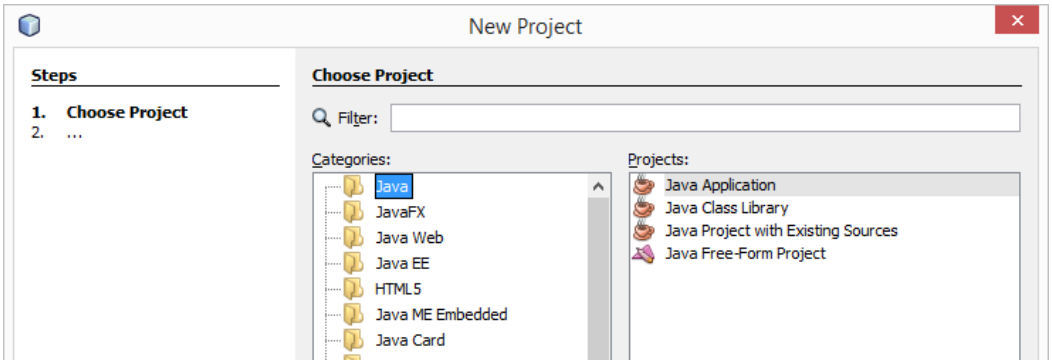
Hướng dẫn bài tập thực hành
Tiền xử lý văn bản

1. Tạo project

Trong Netbeans, chọn File -> New project, tạo Project loại Java Application



2. Cho người dùng nhập đường dẫn thư mục đầu vào và đầu ra

Đọc dữ liệu do người dùng nhập bằng lớp Scanner

```
import java.util.Scanner;

public class Lab01 {
   public static void main(String[] args) {
        /* Cho người dùng nhập thư mục đầu vào */
        // Khai báo phương tiện đọc input từ console
        Scanner in = new Scanner(System.in);

        // Biến input_folder giữ tên thư mục đầu vào do người dùng nhập
        System.out.println("Chọn thư mục đầu vào:");
        String input_folder = in.nextLine();

        // Biến output_folder giữ tên thư mục đầu vào do người dùng nhập
        System.out.println("Chọn thư mục đầu ra:");
        String output_folder = in.nextLine();
   }
}
```

Kiểm tra thư mục do người dùng nhập có tồn tại hay không bằng lớp Files

```
import java.util.Scanner;
import java.nio.file.*;

public class Lab01 {
   public static void main(String[] args) {
        /* Cho người dùng nhập thư mục đầu vào */
        ...
```

```
        /* Kiểm tra thư mục tồn tại không, nếu có xử lý tiếp, nếu không báo lỗi */
        if (!Files.exists(Paths.get(input_folder)) ||
!Files.exists(Paths.get(output_folder)))
        {
            System.out.println("Thư mục không tồn tại");
            return;
        }
    }
}
```

3. Duyệt các thư mục con trong thư mục đầu vào để chọn ra tập tin *.txt

Duyệt các thư mục con trong thư mục đầu vào bằng lớp File và thủ tục đệ quy *duyetThuMuc*

Hàm *xulyMotVanBan* sẽ là hàm chính thực hiện mọi thao tác tiền xử lý

```
import java.util.Scanner;
import java.nio.file.*;
import java.io.File;

public class Lab01 {
   public static void duyetThuMuc(File[] files, String output_folder) {
        for (File file : files) {
            // Nếu là thư mục thì gọi đệ quy tiếp
            if (file.isDirectory()) {
                duyetThuMuc(file.listFiles(), output_folder);
            } else {
                // Nếu là tập tin *.txt thì xử lý
                if (file.getName().endsWith(".txt")) {
                     String output_filepath = output_folder + file.getName();
                   output_filepath = output_filepath.replace(".txt", "_word.txt");
                   xulyMotVanBan(file.getAbsolutePath(), output_folder);
                }
            }
        }
    }
    public static void main(String[] args) {
        /* Cho người dùng nhập thư mục đầu vào */
        ...

         /* Kiểm tra thư mục tồn tại không, nếu có xử lý tiếp, nếu không báo lỗi */
        ...

         /* Duyệt thư mục đầu vào và tiền xử lý file txt */
        File[] files = new File(input_folder).listFiles();
        duyetThuMuc(files, output_folder);
    }
}
```

4. Rút trích nội dung của từ khóa Title và Abstract

Đọc file văn bản bằng lớp BufferedReader và FileReader, nhớ là có mở file thì phải đóng file.

Dùng một vòng lặp để đọc từng dòng văn bản vào biến tạm String *line* cho đến khi hết file.

Mỗi lần lặp, kiểm tra xem *line* có bắt đầu bằng từ khóa cần kiếm hay không.

Qui ước:

- Title chỉ gồm 1 dòng duy nhất
- Abstract có thể chứa nhiều dòng hoặc "Not Available"

```
...
import java.io.BufferedReader;
import java.io.FileReader;

public class Lab01 {
     public static void xulyMotVanBan(String input_filepath, String
output_filepath) {

        /* Đọc file và tách nội dung Title và Abstract */
        // Khai báo chuỗi lưu nội dung Title và Abstract
        String sTitle = "", sAbstract = "";

        // Khai báo biến tạm để biết đã tìm thấy Title hay Abstract chưa
        boolean bTitleFound = false, bAbstractFound = false;

        // Đọc file
        try {
            BufferedReader reader = new BufferedReader(new
FileReader(input_filepath));
            String line;
            while ((line = reader.readLine()) != null) {
                // Tìm abstract
                if (bAbstractFound == false) {
                    if (line.startsWith("Abstract") == true) {
                        bAbstractFound = true;
                        sAbstract = line.substring(line.indexOf(":") + 1);
                    }
                } else {
                    // Vì abstract có nhiều dòng
                    sAbstract += line;
                }

                // TỰ LÀM: Tìm title tương tự như abstract nhưng không cần phần
             "else" vì title chỉ có một dòng
                if (... && ...) {
                    ...
                }
            }
            reader.close();

            // Abstract và title chứa nhiều khoảng trắng, tab, phải loại bỏ
            sTitle = sTitle.trim().replaceAll("\\s+", " ");
            sAbstract = sAbstract.trim().replaceAll("\\s+", " ");

            // Đóng file
            reader.close();

        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    ...
}
```

5. Tách văn bản thành câu, tách câu thành từ, loại bỏ dấu chấm câu

Tự cài đặt hàm để tách văn bản thành câu, tách câu thành từ theo ví dụ 1 và 2 trong Hướng dẫn sử dụng thư viện OpenNLP, nhớ import thư viện tương ứng vào.

```
...
import java.util.ArrayList;

public class Lab01 {
     public static void xulyMotVanBan(String input_filepath, String
output_filepath) {

        /* Đọc file và tách nội dung Title và Abstract */
        ...

        /* Các thao tác tiền xử lý dữ liệu */
        // Khai báo ArrayList chứa các token sau khi đã tiền xử lý
        ArrayList<String> finalTokens = new ArrayList<String>();

        // Chuỗi các loại dấu câu cơ bản, kiểm tra token có thuộc chuỗi này không
        String punctuations = ",.:;?![]()'";

        // xử lý title (nếu tìm thấy)
        if (bTitleFound == true) {
            String tokens[] = tachCau(sTitle); // tự cài tachcau theo hướng dẫn
            for (String token : tokens) {
                // Nếu token thuộc chuỗi punctuations => là dấu câu => bỏ
                if (!punctuations.contains(token)) {
                    finalTokens.add(token);
                }
            }
        }

        // TỰ LÀM: xử lý abstract (nếu tìm thấy)
        if (bAbstractFound == true && sAbstract.startsWith("Not Available") ==
false) {
            ...
        }
   }

   ...
}
```

6. Loại bỏ stopword, stemming

TỰ LÀM.

Gợi ý:

- Bỏ stopword: Đọc mỗi từ trong file EnglishSL.txt vào cấu trúc dữ liệu ArrayList<String>. Kiểm tra token có thuộc ArrayList này không (dùng hàm *contains*). Lưu ý vấn đề chữ viết hoa – viết thường
- Stemming: cài đặt hàm theo ví dụ 3 trong Hướng dẫn sử dụng thư viện OpenNLP

7. Loại bỏ rác trong Abstract

TỰ LÀM.

8. Ghi kết quả xử lý vào file

```
...
import java.io.BufferedWriter;
import java.io.FileWriter;

public class Lab01 {
     public static void xulyMotVanBan(String input_filepath, String
output_filepath) {

        /* Đọc file và tách nội dung Title và Abstract */
        ...

        /* Các thao tác tiền xử lý dữ liệu */
        ...

        /* Ghi danh sách token đã xử lý ra file output */
        try (
            BufferedWriter bw = new BufferedWriter(new
FileWriter(output_filepath))) {
            for (String token : finalTokens) {
                bw.write(token);
                bw.newLine();
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
   }

   ...
}
```