
@Override
public int getStreamPosition()
{
return idx + 1;
}

/**
 * read the next char from the current position
 *
 * @returnint
 * @throws IOException
 */
protected int readNext()
throws IOException
{
int c = reader.read();
if ( c != -1 ) idx++;
return c;
}

/**
 * push back the data to the stream.
 *
 * @param data
 * @throwsIOException
 */
protected void pushBack( int data )
throws IOException
{
reader.unread(data);
idx--;
}

/**
 * set the dictionary of the current segmentor.
 *
 * @paramdic
 */
public void setDict( ADictionary dic )
{
```

```
this.dic = dic;
}

/**
 * get the current dictionary instance .
 *
 * @returnADictionary
 */
public ADictionary getDict()
{
return dic;
}

/**
 * set the current task config .
 *
 * @paramconfig
 */
public void setConfig( JcsegTaskConfig config )
{
this.config = config;
}

/**
 * get the current task config instance.
 *
 * @paramJcsegTaskConfig
 */
public JcsegTaskConfig getConfig()
{
return config;
}

/**
 * @seeISegment#next()
 *
 * @returnIWord or null
 */
@Override
public IWord next() throws IOException
{
int c, i;
```

```
IWordw = null;
StringT = null;

while ( (c = readNext()) != -1 )
{
w= null;
T= null;
isb.clear();

//@Convertor: check if char is an latin letter
//and make the full-width half-width uppercase lowercase
//if it does
if ( ENSCFilter.isHWEnChar(c) || ENSCFilter.isFWEnChar(c) )
{
if ( c > 65280 ) c -= 65248;
if ( c >= 65 && c <= 90 ) c += 32;
}
isb.append((char)c);

//get the temp string
//and check T is a valid word in dictionary
T= isb.toString();
if ( dic.match(ILexicon.CJK_WORD, T) )
{
w = dic.get(ILexicon.CJK_WORD, T);
}

//forward maximum matching loop
for ( i = 1; i < config.MAX_LENGTH; i++ )
{
c= readNext();
if ( c == -1 ) break;

//@see @Convertor
if ( ENSCFilter.isHWEnChar(c) || ENSCFilter.isFWEnChar(c) )
{
if ( c > 65280 ) c -= 65248;
if ( c >= 65 && c <= 90 ) c += 32;
}
isb.append((char)c);

//get the temp string
```

```
T= isb.toString();
//System.out.println(T);

//check T is a valid word in dictionary
if ( dic.match(ILexicon.CJK_WORD, T) )
{
w = dic.get(ILexicon.CJK_WORD, T);
}
}

/*
 * match no word in dictionary
 * push back the char readed except the first one and continue the loop
 */
if ( w == null )
{
for ( i = isb.length() - 1; i > 0; i-- ) pushBack(isb.charAt(i));
continue;
}

//-------------------------------------------------
//yat, match a item and return it as a segment result
//also we need to push back the none-match part
//@Note: we will not check the pinyin, part of speech, synonyms words
//get the need? do it yourself here. @see ASegment#next()
int LENGTH= w.getLength();
for ( i = isb.length() - 1; i >= LENGTH; i-- )
{
pushBack(isb.charAt(i));
}

return w;
}

return null;
}
}
```

18:F:\git\java\search\jcseg-1.9.5\jcseg-core\src\main\java\org\lionsoul\jcseg\Dictionary.java

```
package org.lionsoul.jcseg;

import java.util.HashMap;
```

```
//import java.util.Hashtable;
import java.util.Map;
import java.util.concurrent.ConcurrentHashMap;

import org.lionsoul.jcseg.core.ADictionary;
import org.lionsoul.jcseg.core.ILexicon;
import org.lionsoul.jcseg.core.IWord;
import org.lionsoul.jcseg.core.JcsegTaskConfig;

//import com.webssky.jcseg.core.JHashMap;

/**
 * Dictionary class.
 *
 * @authorchenxin
 */
public class Dictionary extends ADictionary {

/**hash table for the words*/
private Map<String, IWord>[] dics = null;

@SuppressWarnings("unchecked")
public Dictionary( JcsegTaskConfig config, Boolean sync ) {
super(config, sync);
dics = new Map[ILexicon.T_LEN];
if ( this.sync ) {
for ( int j = 0; j < ILexicon.T_LEN; j++ )
dics[j] = new ConcurrentHashMap<String, IWord>(16, 0.80F);
} else {
for ( int j = 0; j < ILexicon.T_LEN; j++ )
dics[j] = new HashMap<String, IWord>(16, 0.80F);
}
}

/**
 * @see ADictionary#match(int, String)
 */
@Override
public boolean match(int t, String key) {
if ( t < 0 || t >= ILexicon.T_LEN ) return false;
return dics[t].containsKey(key);
}
```

```
/**
 * @see ADictionary#add(int, String, int)
 */
@Override
public void add(int t, String key, int type) {
if ( t < 0 || t >= ILexicon.T_LEN ) return;
if ( dics[t].get(key) == null )
dics[t].put(key, new Word(key, type));
}

/**
 * @see ADictionary#add(int, String, int, int)
 */
@Override
public void add(int t, String key, int fre, int type) {
if (  t < 0 || t >= ILexicon.T_LEN  ) return;
if ( dics[t].get(key) == null )
dics[t].put(key, new Word(key, fre, type));
}

/**
 * @see ADictionary#get(int, String)
 */
@Override
public IWord get(int t, String key) {
if (  t < 0 || t >= ILexicon.T_LEN  ) return null;
return dics[t].get(key);
}

/**
 * @see ADictionary#remove(int, String)
 */
@Override
public void remove(int t, String key) {
if (  t < 0 || t >= ILexicon.T_LEN  ) return;
dics[t].remove(key);
}

/**
 * @see ADictionary#size(int)
 */
```

```
@Override
public int size(int t) {
if (  t < 0 || t >= ILexicon.T_LEN  ) return 0;
return dics[t].size();
}
}
```

19:F:\git\java\search\jcseg-1.9.5\jcseg-core\src\main\java\org\lionsoul\jcseg\filter\CNNMFilter.java

```
package org.lionsoul.jcseg.filter;

import java.util.HashMap;
import java.util.Map;


public class CNNMFilter {
/**
 * chinese numeric chars.
 * i have put the chars into the lexicon file lex-cn-numeric.lex for the old version. <r />
 * it's better to follow the current work.
 */
private static final Character[] CN_NUMERIC = {
'', '', '', '', '','', '', '', '',
'', '', '', '', '', '', '', '','',
'', '', '',
'', '', '', '', '', '', '', '', /*',
'', '',*/};

private static Map<Character, Integer> cnNumeric = null;

static {
cnNumeric = new HashMap<Character, Integer>(40, 0.85f);
for ( int j = 0; j < 9; j++ )
cnNumeric.put(CN_NUMERIC[j], j + 1);
for ( int j = 9; j < 18; j++ )
cnNumeric.put(CN_NUMERIC[j], j - 8);
for ( int j = 18; j < 21; j++ )
cnNumeric.put(CN_NUMERIC[j], 0);

cnNumeric.put(' ', 0);
cnNumeric.put('', 2);
cnNumeric.put('', 10);
cnNumeric.put('', 10);
```

```
cnNumeric.put('', 100);
cnNumeric.put('', 100);
cnNumeric.put('', 1000);
cnNumeric.put('', 1000);
cnNumeric.put('', 10000);
cnNumeric.put('', 100000000);
}

/**
 * check the given char is chinese numeric or not.
 *
 * @param c
 * @return boolean true for the char is chinese numeric and false for not.
 */
public static int isCNNumeric( char c )
{
Integer i = cnNumeric.get(c);
if ( i == null ) return -1;
return i.intValue();
}


/**
 * a static method to turn the Chinese numeric to Arabic numbers.
 *
 * @param cnn
 * @param  flag
 * @return int
 */
public static int cnNumericToArabic( String cnn, boolean flag )
{
cnn = cnn.trim();
if ( cnn.length() == 1 ) return isCNNumeric(cnn.charAt(0));

if ( flag ) cnn = cnn.replace('', '')
.replace('', '').replace('', '').replace('', ' ');

int yi = -1, wan = -1, qian = -1, bai = -1, shi = -1;
int val = 0;
yi = cnn.lastIndexOf('');
if ( yi > -1 )
{
```

```
val += cnNumericToArabic( cnn.substring(0, yi), false ) * 100000000;
if ( yi < cnn.length() - 1 )
cnn = cnn.substring(yi + 1, cnn.length());
else
cnn = "";

if ( cnn.length() == 1 ) {
int arbic = isCNNumeric(cnn.charAt(0));
if ( arbic <= 10 )
val += arbic * 10000000;
cnn = "";
}
}

wan = cnn.lastIndexOf('');
if ( wan > -1 )
{
val += cnNumericToArabic( cnn.substring(0, wan), false ) * 10000;
if ( wan < cnn.length() - 1 )
cnn = cnn.substring(wan + 1, cnn.length());
else
cnn = "";
if ( cnn.length() == 1 ) {
int arbic = isCNNumeric(cnn.charAt(0));
if ( arbic <= 10 )
val += arbic * 1000;
cnn = "";
}
}

qian = cnn.lastIndexOf('');
if ( qian > -1 )
{
val +=  cnNumericToArabic( cnn.substring(0, qian), false ) * 1000;
if ( qian < cnn.length() - 1 )
cnn = cnn.substring(qian + 1, cnn.length());
else
cnn = "";
if ( cnn.length() == 1 ) {
int arbic = isCNNumeric(cnn.charAt(0));
if ( arbic <= 10 )
val += arbic * 100;
```

```
cnn = "";
}
}

bai = cnn.lastIndexOf('');
if ( bai > -1 )
{
val += cnNumericToArabic( cnn.substring(0, bai), false ) * 100;
if ( bai < cnn.length() - 1 )
cnn = cnn.substring(bai + 1, cnn.length());
else
cnn = "";
if ( cnn.length() == 1 ) {
int arbic = isCNNumeric(cnn.charAt(0));
if ( arbic <= 10 )
val += arbic * 10;
cnn = "";
}
}

shi = cnn.lastIndexOf('');
if ( shi > -1 )
{
if ( shi == 0 )
val += 1 * 10;
else
val += cnNumericToArabic( cnn.substring(0, shi), false ) * 10;
if ( shi < cnn.length() - 1 )
cnn = cnn.substring(shi + 1, cnn.length());
else
cnn = "";
}

cnn = cnn.trim();
for ( int j = 0; j < cnn.length(); j++ )
val += isCNNumeric(cnn.charAt(j))
* Math.pow(10, cnn.length() - j - 1);

return val;
}

public static int qCNNumericToArabic( String cnn ) {
```

```
int val = 0;
cnn = cnn.trim();
for ( int j = 0; j < cnn.length(); j++ )
val += isCNNumeric(cnn.charAt(j))
* Math.pow(10, cnn.length() - j - 1);
return val;
}

/*public static void main(String[] args) {
//ADictionary.isCNNumeric('');
int val = 0;
long s = System.nanoTime();
//val = cnNumericToArabic("", true);
//val = cnNumericToArabic("", true);
//val = cnNumericToArabic("", true);
//val = cnNumericToArabic("", true);
//val = cnNumericToArabic("", true);
long e = System.nanoTime();
System.out.format("Done["+val+"], cost: %.5fsec\n", ((float)(e - s)) / 1E9);
}*/
}
```

20:F:\git\java\search\jcseg-1.9.5\jcseg-core\src\main\java\org\lionsoul\jcseg\filter\ENSCFilter.java

```
package org.lionsoul.jcseg.filter;

/**
 * a class to deal with the english stop char
 * like the english punctuation.
 *
 * @authorchenxin
 */
public class ENSCFilter
{
//type constants
public static final int EN_LETTER = 0;
public static final int EN_NUMERIC = 1;
public static final int EN_PUNCTUATION = 2;
public static final int EN_WHITESPACE = 3;
public static final int EN_UNKNOW = -1;

private static final String EN_KEEP_CHARS = "@%&.'#+";
```

```
/*private static final Character[] EN_KEEP_CHARS = {
'@', '$', '%', '^', '&', '-', ':', '.', '/', '\'', '#', '+'};

private static Map<Character, Character> enKeepChar = null;

static {
enKeepChar = new HashMap<Character, Character>(
( int )(EN_KEEP_CHARS.length / 1.7) + 1, 0.85f );
//set the keep char's keep status
for ( int j = 0; j < EN_KEEP_CHARS.length; j++ )
enKeepChar.put(EN_KEEP_CHARS[j], EN_KEEP_CHARS[j]);
}*/

/**
 * check the given char is english keep punctuation.
 *
 *  c
 * @returnboolean
 */
public static boolean isENKeepPunctuaton( char c ) {
return (EN_KEEP_CHARS.indexOf(c) > -1);
//return enKeepChar.containsKey(c);
}

public static boolean isUpperCaseLetter( int u ) {
return ( u >= 65 && u <= 90 );
}

public static boolean isLowerCaseLetter( int u ) {
return ( u >= 97 && u <= 122 );
}

public static int toLowerCase( int u ) {
return ( u + 32 );
}

public static int toUpperCase( int u ) {
return ( u - 32 );
}

/**
 * include the full-width and half-width char.
```

```
 *
 *  u
 */
public static boolean isEnLetter( int u ) {
if ( u > 65280 ) u -= 65248;//make full-with half-width
return ( (u >= 65 && u <= 90) || ( u >= 97 && u <= 122 ) );
}

/**
 * check the specifield char is an english numeric(48-57)
 *  including the full-width char
 *
 *   u
 */
public static boolean isEnNumeric( int u )
{
if ( u > 65280 ) u -= 65248;//make full-with half-width
return ( (u >= 48 && u <= 57) );
}

/**
 * get the type of the english char
 * defined in this class and start
 *
 *  uchar to identity.
 * @returninttype keywords.
 */
public static int getEnCharType( int u ) {
//if ( u > 65280 ) u -= 65248;//make full-with half-width
if ( u > 126 )return EN_UNKNOW;
if ( u == 32 )return EN_WHITESPACE;
if ( u >= 48 && u <= 57 )return EN_NUMERIC;
if ( u >= 65 && u <= 90 )return EN_LETTER;
if ( u >= 97 && u <= 122 )return EN_LETTER;
return EN_PUNCTUATION;
}

/**
 * check the given char is a half-width char or not.
 *
 *   int
 * @return boolean
```

```
*/
   public static boolean isHWEnChar( int c ) {
       return (c >= 32 && c <= 126);
   }

   /**
    * check the given char is a full-width char.
    * the full-width punctuation is not included here.
    *
    *   c
    * @return boolean
    */
   public static boolean isFWEnChar( int c ) {
   return ( ( c >= 65296 && c <= 65305 )
   || ( c >= 65313 && c <= 65338 )
   || ( c >= 65345 && c < 65370 ) );
   }

   /**
    * check the given char is half-width punctuation.
    *
    *   c
    * @return boolean
    */
   public static boolean isEnPunctuation( int c ) {
   return ( (c > 32 && c < 48)
   || ( c > 57 && c < 65 )
   || ( c > 90 && c < 97 )
   || ( c > 122 && c < 127 ));
   }

   public static boolean isCnPunctuation( int c ) {
   return ( (c > 65280 && c < 65296)
   || ( c > 65305 && c < 65312 )
   || ( c > 65338 && c < 65345 )
   || ( c > 65370 && c < 65382 )
   //CJK symbol and punctuations (added 2013-09-06)
   //from http://www.unicode.org/charts/PDF/U3000.pdf
   || ( c >= 12289 && c <= 12319 ) );
   }

   /**
```

```
     * check the given string is a whitespace.
     *
     *   c
     * @return boolean;
     */
    public static boolean isWhitespace( int c ) {
    return ( c == 32 || c == 12288 );
    }

/**
 * check the specified char is a digit or not.
 * true will return if it is or return false
 * this method can recognize full-with char.
 *
 *  str
 * @returnboolean
 */
public static boolean isDigit( String str )
{
char c;
for ( int j = 0; j < str.length(); j++ )
{
c = str.charAt(j);
//make full-width char half-width
if ( c > 65280 ) c -= 65248;
if ( c < 48 || c > 57 ) return false;
}
return true;
}

/**
 * check the specified char is a decimal.
 * including the full-width char.
 *
 *  str
 * @returnboolean
 */
public static boolean isDecimal( String str )
{
if ( str.charAt(str.length() - 1) == '.'
|| str.charAt(0) == '.' ) return false;
char c;
```

```
int p= 0;//number of point
for ( int j = 1; j < str.length(); j++ )
{
c = str.charAt(j);
if ( c == '.' ) p++;
else
{
//make full-width half-width
if ( c > 65280 ) c -= 65248;
if ( c < 48 || c > 57 ) return false;
}
}

return (p==1);
}

    /**
     * a static method to replace the full-width char to the half-width char
     * in a given string.
     * (65281-65374 for full-width char)
     *
     *   str
     * @return String the new String after the replace.
     */
    public static String fwsTohws( String str ) {
    char[] chars = str.toCharArray();
    for ( int j = 0; j < chars.length; j++ ) {
    if ( chars[j] == '\u3000' )
    chars[j] = '\u0020';
    else if ( chars[j] > '\uFF00' && chars[j] < '\uFF5F' )
    chars[j] = ( char ) (chars[j] - 65248);
    }
    return new String(chars);
    }

    /**
     * a static method to replace the half-width char to the full-width char.
     * in a given string.
     *
     *   str
     * @return String the new String after the replace.
     */
```

```
    public static String hwsTofws( String str ) {
    char[] chars = str.toCharArray();
    for ( int j = 0; j < chars.length; j++ ) {
    if ( chars[j] == '\u0020' )
    chars[j] = '\u3000';
    else if ( chars[j] < '\177' )
    chars[j] = ( char ) ( chars[j] + 65248 );
    }
    return new String(chars);
    }
}
```

21:F:\git\java\search\jcseg-1.9.5\jcseg-core\src\main\java\org\lionsoul\jcseg\filter\PPTFilter.java

```
package org.lionsoul.jcseg.filter;

import java.util.HashMap;
import java.util.Map;

/**
 * a class to deal with the text bettween the pair punctuations.
 *
 * @authorchenxin
 */
public class PPTFilter {

private static final Character[] PAIR_PUNCTUATION = {
/*'‘', '’', '“', '”',*/ '', '', '', '', '', ''};

private static Map<Character, Character> pairPunctuation = null;

static {
pairPunctuation = new HashMap<Character, Character>(
( int )(PAIR_PUNCTUATION.length / 1.7) + 1, 0.85f);
for ( int j = 0; j < PAIR_PUNCTUATION.length; j += 2 )
pairPunctuation.put(PAIR_PUNCTUATION[j], PAIR_PUNCTUATION[j+1]);
}

/**
 * check the given char is pair punctuation or not.
 *
 * @param c
 */
```

```
public static boolean isPairPunctuation( char c ) {
return pairPunctuation.containsKey(c);
}

/**
 * get the pair punctuation' pair.
 *
 * @param c
 * @return char
 */
public static char getPunctuationPair( char c ) {
return pairPunctuation.get(c);
}

}
```

22:F:\git\java\search\jcseg-1.9.5\jcseg-core\src\main\java\org\lionsoul\jcseg\LASTRule.java

```
package org.lionsoul.jcseg;

import org.lionsoul.jcseg.core.IChunk;
import org.lionsoul.jcseg.core.ILastRule;
import org.lionsoul.jcseg.core.IRule;

/**
 * the last rule.
 * -clear the ambiguity after the four rule.
 *
 * @authorchenxin
 */
public class LASTRule implements ILastRule {

/**
 * maxmum match rule instance.
 */
private static LASTRule __instance = null;

/**
 * return the quote to the maximum match instance.
 *
 * @return MMRule
 */
public static LASTRule createRule() {
```

```
if ( __instance == null )
__instance = new LASTRule();
return __instance;
}

private LASTRule() {}

/**
 * last rule interface.
 * here we simply return the first chunk.
 *
 * @see IRule#call(IChunk[])
 */
@Override
public IChunk call(IChunk[] chunks) {
return chunks[0];
}

}
```

23:F:\git\java\search\jcseg-1.9.5\jcseg-core\src\main\java\org\lionsoul\jcseg\LAWLRule.java

```
package org.lionsoul.jcseg;

import java.util.ArrayList;

import org.lionsoul.jcseg.core.IChunk;
import org.lionsoul.jcseg.core.IRule;


/**
 * the second filter rule
 * - largest average word length.
 * this rule will return the chunks that own
 * the largest average word length.
 *
 * @authorchenxin
 */
public class LAWLRule implements IRule {

/**
 * maxmum match rule instance.
 */
```

```
private static LAWLRule __instance = null;

/**
 * return the quote to the maximum match instance.
 *
 * @return MMRule
 */
public static LAWLRule createRule() {
if ( __instance == null )
__instance = new LAWLRule();
return __instance;
}

private LAWLRule() {}

/**
 * interface for largest average word length.
 *
 * @see IRule#call(IChunk[])
 */
@Override
public IChunk[] call(IChunk[] chunks) {

double largetAverage = chunks[0].getAverageWordsLength();
int j;

//find the largest average word length
for ( j = 1; j < chunks.length; j++ ) {
if ( chunks[j].getAverageWordsLength() > largetAverage )
largetAverage = chunks[j].getAverageWordsLength();
}

//get the items that the average word length equals to
//the max's.
ArrayList<IChunk> chunkArr = new ArrayList<IChunk>(chunks.length);
for ( j = 0; j < chunks.length; j++ ) {
if ( chunks[j].getAverageWordsLength() == largetAverage)
chunkArr.add(chunks[j]);
}

IChunk[] lchunk = new IChunk[chunkArr.size()];
chunkArr.toArray(lchunk);
```

```
chunkArr.clear();

return lchunk;
}

}

24:F:\git\java\search\jcseg-1.9.5\jcseg-core\src\main\java\org\lionsoul\jcseg\LSWMFRule.java
package org.lionsoul.jcseg;

import java.util.ArrayList;

import org.lionsoul.jcseg.core.IChunk;
import org.lionsoul.jcseg.core.IRule;


/**
 * the fouth filter rule
 * -the largest sum of degree of morphemic freedom
 * of one-character words.
 * this rule will return the chunks that own
 * the largest sum of degree of morphemic freedom of one-character.
 *
 * @authorchenxin
 */
public class LSWMFRule implements IRule {

/**
 * maxmum match rule instance.
 */
private static LSWMFRule __instance = null;

/**
 * return the quote to the maximum match instance.
 *
 * @return MMRule
 */
public static LSWMFRule createRule() {
if ( __instance == null )
__instance = new LSWMFRule();
return __instance;
}
```

```
private LSWMFRule() {}

/**
 * largest single word morphemic freedom.
 *
 * @see IRule#call(IChunk[])
 */
@Override
public IChunk[] call(IChunk[] chunks) {

double largestFreedom = chunks[0].getSingleWordsMorphemicFreedom();
int j;

//find the maximum sum of single morphemic freedom
for ( j = 1; j < chunks.length; j++ ) {
if ( chunks[j].getSingleWordsMorphemicFreedom() > largestFreedom )
largestFreedom = chunks[j].getSingleWordsMorphemicFreedom();
}

//get the items that the word length equals to
//the max's length.
ArrayList<IChunk> chunkArr = new ArrayList<IChunk>(chunks.length);
for ( j = 0; j < chunks.length; j++ ) {
if ( chunks[j].getSingleWordsMorphemicFreedom() == largestFreedom)
chunkArr.add(chunks[j]);
}

IChunk[] lchunk = new IChunk[chunkArr.size()];
chunkArr.toArray(lchunk);
chunkArr.clear();

return lchunk;
}

}
```

25:F:\git\java\search\jcseg-1.9.5\jcseg-core\src\main\java\org\lionsoul\jcseg\MMRule.java

```
package org.lionsoul.jcseg;

import java.util.ArrayList;
```

```
import org.lionsoul.jcseg.core.IChunk;
import org.lionsoul.jcseg.core.IRule;


/**
 * the first filter rule
 * - the maxmum match rule for JCSeg.
 * this rule will return the chunks that own
 * the largest word length.
 *
 * @authorchenxin
 */
public class MMRule implements IRule {

/**
 * maxmum match rule instance.
 */
private static MMRule __instance = null;

/**
 * return the quote to the maximum match instance.
 *
 * @return MMRule
 */
public static MMRule createRule() {
if ( __instance == null )
__instance = new MMRule();
return __instance;
}

private MMRule() {}

/**
 * interface for maximum match rule.
 *
 * @see IRule#call(IChunk[])
 */
@Override
public IChunk[] call(IChunk[] chunks) {

int maxLength = chunks[0].getLength();
int j;
```

```
//find the maximum word length
for ( j = 1; j < chunks.length; j++ ) {
if ( chunks[j].getLength() > maxLength )
maxLength = chunks[j].getLength();
}

//get the items that the word length equals to
//the max's length.
ArrayList<IChunk> chunkArr = new ArrayList<IChunk>(chunks.length);
for ( j = 0; j < chunks.length; j++ ) {
if ( chunks[j].getLength() == maxLength)
chunkArr.add(chunks[j]);
}

IChunk[] lchunk = new IChunk[chunkArr.size()];
chunkArr.toArray(lchunk);
chunkArr.clear();

return lchunk;
}

}
```

26:F:\git\java\search\jcseg-1.9.5\jcseg-core\src\main\java\org\lionsoul\jcseg\SimpleSeg.java

```
package org.lionsoul.jcseg;

import java.io.IOException;
import java.io.Reader;

import org.lionsoul.jcseg.core.ADictionary;
import org.lionsoul.jcseg.core.IChunk;
import org.lionsoul.jcseg.core.IWord;
import org.lionsoul.jcseg.core.JcsegTaskConfig;



/**
 * simplex segment for JCSeg,
 * has extend from ASegment.
 *
 * @authorchenxin
 */
public class SimpleSeg extends ASegment {
```

```
public SimpleSeg( JcsegTaskConfig config, ADictionary dic ) throws IOException {
super(config, dic);
}

public SimpleSeg( Reader input,
JcsegTaskConfig config, ADictionary dic ) throws IOException {
super(input, config, dic);
}

/**
 * @see ASegment#getBestCJKChunk(char[], int)
 */
@Override
public IChunk getBestCJKChunk(char[] chars, int index) {
IWord[] words = getNextMatch(chars, index);
return new Chunk(new IWord[]{words[words.length - 1]});
}

}
```

27:F:\git\java\search\jcseg-1.9.5\jcseg-core\src\main\java\org\lionsoul\jcseg\SVWLRule.java

```
package org.lionsoul.jcseg;

import java.util.ArrayList;

import org.lionsoul.jcseg.core.IChunk;
import org.lionsoul.jcseg.core.IRule;


/**
 * the third filter rule.
 * - the smallest variance word length.
 * this rule will the chunks that one
 * the smallest variance word length.
 *
 * @authorchenxin
 */
public class SVWLRule implements IRule {

/**
 * maxmum match rule instance.
```

```
 */
private static SVWLRule __instance = null;

/**
 * return the quote to the maximum match instance.
 *
 * @return MMRule
 */
public static SVWLRule createRule() {
if ( __instance == null )
__instance = new SVWLRule();
return __instance;
}

private SVWLRule() {}

/**
 * smallest variance word length interface.
 *
 * @see IRule#call(IChunk[])
 */
@Override
public IChunk[] call(IChunk[] chunks) {

double smallestVariance = chunks[0].getWordsVariance();
int j;

//find the smallest variance word length
for ( j = 1; j < chunks.length; j++ ) {
if ( chunks[j].getWordsVariance() < smallestVariance )
smallestVariance = chunks[j].getWordsVariance();
}

//get the items that the variance word length equals to
//the max's.
ArrayList<IChunk> chunkArr = new ArrayList<IChunk>(chunks.length);
for ( j = 0; j < chunks.length; j++ ) {
if ( chunks[j].getWordsVariance() == smallestVariance)
chunkArr.add(chunks[j]);
}

IChunk[] lchunk = new IChunk[chunkArr.size()];
```

```
chunkArr.toArray(lchunk);
chunkArr.clear();

return lchunk;
}

}
```

28:F:\git\java\search\jcseg-1.9.5\jcseg-core\src\main\java\org\lionsoul\jcseg\test\IHashQueueTest.java

```
package org.lionsoul.jcseg.test;

import org.lionsoul.jcseg.Word;
import org.lionsoul.jcseg.core.IWord;
import org.lionsoul.jcseg.util.IHashQueue;

/**
 * IHashQueue util class test program
 *
 * @author chenxin
 */

public class IHashQueueTest
{

/**
 * @param args
 */
public static void main(String[] args)
{
IWord[] ws = {
new Word("", IWord.T_CJK_WORD),
new Word("", IWord.T_BASIC_LATIN),
new Word("", IWord.T_CJK_WORD),
new Word("", IWord.T_BASIC_LATIN),
new Word("", IWord.T_CJK_WORD),
new Word("java",IWord.T_BASIC_LATIN),

new Word("", IWord.T_CJK_WORD),
new Word("", IWord.T_BASIC_LATIN),
new Word("java",IWord.T_BASIC_LATIN)
};
```

```
//-----------------------------------------------

IHashQueue<IWord> wordPool = new IHashQueue<IWord>();

int idx = 0;
for ( IWord w : ws )
{
if ( wordPool.contains(w) )
{
System.out.println("repeat: "+w);
}
else
{
w.setPosition(idx);
wordPool.add(w);
idx++;
}
}

//-------------------------------------------

while ( wordPool.size() > 0 )
{
System.out.println(wordPool.remove());
}

System.out.println("Done");
}

}
```

29:F:\git\java\search\jcseg-1.9.5\jcseg-core\src\main\java\org\lionsoul\jcseg\test\IIntFIFOTest.java

```
package org.lionsoul.jcseg.test;

import org.lionsoul.jcseg.util.IIntFIFO;

/**
 * IIntFIFO test program
 *
 * @author chenxin
 */
```

```
public class IIntFIFOTest {

/**
 * @param args
 */
public static void main(String[] args)
{
IIntFIFO q = new IIntFIFO();

q.enQueue('A');
q.enQueue('B');
System.out.println("size: " + q.size());

q.enQueue('C');
q.enQueue('D');
System.out.println("size: " + q.size());

while ( q.size() > 0 )
{
System.out.println("size: " + q.size() + ", " + (char)q.deQueue());
}
}

}

30:F:\git\java\search\jcseg-1.9.5\jcseg-
core\src\main\java\org\lionsoul\jcseg\test\IIntQueueTest.java
package org.lionsoul.jcseg.test;

import org.lionsoul.jcseg.util.IIntQueue;

/**
 * IIntQueue class test program
 *
 * @author chenxin
 */
public class IIntQueueTest {

/**
 * @param args
 */
public static void main(String[] args)
```

```
{
IIntQueue q = new IIntQueue();

//enqueue
/*for ( int i = 0; i < 100; i++ )
{
q.enQueue(i);
}
System.out.println("size: "+q.size());*/

//dequeue test
/*for ( int i = 0; i < 1000; i++ )
{
if ( i != 0 ) System.out.print(", ");
System.out.print(q.deQueue());
}
System.out.println("size: "+q.size());*/

q.enQueue('A');
q.enQueue('B');
System.out.println("size: "+q.size());

q.enQueue('C');
System.out.println((char)q.deQueue());
q.enQueue('D');
q.enQueue('E');

while ( q.size() > 0 )
{
System.out.println((char)q.deQueue()+", size: " + q.size());
}
}

}
```

31:F:\git\java\search\jcseg-1.9.5\jcseg-core\src\main\java\org\lionsoul\jcseg\test\IntArrayListTest.java

```
package org.lionsoul.jcseg.test;

import org.lionsoul.jcseg.util.IntArrayList;

/**
```

```
 * IntArrayList class Simple test program.
 *
 * @author chenxin
 */
public class IntArrayListTest {

/**
 * @param args
 */
public static void main(String[] args) {
IntArrayList list = new IntArrayList();

System.out.println("+---Test add: ");
//add some elements.
for ( int j = 0; j < 10; j++ ) {
list.add(j);
}
System.out.println("size="+list.size()+"\n");

list.set(0, 11);
list.set(3, 10);

System.out.println("+---Test get: ");
for ( int j = 0; j < list.size(); j++ )
System.out.println("get("+j+")="+list.get(j));
System.out.println("\n");

System.out.println("+---Test remove: ");
for ( int j = 0; j < 3; j++ ) {
int i = ((int)( Math.random() * 1000))%list.size();
list.remove(i);
System.out.print("remove("+i+")");
System.out.println(", size="+list.size());
}
System.out.println("\n");

System.out.println("+---Left: ");
for ( int j = 0; j < list.size(); j++ ) {
System.out.println("get("+j+")="+list.get(j));
}
}
```

```
}

32:F:\git\java\search\jcseg-1.9.5\jcseg-core\src\main\java\org\lionsoul\jcseg\test\IStringBufferTest.java
package org.lionsoul.jcseg.test;

import org.lionsoul.jcseg.util.IStringBuffer;

public class IStringBufferTest {

/**
 * @param args
 */
public static void main(String[] args) {
IStringBuffer isb = new IStringBuffer();

long s = System.currentTimeMillis();
for ( int j = 0; j < 10; j++ ) {
isb.append(j+"");
isb.append(',');
}
long e = System.currentTimeMillis();
System.out.println("Done, cost:"+(e-s)+"msec, "+isb.toString());

isb.clear();
s = System.currentTimeMillis();
for ( int j = 0; j < 10; j++ ) {
isb.append(""+j);
isb.append(',');
}
e = System.currentTimeMillis();
System.out.println("charAt(4)="+isb.charAt(4));
isb.deleteCharAt(4);
System.out.println("Done, cost:"+(e-s)+"msec, "+isb.toString());
}

}

33:F:\git\java\search\jcseg-1.9.5\jcseg-core\src\main\java\org\lionsoul\jcseg\test\JcsegCustomTest.java
package org.lionsoul.jcseg.test;
```

```
import org.lionsoul.jcseg.core.*;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.StringReader;


/**
 * jcseg test program.
 *
 * @authorchenxin
 */
public class JcsegCustomTest
{

ISegment seg = null;

public JcsegCustomTest() throws JcsegException, IOException {

JcsegTaskConfig config = new JcsegTaskConfig();
//JcsegTaskConfig config = new JcsegTaskConfig("/java/JavaSE/jcseg/jcseg.properties");
//JcsegTaskConfig config = new JcsegTaskConfig(null);
//reset the options from a property file.
//config.resetFromPropertyFile("/java/JavaSE/jcseg/jcseg.properties");

ADictionary dic = DictionaryFactory.createDefaultDictionary(config);

//two ways to reload lexicons
//for ( String lpath : config.getLexiconPath() )
//dic.loadFromLexiconDirectory(lpath);
//dic.loadFromLexiconFile("/java/lex-main.lex");
seg = SegmentFactory
.createJcseg(JcsegTaskConfig.COMPLEX_MODE, new Object[]{config, dic});

//detect mode test
//seg= SegmentFactory
//.createJcseg(JcsegTaskConfig.DETECT_MODE, new Object[]{config, dic});

//append pinyin
//config.setAppendCJKPinyin(true);
System.out.println("jcseg");
```

```
System.out.println(""+config.getPropertieFile());
System.out.println(""+config.MAX_LENGTH);
System.out.println(""+config.MIX_CN_LENGTH);
System.out.println(""+config.I_CN_NAME);
System.out.println(""+config.MAX_CN_LNADRON);
System.out.println(""+config.PPT_MAX_LENGTH);
System.out.println(""+config.LOAD_CJK_PINYIN);
System.out.println(""+config.APPEND_CJK_PINYIN);
System.out.println(""+config.LOAD_CJK_SYN);
System.out.println(""+config.APPEND_CJK_SYN);
System.out.println(""+config.LOAD_CJK_POS);
System.out.println(""+config.CLEAR_STOPWORD);
System.out.println(""+config.CNNUM_TO_ARABIC);
System.out.println(""+config.CNFRA_TO_ARABIC);
System.out.println(""+config.KEEP_UNREG_WORDS);
System.out.println(""+config.EN_SECOND_SEG);
System.out.println(""+config.NAME_SINGLE_THRESHOLD+"\n");
}

public void segment(String str) throws IOException
{

StringBuffer sb = new StringBuffer();
//seg.setLastRule(null);
IWord word = null;

long _start = System.nanoTime();
boolean isFirst = true;
int counter = 0;
seg.reset(new StringReader(str));
while ( (word = seg.next()) != null )
{
if ( isFirst ) {
sb.append(word.getValue());
isFirst = false;
//
sb.append(" ");
sb.append(word.getPosition());
}
else {
sb.append(" ");
sb.append(word.getValue());
```

```
//
sb.append(" ");
sb.append(word.getPosition());
}

//----for testing append word position and length
/*sb.append("[");
sb.append(word.getPosition());
sb.append("/");
sb.append(word.getLength());
sb.append("]");*/

//append the part of the speech
if ( word.getPartSpeech() != null ) {
sb.append('/');
sb.append(word.getPartSpeech()[0]);
}
//clear the allocations of the word.
word = null;
counter++;
}

long e = System.nanoTime();
System.out.println("");
System.out.println(sb.toString());
System.out.format("Done, total:"
+seg.getStreamPosition()+", split:" +
+counter+", cost: %.5fsec(less than)\n", ((float)e - _start)/1E9);
}

/**
 * @param args
 * @throws JcsegException
 * @throws IOException
 */
public static void main(String[] args) throws JcsegException, IOException {
String str = ":" +
": Bxktvoka" +
": 200938.6, 101.48, " +
"/: , ,"+
": jcesg" +
"" +
```

```
": 09-2BFC++PHP"+
": " +
": bug report chenxin619315@gmail.com or visit http://code.google.com/p/jcseg, we all admire the hacker spirit!" +
":    .";
//str = "";
//str = ""“”“09-2BF”PHP";
//str = "";
//str = "";
//str = "Java";
//str = "c++,c#.net,b";
//str = ", ?";

String cmd = null;
BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));
JcsegCustomTest demo = new JcsegCustomTest();
System.out.println(str);
try {
demo.segment(str);
System.out.println("+--------jcseg chinese word segment demo-------+");
System.out.println("|- Suggest email: chenxin619315@gmail.com      |");
System.out.println("|- Run quit or exit to exit.                   |");
System.out.println("+----------------------------------------------+");
do {
System.out.print("jcseg>> ");
cmd = reader.readLine();
if ( cmd == null ) break;
cmd = cmd.trim();
if ( "".equals(cmd) ) continue;
if ( cmd.equals("quit") || cmd.equals("exit")) {
System.out.println("Thanks for trying jcseg, Bye!");
System.exit(0);
}

//segment
demo.segment(cmd);
} while ( true );
} catch (IOException e) {
e.printStackTrace();
}

System.out.println("Bye!");
```

```
}
}

34:F:\git\java\search\jcseg-1.9.5\jcseg-core\src\main\java\org\lionsoul\jcseg\test\JcsegTest.java
package org.lionsoul.jcseg.test;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.StringReader;

import org.lionsoul.jcseg.core.ADictionary;
import org.lionsoul.jcseg.core.DictionaryFactory;
import org.lionsoul.jcseg.core.ISegment;
import org.lionsoul.jcseg.core.IWord;
import org.lionsoul.jcseg.core.JcsegException;
import org.lionsoul.jcseg.core.JcsegTaskConfig;
import org.lionsoul.jcseg.core.SegmentFactory;


/**
 * jcseg test program.
 *
 * @authorchenxin
 */
public class JcsegTest
{

ISegment seg = null;

public JcsegTest() throws JcsegException, IOException {

JcsegTaskConfig config = new JcsegTaskConfig();
//JcsegTaskConfig config = new JcsegTaskConfig("/java/JavaSE/jcseg/jcseg.properties");
//JcsegTaskConfig config = new JcsegTaskConfig(null);
//reset the options from a property file.
//config.resetFromPropertyFile("/java/JavaSE/jcseg/jcseg.properties");

ADictionary dic = DictionaryFactory.createDefaultDictionary(config);

//two ways to reload lexicons
//for ( String lpath : config.getLexiconPath() )
```

```
//dic.loadFromLexiconDirectory(lpath);
//dic.loadFromLexiconFile("/java/lex-main.lex");
seg = SegmentFactory
.createJcseg(JcsegTaskConfig.COMPLEX_MODE, new Object[]{config, dic});

//detect mode test
//seg= SegmentFactory
//.createJcseg(JcsegTaskConfig.DETECT_MODE, new Object[]{config, dic});

//append pinyin
//config.setAppendCJKPinyin(true);
System.out.println("jcseg");
System.out.println(""+config.getPropertieFile());
System.out.println(""+config.MAX_LENGTH);
System.out.println(""+config.MIX_CN_LENGTH);
System.out.println(""+config.I_CN_NAME);
System.out.println(""+config.MAX_CN_LNADRON);
System.out.println(""+config.PPT_MAX_LENGTH);
System.out.println(""+config.LOAD_CJK_PINYIN);
System.out.println(""+config.APPEND_CJK_PINYIN);
System.out.println(""+config.LOAD_CJK_SYN);
System.out.println(""+config.APPEND_CJK_SYN);
System.out.println(""+config.LOAD_CJK_POS);
System.out.println(""+config.CLEAR_STOPWORD);
System.out.println(""+config.CNNUM_TO_ARABIC);
System.out.println(""+config.CNFRA_TO_ARABIC);
System.out.println(""+config.KEEP_UNREG_WORDS);
System.out.println(""+config.EN_SECOND_SEG);
System.out.println(""+config.NAME_SINGLE_THRESHOLD+"\n");
}

public void segment(String str) throws IOException
{

StringBuffer sb = new StringBuffer();
//seg.setLastRule(null);
IWord word = null;

long _start = System.nanoTime();
boolean isFirst = true;
int counter = 0;
seg.reset(new StringReader(str));
```

```
while ( (word = seg.next()) != null )
{
if ( isFirst ) {
sb.append(word.getValue());
isFirst = false;
sb.append(" ");
sb.append(word.getPosition());
}
else {
sb.append(" ");
sb.append(word.getValue());

sb.append(" ");
sb.append(word.getPosition());
}

//----for testing append word position and length
/*sb.append("[");
sb.append(word.getPosition());
sb.append("/");
sb.append(word.getLength());
sb.append("]");*/

//append the part of the speech
if ( word.getPartSpeech() != null ) {
sb.append('/');
sb.append(word.getPartSpeech()[0]);
}
//clear the allocations of the word.
word = null;
counter++;
}

long e = System.nanoTime();
System.out.println("");
System.out.println(sb.toString());
System.out.format("Done, total:"
+seg.getStreamPosition()+", split:" +
+counter+", cost: %.5fsec(less than)\n", ((float)e - _start)/1E9);
}

/**
```

```
 * @param args
 * @throws JcsegException
 * @throws IOException
 */
public static void main(String[] args) throws JcsegException, IOException {
String str = ":" +
": Bxktvoka" +
": 200938.6, 101.48, " +
"/: , ,"+
": jcesg" +
"" +
": 09-2BFC++PHP"+
": " +
": bug report chenxin619315@gmail.com or visit http://code.google.com/p/jcseg, we all admire the hacker spirit!" +
":    .";
//str = "";
//str = ""“”“09-2BF”PHP";
//str = "";
//str = "";
//str = "Java";
//str = "c++,c#.net,b";
//str = ", ?";

String cmd = null;
BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));
JcsegTest demo = new JcsegTest();
System.out.println(str);
try {
demo.segment(str);
System.out.println("+--------jcseg chinese word segment demo-------+");
System.out.println("|- Suggest email: chenxin619315@gmail.com      |");
System.out.println("|- Run quit or exit to exit.                   |");
System.out.println("+----------------------------------------------+");
do {
System.out.print("jcseg>> ");
cmd = reader.readLine();
if ( cmd == null ) break;
cmd = cmd.trim();
if ( "".equals(cmd) ) continue;
if ( cmd.equals("quit") || cmd.equals("exit")) {
System.out.println("Thanks for trying jcseg, Bye!");
```

```
System.exit(0);
}

//segment
demo.segment(cmd);
} while ( true );
} catch (IOException e) {
e.printStackTrace();
}

System.out.println("Bye!");
}
}
```

35:F:\git\java\search\jcseg-1.9.5\jcseg-core\src\main\java\org\lionsoul\jcseg\test\SpeedTest.java

```
package org.lionsoul.jcseg.test;

import java.io.BufferedReader;
import java.io.FileInputStream;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.Reader;
import java.io.StringReader;

import org.lionsoul.jcseg.core.ADictionary;
import org.lionsoul.jcseg.core.DictionaryFactory;
import org.lionsoul.jcseg.core.ISegment;
import org.lionsoul.jcseg.core.IWord;
import org.lionsoul.jcseg.core.JcsegException;
import org.lionsoul.jcseg.core.JcsegTaskConfig;
import org.lionsoul.jcseg.core.SegmentFactory;


/**
 * jcseg speed test program .
 *
 * @authorchenxin
 */
public class SpeedTest {

public static ISegment seg = null;
```

```
public static String segment(Reader reader, int type)
throws JcsegException, IOException
{

if ( seg == null )
{
long start = System.currentTimeMillis();
JcsegTaskConfig  config = new JcsegTaskConfig();
ADictionary dic = DictionaryFactory.createDefaultDictionary(config);
//load lexicon
//for ( String lpath : config.getLexiconPath() )
//dic.loadFromLexiconDirectory(lpath);
seg = SegmentFactory.createJcseg(JcsegTaskConfig.COMPLEX_MODE,
new Object[]{config, dic});
System.out.println("Diciontary Loaded, cost:"+
(System.currentTimeMillis() - start)+" msec");
}

StringBuilder sb = new StringBuilder();
seg.reset(reader);
//seg.setLastRule(null);
IWord word = null;

int counter = 0;
long _start = System.currentTimeMillis();
while ( (word = seg.next()) != null )
{
sb.append(word.getValue());
sb.append("  ");
counter++;
}
System.out.println("Done, cost:"+(System.currentTimeMillis() - _start)+" msec");
System.out.println(""+seg.getStreamPosition()+", split: "+counter);

return sb.toString();
}

/**
 * @param args
 */
public static void main(String[] args)
{
```

```
String filename = "/java/products/jcseg_o/article/article";
if (  args.length >= 1 )
filename = args[0];
try {
segment(new StringReader("jcseg"), JcsegTaskConfig.COMPLEX_MODE);
segment(new BufferedReader(
new InputStreamReader(
new FileInputStream(filename), "UTF-8")),
JcsegTaskConfig.COMPLEX_MODE);
//System.out.println("Complex-> "+segment(sb.toString(), Config.COMPLEX_MODE));
} catch (Exception e) {
e.printStackTrace();
}
}

}
```

36:F:\git\java\search\jcseg-1.9.5\jcseg-core\src\main\java\org\lionsoul\jcseg\test\STConverterTest.java

```
package org.lionsoul.jcseg.test;

import org.lionsoul.jcseg.util.STConverter;

public class STConverterTest {

/**
 * @param args
 */
public static void main(String[] args) {

String str = "Jcseg, java.";
System.out.println("str = " + str);

String tra = STConverter.SimToTraditional(str);
System.out.println("Simplified to traditional: " + tra);

String sim = STConverter.TraToSimplified(tra);
System.out.println("Traditional to simplified: " + sim);
}
}
```

37:F:\git\java\search\jcseg-1.9.5\jcseg-core\src\main\java\org\lionsoul\jcseg\util\IHashQueue.java

```
package org.lionsoul.jcseg.util;

import java.util.HashMap;
import java.util.Map;

import org.lionsoul.jcseg.core.IWord;

/**
 * A normal queue base one single link
 * but with hash index, so, it is fast for searching
 *
 * Currently build to replace the LinkList work pool
 * of class org.lionsoul.jcseg.ASegment
 *
 *
 * @author chenxin
 */

 //-------------------------------------------------

public class IHashQueue<T extends IWord>
{
private int size;//size of the current queue
private Entry<T> head;//head of the queue
private Entry<T> tail;//tail of the queue

private Map<String, T>index;//hash index layer

public IHashQueue()
{
size = 0;
tail = new Entry<T>(null, null, null);
head = new Entry<T>(null, null, tail);
tail.prev = head;

//initialize the hash indexer
index= new HashMap<String, T>(16, 0.85F);
}

/**
 * append a item from the tail
 *
```

```
 * @paramword
 * @returnboolean
 */
public boolean add( T word )
{
Entry<T> o = new Entry<T>(word, tail.prev, tail);
tail.prev.next = o;
tail.prev = o;

//set the size and set the index
size++;
index.put(word.getValue(), word);

return true;
}

/**
 * check the specifield T is aleady exists in the queue or not
 *
 * @paramword
 * @returnboolean
 */
public boolean contains( T word )
{
return index.containsKey(word.getValue());
}

/**
 * remove the node from the head
 * and you should make sure the size is larger than 0 by calling size()
 * before you invoke the method or you will just get null
 *
 */
public T remove()
{
if ( size == 0 ) return null;

//remove the first element
Entry<T> o = head.next;
head.next = o.next;
o.next.prev = head;
```

```
//bakup the data
T v = o.data;
size--;
index.remove(v.getValue());

o = null;//Let gc do its work

return v;
}

/**
 * get the size of the queue
 *
 * @return int
 */
public int size()
{
return size;
}

/**
 * innner Entry node class
 *
 * @author chenxin
 */
public static class Entry<T>
{
public T data;//data of the current node
public Entry<T> prev;//prev entry quote
public Entry<T> next;//next entry quote

public Entry( T data,
Entry<T> prev, Entry<T> next )
{
this.data = data;
this.prev = prev;
this.next = next;
}
}
}
```

38:F:\git\java\search\jcseg-1.9.5\jcseg-core\src\main\java\org\lionsoul\jcseg\util\IIntFIFO.java

```
package org.lionsoul.jcseg.util;

/**
 * int first in first out queue
 * base on single link.
 *
 * @author chenxin
 */
public class IIntFIFO
{

//size of the queue
private int size;

//head entry of the queue
private Entry head;

public IIntFIFO()
{
size = 0;
head = new Entry(-1, null);
}

/**
 * add a new item to the queue
 *
 * @paramdata
 * @returnboolean
 */
public boolean enQueue( int data )
{
Entry o = new Entry(data, head.next);
head.next = o;
size++;

return true;
}

/**
 * remove the first item from the queue
 *
 * @returnint (It not good to return int)
```

```
 */
public int deQueue()
{
if ( size == 0 ) return -1;
Entry o = head.next;
head.next = o.next;

int v = o.data;//backup the data
o = null;//Let gc do its work
size--;

return v;
}

/**
 * get the size of the queue
 *
 * @returnint
 */
public int size()
{
return size;
}

/**
 * Item Entry inner class
 */
public static class Entry
{
public int data;//entry data
public Entry next;//next item

public Entry( int data, Entry next )
{
this.data = data;
this.next = next;
}
}
}
```

39:F:\git\java\search\jcseg-1.9.5\jcseg-core\src\main\java\org\lionsoul\jcseg\util\IIntQueue.java

```
package org.lionsoul.jcseg.util;
```

```
/**
 * char queue class base on double link
 * (Not thread safe)
 *
 * @author chenxin
 */
public class IIntQueue
{
private int size;//size of the current queue
private Entry head;//head of the queue
private Entry tail;//tail of the queue

public IIntQueue()
{
size = 0;
tail = new Entry(-1, null, null);
head = new Entry(-1, null, tail);
tail.prev = head;
}

/**
 * append a int from the tail
 *
 * @paramdata
 * @returnboolean
 */
public boolean enQueue( int data )
{
Entry o = new Entry(data, tail.prev, tail);
tail.prev.next = o;
tail.prev = o;

//set the size
size++;

return true;
}

/**
 * remove the node from the head
 * and you should make sure the size is larger than 0 by calling size()
```

```
 * before you invoke the method or you will just get -1
 *
 */
public int deQueue()
{
if ( size == 0 ) return -1;

//remove the first element
Entry o = head.next;
head.next = o.next;
o.next.prev = head;

//bakup the data
int v = o.data;
size--;
o = null;//Let gc do its work

return v;
}

/**
 * get the size of the queue
 *
 * @return int
 */
public int size()
{
return size;
}

/**
 * innner Entry node class
 */
public static class Entry
{
public int data;//data of the current node
public Entry prev;//prev entry quote
public Entry next;//next entry quote

public Entry( int data, Entry prev, Entry next )
{
this.data = data;
```

```
this.prev = prev;
this.next = next;
}
}
}
```

40:F:\git\java\search\jcseg-1.9.5\jcseg-core\src\main\java\org\lionsoul\jcseg\util\IntArrayList.java

```
package org.lionsoul.jcseg.util;

/**
 * array list for basic int data type.
 * to intead of ArrayList, Well, this will save a lot
 * work to Reopened and Unpacking.
 *
 * @author chenxin
 */
public class IntArrayList
{
private int size = 0;
//int items array.
private int[] items;

public IntArrayList()
{
this(6);
}

public IntArrayList( int opacity )
{
if ( opacity <= 0 )
throw new IndexOutOfBoundsException("opacity <= 0");
items = new int[opacity];
}

private void resize( int size )
{
int[] tmp = items;
items = new int[size];
int length = (size > tmp.length) ? tmp.length : size;

//copy the items to the tmp
/*for ( int j = 0; j < length; j++ ) {
```

```
items[j] = tmp[j];
}*/
System.arraycopy(tmp, 0, items, 0, length);
}

/**
 * Append a new Integer to the end.
 *
 * @paramval.
 */
public void add( int val )
{
if ( size == items.length )
resize( items.length * 2 + 1 );
items[size++] = val;
}

public int get( int idx )
{
if ( idx < 0 || idx > size )
throw new IndexOutOfBoundsException();
return items[idx];
}

public void set( int idx, int val )
{
if ( idx < 0 || idx > size )
throw new IndexOutOfBoundsException();
items[idx] = val;
}

/**
 * remove the element at the specified position.
 * use System.arraycopy intead of a loop may be
 * more effcient.
 *
 * @param idx
 */
public void remove( int idx )
{
if ( idx < 0 || idx > size )
throw new IndexOutOfBoundsException();
```

```
int numMove = size - idx - 1;
if ( numMove > 0 )
System.arraycopy(items, idx + 1, items, idx, numMove);
size--;
}

public int size()
{
return size;
}

public void clear()
{
size = 0;
}
}
```

41:F:\git\java\search\jcseg-1.9.5\jcseg-core\src\main\java\org\lionsoul\jcseg\util\IPushbackReader.java

```
package org.lionsoul.jcseg.util;

import java.io.IOException;
import java.io.Reader;

/**
 * IPushBackReader based on Reader
 * Not thread safe support unlimited unread operation
 *
 * @author chenxin
 */
public class IPushbackReader
{
//reader
private Reader reader = null;

//push buffer
private IIntFIFO queue = null;

public IPushbackReader( Reader reader )
{
this.reader = reader;
queue = new IIntFIFO();
```

```
}

/**
 * read the next int from the stream
 * this will check the buffer queue first
 * and take the first item of the buffer as the result
 *
 * @returnint
 * @throwsIOException
 */
public int read() throws IOException
{
//check the queue first
if ( queue.size() > 0 ) return queue.deQueue();

//load from the normal reader
return reader.read();
}

/**
 * read the specified block from the stream
 * @see #read()
 *
 * @return int
 * @throws IOException
 */
public int read( char[] cbuf, int off, int len ) throws IOException
{
//check the buffer queue
int size = queue.size();
if ( size > 0 )
{
//TODO
//int num = size <= len ? size : len;
//System.arraycopy(src, srcPos, dest, destPos, length)
throw new IOException("Method not implemented yet");
}

return reader.read(cbuf, off, len);
}

/**
```

```
 * unread the speicfied data to the stream
 * push the data back to the queue in fact, you know
 */
public void unread( int data )
{
queue.enQueue(data);
}

/**
 * get the buffer size - the number of buffered data
 *
 * @returnint
 */
public int getQueueSize()
{
return queue.size();
}

/**
 * unread a block from a char array to the stream
 *
 * @see#unread(int)
 */
public void unread( char[] cbuf, int off, int len )
{
for ( int i = 0; i < len; i++ )
queue.enQueue(cbuf[off+i]);
}
}
```

42:F:\git\java\search\jcseg-1.9.5\jcseg-core\src\main\java\org\lionsoul\jcseg\util\IStringBuffer.java

```
package org.lionsoul.jcseg.util;

/**
 * string buffer class.
 *
 * @authorchenxin
 */
public class IStringBuffer {

/**
 * buffer char array.
```

```
 */
private char buff[];
private int count;

/**
 * create a buffer with a default length 16.
 */
public IStringBuffer() {
buff = new char[16];
count = 0;
}

/**
 * create a buffer with a specified length.
 *
 * @paramlength
 */
public IStringBuffer( int length ) {
if ( length <= 0 )
throw new IllegalArgumentException("length <= 0");
buff = new char[length];
count = 0;
}

/**
 * create a buffer with a specified string.
 *
 * @paramstr
 */
public IStringBuffer( String str ) {
if ( str == null )
throw new NullPointerException();

buff = new char[str.length() + 16];
append(str);
count = 0;
}

/**
 * resize the buffer.
 * this will have to copy the old chars from the old buffer to the new buffer.
 */
```

```
private void resizeTo( int length ) {
if ( length <= 0 )
throw new IllegalArgumentException("length <= 0");
if ( length != buff.length ) {
int len = ( length > buff.length ) ? buff.length : length;
//System.out.println("resize:"+length);
char[] obuff = buff;
buff = new char[length];
/*for ( int j = 0; j < len; j++ ) {
buff[j] = obuff[j];
}*/
System.arraycopy(obuff, 0, buff, 0, len);
}
}

/**
 * append a string to the buffer.
 *
 * @paramstrstring to append to
 */
public IStringBuffer append( String str ) {
if ( str == null )
throw new NullPointerException();
//check the necessary to resize the buffer.
if ( count + str.length() > buff.length )
resizeTo( (count + str.length()) * 2 + 1 );
for ( int j = 0; j < str.length(); j++ ) {
buff[count++] = str.charAt(j);
}

return this;
}

/**
 * append parts of the chars to the buffer.
 *
 * @paramchars
 * @paramstartthe start index.
 * @paramlengthlength of chars to append to.
 */
public IStringBuffer append( char[] chars, int start, int length ) {
if ( chars == null )
```

```
throw new NullPointerException();
if ( start < 0 )
throw new IndexOutOfBoundsException();
if ( length <= 0 )
throw new IndexOutOfBoundsException();
if ( start + length >= chars.length )
throw new IndexOutOfBoundsException();

//check the necessary to resize the buffer.
if ( count + length > buff.length )
resizeTo( (count + length) * 2 + 1 );
for ( int j = 0; j < length; j++ ) {
buff[count++] = chars[start+j];
}

return this;
}

/**
 * append some chars to the buffer.
 *
 * @paramchars
 */
public IStringBuffer append( char[] chars ) {
return append(chars, 0, chars.length);
}

/**
 * append a char to the buffer.
 *
 * @paramcthe char to append to
 */
public IStringBuffer append( char c ) {
if ( count == buff.length ) resizeTo( buff.length * 2 + 1 );
buff[count++] = c;

return this;
}

/**
 * return the lenght of the buffer.
 *
```

```
 * @returnintthe length of the buffer.
 */
public int length() {
return count;
}

/**
 * get the char at a specified position in the buffer.
 */
public char charAt( int idx ) {
if ( idx < 0 )
throw new IndexOutOfBoundsException("idx < 0");
if ( idx >= count )
throw new IndexOutOfBoundsException("idx >= buffer.length");
return buff[idx];
}

/**
 * delete the char at the specified position.
 */
public IStringBuffer deleteCharAt( int idx ) {
if ( idx < 0 )
throw new IndexOutOfBoundsException("idx < 0");
if ( idx >= count )
throw new IndexOutOfBoundsException("idx >= buffer.length");

//here we got a bug for j < count
//change over it to count - 1
//thanks for the feedback of xuyijun@gmail.com
//@date 2013-08-22
for ( int j = idx; j < count - 1; j++ ) {
buff[j] = buff[j+1];
}
count--;

return this;
}

/**
 * return the chars of the buffer.
 *
 * @returnchar[]
```

```
 */
public char[] buffer() {
return buff;
}

/**
 * clear the buffer by
 * reset the count to 0.
 */
public IStringBuffer clear() {
count = 0;

return this;
}

/**
 * return the string of the current buffer.
 *
 * @returnString
 * @see Object#toString()
 */
public String toString() {
return new String(buff, 0, count);
}
}
```

43:F:\git\java\search\jcseg-1.9.5\jcseg-core\src\main\java\org\lionsoul\jcseg\util\Sort.java

```
package org.lionsoul.jcseg.util;

/**
 * All kind of Sort alogrithm implemented method.
 * use the default compare method.
 *
 * @author chenxin
 */
public class Sort {

private static final int CUTOFF = 11;
/*private static final int[] GAPS = new int[]{
1, 5,
13, 43,
113, 297, 815,
```

```
1989, 4711,
11969, 27901, 84801,
        213331, 543749,//1000th
        1355339, 3501671, 8810089,
        21521774, 58548857,
        157840433, 410151271,
        1131376761, 2147483647};*/

/**
 * shell sort gaps array.
 * generate with 9*pow(4, j) - 9 * pow(2, j) + 1,
 * and pow(4, j) - 3 * pow(2, j) + 1 .
 */
private static final int[] GAPS = new int[] {
1, 5,
19, 41,
109, 209, 505, 929,
2161, 8929,
16001, 36289, 64769,
146305, 260609, 587521,//1000th
1045505, 2354689, 4188161, 9427969,
16764929, 37730305, 67084289,
150958081, 268386305, 603906049,
1073643521, 2147483647};



/**
 * insert sort method.
 *
 * @param arr an array of a comparable items.
 */
public static <T extends Comparable<? super T>> void insertionSort( T[] arr ) {
int j;
for ( int i = 1; i < arr.length; i++ ) {
T tmp = arr[i];
for ( j = i; j > 0 && tmp.compareTo(arr[j-1]) < 0; j--) {
arr[j] = arr[j-1];
}
if ( j < i ) arr[j] = tmp;
}
}
```

```
/**
 * shell sort algorithm.
 *
 * @param arr an array of Comparable items.
 */
public static <T extends Comparable<? super T>> void shellSort( T[] arr ) {
int j, k = 0, gap;
for ( ; GAPS[k] < arr.length; k++ ) ;

while ( k-- > 0 ) {
gap = GAPS[k];
for ( int i = gap; i < arr.length; i++ ) {
T tmp = arr[ i ];
for ( j = i;
j >= gap && tmp.compareTo( arr[ j - gap ] ) < 0; j -= gap ) {
arr[ j ] = arr[ j - gap ];
}
if ( j < i ) arr[ j ] = tmp;
}
}
}
```

```
/**
 * merge sort algorithm.
 *
 * @param arr an array of Comparable item.
 */
@SuppressWarnings("unchecked")
public static <T extends Comparable<? super T>> void mergeSort( T[] arr ) {
/*if ( arr.length < 15 ) {
insertionSort( arr );
return;
}*/

T[] tmpArr = (T[]) new Comparable[arr.length];
```

```
mergeSort(arr, tmpArr, 0, arr.length - 1);
}

/**
 * internal method to make a recursive call.
 *
 * @param arr an array of Comparable items.
 * @param tmpArr temp array to placed the merged result.
 * @param left left-most index of the subarray.
 * @param right right-most index of the subarray.
 */
private static <T extends Comparable<? super T>>
void mergeSort( T[] arr, T[] tmpArr,
int left, int right ) {
//recursive way
if ( left < right ) {
int center = ( left + right ) / 2;
mergeSort(arr, tmpArr, left, center);
mergeSort(arr, tmpArr, center + 1, right);
merge(arr, tmpArr, left, center + 1, right);
}

//loop instead
/*int len = 2, pos;
int rpos, offset, cut;
while ( len <= right ) {
pos = 0;
offset = len / 2;
while ( pos + len <= right  ) {
rpos = pos + offset;
merge( arr, tmpArr, pos, rpos, rpos + offset - 1 );
pos += len;
}

//merge the rest
cut = pos + offset;
if ( cut <= right )
merge( arr, tmpArr, pos, cut, right );

len *= 2;
}
```

```
merge( arr, tmpArr, 0, len / 2, right );*/
}

/**
 * internal method to merge the sorted halves of a subarray.
 *
 * @param arr an array of Comparable items.
 * @param tmpArr temp array to placed the merged result.
 * @param leftPos left-most index of the subarray.
 * @param rightPos right start index of the subarray.
 * @param endPos right-most index of the subarray.
 */
private static <T extends Comparable<? super T>>
void merge( T[] arr, T[] tmpArr,
int lPos, int rPos, int rEnd ) {
int lEnd = rPos - 1;
int tPos = lPos;
int leftTmp = lPos;

while ( lPos <= lEnd && rPos <= rEnd  ) {
if ( arr[lPos].compareTo( arr[rPos] ) <= 0 )
tmpArr[ tPos++ ] = arr[ lPos++ ];
else
tmpArr[ tPos++ ] = arr[ rPos++ ];
}

//copy the rest element of the left half subarray.
while ( lPos <= lEnd )
tmpArr[ tPos++ ] = arr[ lPos++ ];
//copy the rest elements of the right half subarray. (only one loop will be execute)
while ( rPos <= rEnd )
tmpArr[ tPos++ ] = arr[ rPos++ ];

//copy the tmpArr back cause we need to change the arr array items.
for ( ; rEnd >= leftTmp; rEnd-- )
arr[rEnd] = tmpArr[rEnd];
}




/**
```

```
 * method to swap elements in an array.
 *
 * @param arr an array of Objects.
 * @param idx1 the index of the first element.
 * @param idx2 the index of the second element.
 */
private static <T> void swapReferences( T[] arr, int idx1, int idx2 ) {
T tmp = arr[idx1];
arr[idx1] = arr[idx2];
arr[idx2] = tmp;
}



/**
 * quick sort algorithm.
 *
 * @param arr an array of Comparable items.
 */
public static <T extends Comparable<? super T>> void quicksort( T[] arr ) {
quicksort( arr, 0, arr.length - 1 );
}

/**
 * get the median of the left, center and right.
 * order these and hide the pivot by put it the end of
 * of the array.
 *
 * @param arr an array of Comparable.
 * @param left the most-left index of the subarray.
 * @param right the most-right index of the subarray.
 * @return T
 */
private static <T extends Comparable<? super T>>
T median( T[] arr, int left, int right ) {

int center = ( left + right ) / 2;

if ( arr[left].compareTo( arr[center] ) > 0 )
swapReferences( arr, left, center );
if ( arr[left].compareTo( arr[right] ) > 0 )
```

```
swapReferences( arr, left, right );
if ( arr[center].compareTo( arr[right] ) > 0 )
swapReferences( arr, center, right );

swapReferences( arr, center, right - 1 );
return arr[ right - 1 ];
}

/**
 * method to sort an subarray from start to end
 * with insertion sort algorithm.
 *
 * @param arr an array of Comparable items.
 * @param start the begining position.
 * @param end the end position.
 */
public static <T extends Comparable<? super T>>
void insertionSort( T[] arr, int start, int end ) {
int i;
for ( int j = start + 1; j <= end; j++ ) {
T tmp = arr[j];
for ( i = j; i > start && tmp.compareTo( arr[i - 1] ) < 0; i-- ) {
arr[ i ] = arr[ i - 1 ];
}
if ( i < j ) arr[ i ] = tmp;
}
}

/**
 * internal method to sort the array with quick sort algorithm.
 *
 * @param arr an array of Comparable Items.
 * @param left the left-most index of the subarray.
 * @param right the right-most index of the subarray.
 */
private static <T extends Comparable<? super T>>
void quicksort( T[] arr, int left, int right ) {
if ( left + CUTOFF <= right  ) {
//find the pivot
T pivot = median( arr, left, right );

//start partitioning
```

```
int i = left, j = right - 1;
for ( ; ; ) {
while ( arr[++i].compareTo( pivot ) < 0 ) ;
while ( arr[--j].compareTo( pivot ) > 0 ) ;
if ( i < j )
swapReferences( arr, i, j );
else
break;
}

//swap the pivot reference back to the small collection.
swapReferences( arr, i, right - 1 );

quicksort( arr, left, i - 1 );//sort the small collection.
quicksort( arr, i + 1, right );//sort the large collection.

} else {
//if the total number is less than CUTOFF we use insertion sort instead.
insertionSort( arr, left, right );
}
}
```

```
/**
 * quick select algorithm.
 *
 * @param arr an array of Comparable items.
 * @param k the k-th small index.
 */
public static <T extends Comparable<? super T>>
void quickSelect( T[] arr, int k ) {
quickSelect( arr, 0, arr.length - 1, k );
}

/**
 * internal method to find the Kth small element for the given array.
 *
 * @param arr an array of Comparable items.
 * @param left the left-most index of the subarray.
 * @param right the right-most index of the subarray.
```

```
 * @param k the k-th small element.
 */
private static <T extends Comparable<? super T>>
void quickSelect( T[] arr, int left, int right, int k ) {
if ( left + CUTOFF <= right ) {
//find the pivot
T pivot = median( arr, left, right );

int i = left, j = right - 1;
for ( ; ; ) {
while ( arr[ ++i ].compareTo( pivot ) < 0 ) ;
while ( arr[ --j ].compareTo( pivot ) > 0 ) ;
if ( i < j )
swapReferences( arr, i, j );
else
break;
}

//swap the pivot
swapReferences( arr, i, right - 1 );

if ( k <= i )
quickSelect( arr, left, i - 1, k );
else if ( k > i + 1 )
quickSelect( arr, i + 1, right, k );

} else {
insertionSort( arr, left, right );
}
}
```

```
/**
 * bucket sort algorithm.
 *
 * @param arr an int array.
 * @param m the large-most one for all the Integers in arr
 */
public static void bucketSort( int[] arr, int m ) {
int[] count = new int[m];
int j, i = 0;
```

```
//System.out.println(count[0]==0?"true":"false");
for ( j = 0; j < arr.length; j++ )
count[ arr[j] ]++;

//loop and filter the elements
for ( j = 0; j < m; j++ ) {
if ( count[j] > 0 ) {
while ( count[j]-- > 0 )
arr[i++] = j;
}
}
}

/**
 * bucket sort algorithm.
 *
 * @param arr an array of Integer items.
 * @param m the large-most one for all the Integers in arr
 */
public static void bucketSort( Integer[] arr, int m ) {
int[] count = new int[m];
int j, i = 0;
for ( j = 0; j < arr.length; j++ )
count[ arr[j] ]++;

//loop and filter the elements
for ( j = 0; j < m; j++ )
if ( count[j] > 0 ) {
while ( count[j]-- > 0 )
arr[i++] = new Integer(j);
}
}

}
```

44:F:\git\java\search\jcseg-1.9.5\jcseg-core\src\main\java\org\lionsoul\jcseg\util\STConverter.java

```
package org.lionsoul.jcseg.util;


/**
 * Simplified and traditional chinese convert class.
 * all the search work base on {@link String#indexOf(int)}.
```

```
 * you may store all the words in a HashMap for the purpuse of a faster fetch.
 *
 * @author chenxin
 */
public class STConverter {

//simplified string.
public static final String SIMSTR = "";

//traditional string.
public static final String TRASTR = "Pq";

/**
 * convert the simplified words to traditional words
 * of the specified string.
 *
 * @paramstr
 * @returnString
 */
public static String SimToTraditional( String str ) {
StringBuffer sb = new StringBuffer();
int idx;
for ( int j = 0; j < str.length(); j++ )
{
if ( (idx = SIMSTR.indexOf(str.charAt(j))) != -1 )
sb.append(TRASTR.charAt(idx));
else
sb.append(str.charAt(j));
}

return sb.toString();
}

public static void SimToTraditional( String str, IStringBuffer isb ) {
int idx;
for ( int j = 0; j < str.length(); j++ )
{
if ( (idx = SIMSTR.indexOf(str.charAt(j))) != -1 )
isb.append(TRASTR.charAt(idx));
else
isb.append(str.charAt(j));
}
```

```
}

/**
 * convert the traditional words to simplified words.
 * of the specified string.
 *
 * @paramstr
 * @returnString
 */
public static String TraToSimplified( String str ) {
StringBuffer sb = new StringBuffer();
int idx;
for ( int j = 0; j < str.length(); j++ )
{
if ( (idx = TRASTR.indexOf(str.charAt(j))) != -1 )
sb.append(SIMSTR.charAt(idx));
else
sb.append(str.charAt(j));
}
return sb.toString();
}

public static void TraToSimplified( String str, IStringBuffer isb ) {
int idx;
for ( int j = 0; j < str.length(); j++ )
{
if ( (idx = TRASTR.indexOf(str.charAt(j))) != -1 )
isb.append(SIMSTR.charAt(idx));
else
isb.append(str.charAt(j));
}
}
}

45:F:\git\java\search\jcseg-1.9.5\jcseg-core\src\main\java\org\lionsoul\jcseg\util\Util.java
package org.lionsoul.jcseg.util;

import java.io.File;

/**
 * static method for jcseg.
 *
```

```
 * @authorchenxin
 */
public class Util {

/**
 * get the absolute parent path for the jar file.
 *
 * @param o
 * @return String
 */
public static String getJarHome(Object o) {
String path = o.getClass().getProtectionDomain()
.getCodeSource().getLocation().getFile();
File jarFile = new File(path);
return jarFile.getParentFile().getAbsolutePath();
}

}
```

46:F:\git\java\search\jcseg-1.9.5\jcseg-core\src\main\java\org\lionsoul\jcseg\Word.java

```
package org.lionsoul.jcseg;

import org.lionsoul.jcseg.core.IWord;


/**
 * word class for jcseg has implements IWord interface
 *
 * @authorchenxin
 */
public class Word implements IWord,Cloneable
{

private String value;
private int fre = 0;
private int type;
private int position;
private String pinyin = null;
private String[] partspeech = null;
private String[] syn = null;

public Word( String value, int type )
```

```
{
this.value = value;
this.type = type;
}

public Word( String value, int fre, int type )
{
this.value = value;
this.fre = fre;
this.type = type;
}

/**
 * @see IWord#getValue()
 */
@Override
public String getValue()
{
return value;
}

/**
 * @see IWord#getLength()
 */
@Override
public int getLength()
{
return value.length();
}

/**
 * @see IWord#getFrequency()
 */
@Override
public int getFrequency()
{
return fre;
}

/**
 * @see IWord#getType()
 */
```

```
@Override
public int getType()
{
return type;
}

/**
 * @see IWord#setPosition(int)
 */
@Override
public void setPosition( int pos )
{
position = pos;
}

/**
 * @see IWord#getPosition()
 */
public int getPosition()
{
return position;
}

/**
 * @see IWord#getPinying()
 */
@Override
public String getPinyin()
{
return pinyin;
}

/**
 * @see IWord#getSyn()
 */
@Override
public String[] getSyn()
{
return syn;
}

@Override
```

```
public void setSyn(String[] syn)
{
this.syn = syn;
}

/**
 * @see IWord#getPartSpeech()
 */
@Override
public String[] getPartSpeech()
{
return partspeech;
}

@Override
public void setPartSpeech(String[] partspeech)
{
this.partspeech = partspeech;
}

/**
 * @see IWord#setPinying(String)
 */
public void setPinyin( String py )
{
pinyin = py;
}

/**
 * @see IWord#addPartSpeech( String );
 */
@Override
public void addPartSpeech( String ps )
{
if ( partspeech == null ) {
partspeech = new String[1];
partspeech[0] = ps;
} else {
String[] bak = partspeech;
partspeech = new String[partspeech.length + 1];
int j;
for ( j = 0; j < bak.length; j++ )
```

```
partspeech[j] = bak[j];
partspeech[j] = ps;
bak = null;
}
}

/**
 * @see IWord#addSyn(String)
 */
@Override
public void addSyn( String s )
{
if ( syn == null ) {
syn = new String[1];
syn[0] = s;
} else {
String[] tycA = syn;
syn = new String[syn.length + 1];
int j;
for ( j = 0; j < tycA.length; j++ )
syn[j] = tycA[j];
syn[j] = s;
tycA = null;
}
}

/**
 * @see Object#equals(Object)
 * @see IWord#equals(Object)
 */
public boolean equals( Object o )
{
if ( this == o ) return true;

if ( o instanceof IWord )
{
IWord word = (IWord) o;
boolean bool = word.getValue().equalsIgnoreCase(this.getValue());
/*
 * value equals and the type of the word must
 * be equals too, for there is many words in
 * different lexicon with a same value but
```

```
 * in different use.
 */
return (bool && (word.getType() == this.getType()));
}

return false;
}

/**
 * Interface to clone the current object
 *
 * @return IWord
 */
@Override
public IWord clone()
{
IWord w = null;
try {
w = (IWord) super.clone();
} catch (CloneNotSupportedException e) {
e.printStackTrace();
}

return w;
}

/**
 * @see Object#toString()
 */
public String toString()
{
StringBuilder sb = new StringBuilder();
sb.append(value);
sb.append('/');
//append the cx
if ( partspeech != null ) {
for ( int j = 0; j < partspeech.length; j++ ) {
if ( j == 0 ) sb.append(partspeech[j]);
else {
sb.append(',');
sb.append(partspeech[j]);
}
```

```
}
} else
sb.append("null");
sb.append('/');
sb.append(pinyin);
sb.append('/');
//append the tyc
if ( syn != null ) {
for ( int j = 0; j < syn.length; j++ ) {
if ( j == 0 ) sb.append(syn[j]);
else {
sb.append(',');
sb.append(syn[j]);
}
}
} else
sb.append("null");

if ( value.length() == 1 ) {
sb.append('/');
sb.append(fre);
}

return sb.toString();
}
}
```

47:F:\git\java\search\jcseg-1.9.5\jcseg-core\src\test\java\org\lionsoul\jcseg\AppTest.java

```
package org.lionsoul.jcseg;

import junit.framework.Test;
import junit.framework.TestCase;
import junit.framework.TestSuite;

/**
 * Unit test for simple App.
 */
public class AppTest
    extends TestCase
{
    /**
     * Create the test case
```

```
     *
     * @param testName name of the test case
     */
    public AppTest( String testName )
    {
        super( testName );
    }

    /**
     * @return the suite of tests being tested
     */
    public static Test suite()
    {
        return new TestSuite( AppTest.class );
    }

    /**
     * Rigourous Test :-)
     */
    public void testApp()
    {
        assertTrue( true );
    }
}

48:F:\git\java\search\jcseg-1.9.5\jcseg-core\target\classes\META-
INF\maven\org.lionsoul.jcseg\jcseg-core\pom.xml
<?xml version="1.0"?>
<project xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd" xmlns="http://maven.apache.org/POM/4.0.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <modelVersion>4.0.0</modelVersion>

  <parent>
    <groupId>org.lionsoul.jcseg</groupId>
    <artifactId>jcseg</artifactId>
    <version>${jcseg.version}</version>
  </parent>

  <artifactId>jcseg-core</artifactId>
  <name>jcseg-core</name>
  <url>http://code.google.com/p/jcseg</url>
```

```
  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <jcseg.version>1.9.5</jcseg.version>
    <maven.test.skip>true</maven.test.skip>
<maven.javadoc.skip>true</maven.javadoc.skip>
  </properties>




  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>3.8.1</version>
      <scope>test</scope>
    </dependency>
  </dependencies>

  <build>
<plugins>
<plugin>
<artifactId>maven-resources-plugin</artifactId>
<version>2.5</version>
<executions>
<execution>
  <id>properties-file-copy</id>
  <phase>generate-resources</phase>
  <goals>
<goal>copy-resources</goal>
  </goals>
  <configuration>
  <outputDirectory>${basedir}/target/classes</outputDirectory>
  <resources>
 <resource>
<directory>../</directory>
<includes>
<include>jcseg.properties</include>
</includes>
<filtering>true</filtering>
```

```
 </resource>
   </resources>
 </configuration>
</execution>
</executions>
</plugin>

<plugin>
<groupId>org.apache.maven.plugins</groupId>
<artifactId>maven-source-plugin</artifactId>
<version>2.1.2</version>
<executions>
  <execution>
<id>attach-sources</id>
<phase>package</phase>
<goals>
  <goal>jar</goal>
</goals>
  </execution>
</executions>
  </plugin>

  <plugin>
<groupId>org.apache.maven.plugins</groupId>
<artifactId>maven-javadoc-plugin</artifactId>
<version>2.7</version>
<executions>
  <execution>
<id>attach-javadocs</id>
  <goals>
<goal>jar</goal>
  </goals>
  </execution>
</executions>
   </plugin>

  <plugin>
<groupId>org.apache.maven.plugins</groupId>
<artifactId>maven-shade-plugin</artifactId>
<version>1.4</version>
<executions>
  <execution>
```

```
<phase>package</phase>
<goals>
  <goal>shade</goal>
</goals>
<configuration>
  <transformers>
<transformer
implementation="org.apache.maven.plugins.shade.resource.ManifestResourceTransformer">
  <mainClass>org.lionsoul.jcseg.test.JcsegTest</mainClass>
</transformer>
  </transformers>
</configuration>
  </execution>
</executions>
  </plugin>
</plugins>
  </build>
</project>
```

49:F:\git\java\search\jcseg-1.9.5\jcseg-core\target\javadoc-bundle-options\javadoc-options-javadoc-resources.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<javadocOptions>
  <docletArtifacts>
    <docletArtifact />
  </docletArtifacts>
  <tagletArtifacts>
    <tagletArtifact />
  </tagletArtifacts>
  <javadocResourcesDirectory>src/main/javadoc</javadocResourcesDirectory>
</javadocOptions>
```