Web Crawler

Project Overview

Application will take four value as a input from user

1. URL to a web page
2. Word to search
3. Limit to number pages search
4. File name

“This Application will go to that web page . it will count of the user given words on the page and it will collect all of the URLs on the page. After complete one page this system will go to the next page and it will repeat same task till user given limit .
After completed process this Application will write all console information into json file”

Day 1

Task Of Day

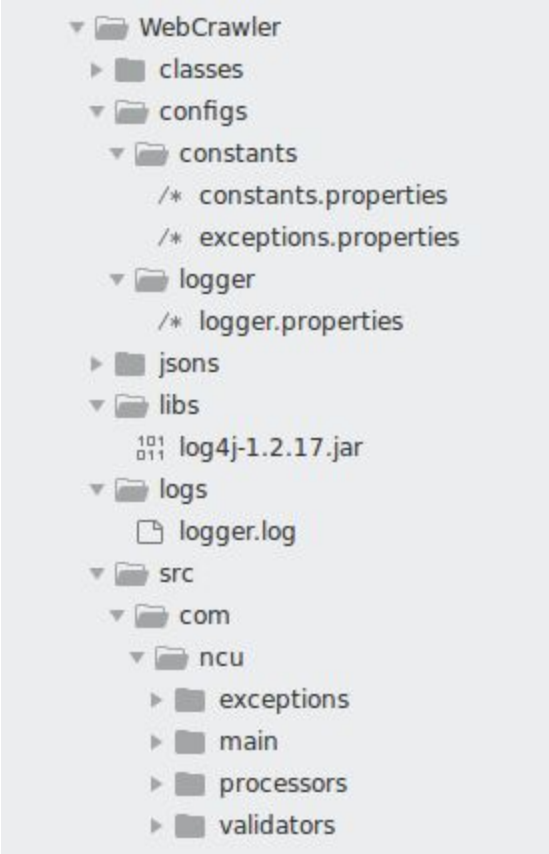
I. Read all Requirement of crawler Project .
II. Create Directory Structure of Project .
III. Introduction of logger.
IV. Implementation of logger In Project .
V. Implementation of constants.properties and exceptions.properties in Project .
VI. Implementation of main class in Project .
VII. Implementation of java file to take user inputs
URL to a web page
Search Words which user want to search on page
Limit of search for Project

Story 1 - Read all requirement of Application

- Retrieve a web page (we'll call it a document) from a website
- Collect all the links on that document
- Count the occurrences of user given word on document
- Visit the next link

  But what if we start at Page A and find that it contains links to Page B and Page C. That's fine, we'll go to Page B next if we don't find the word we're looking for on Page A. But what if Page B contains a bunch more links to other pages, and one of those pages links back to Page A? So let's add a few more things our crawler needs to do:
- Keep track of pages that we've already visited
- Put a limit on the number of pages to search so the system should not run for infinite time .
- After completed whole process of application . it should display process time
- Write console information into file

Story 2 - Create Directory Structure Of Application



Task 1 - Read description of Directory Structure

I. "**src/com/ncu"** directory to contains whole source code of this application .directory to contains whole source code of this application . but we will separate this project into different section inside this directory . it have four folders.

   A. **exceptions** folder to contain all Custom exceptions file .

   B. **main** folder to contain Crawler.java class which is our starting point of Application

   C. **Processors** folder to implements logic of application

   D. **Validators** folder to validate user given input

II. **libs** folder to contain all external jar file which project should required

III. **logs** folder to store output of logger

IV. **Configs** folder contains two folders

constants folder to store all constants value and custom exception which used in this Application .

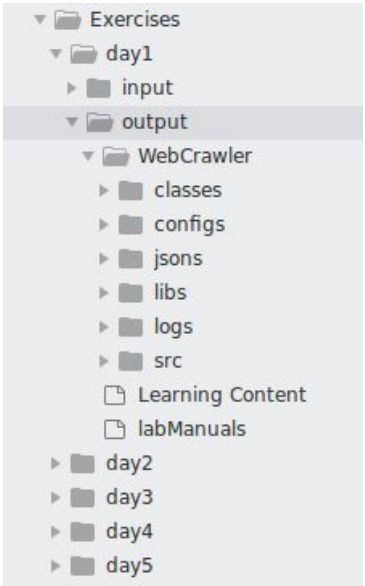
Logger folder to store logger properties file . which use to configure Logger .

V. **Classes** folder to store all . classes file of java files

VI. **Exercises** This folder we will use to make whole application within 5 days

Note - "This whole project we will complete within 5 days so create a day 1 folder inside Exercises folder "

Task 2 - Directory Structure of Day 1



Task 3 - Read Description of Input/Output folder

I. Input folder will contain previous day output
II. Output folder will contain current day task and codes

Story 3 - Introduction of logger

Task 1- Read about logger

If we use SOP (System.out.print()) statements to print log messages, then we can run into some disadvantages:

1. We can print log messages on the console only. So, when the console is closed, we will lose all of those logs.
2. We can't store log messages in any permanent place. These messages will print one by one on the console because it is a single-threaded environment.

To overcome these problems, the Log4j framework came into the picture. Log4j is an open source framework provided by Apache for Java projects.

Task 2- Read about Log4j Components

Log4j has three main components, which are the following:

1. Logger
2. Appender
3. Layout

Task 3 - Read about Logger Object

Logger is a class in the *org.apache.log4j.** package. We have to initialize one Logger object for each Java class. We use Logger's methods to generate log statements. Log4j provides the factory method to get Logger objects.
Syntax to get Logger objects:

```
static Logger logger=Logger.getLogger(CurrentClass.class.getName()).
```

Note: *CurrentClass* is a Java class name for which we are getting logger object.

Example -

```
public class Student{
   private static final Logger LOGGER = Logger.getLogger(Student.class);
   public void getStudentRecord() {
   }
}
```

The Logger class has some methods that are used to print application status. We have five methods in the Logger class

1. info()
2. debug()
3. warn()
4. error()

How and when to use these methods depends on us. Here, the method names are different, but the process is the same for all of them: all will print a message only.

Task 4 - Read about Log4j- Configuration

log4j.properties -

```
# Root logger option
log4j.rootLogger=INFO, file, stdout
# configuration to print into file
log4j.appender.file=org.apache.log4j.RollingFileAppender
log4j.appender.file.File=D:\\log\\logging.log
log4j.appender.file.MaxFileSize=12MB
log4j.appender.file.MaxBackupIndex=10
log4j.appender.file.layout=org.apache.log4j.PatternLayout
log4j.appender.file.layout.ConversionPattern=%d{yyyy-MM-dd HH:mm:ss} %-5p %c{1}:%L
- %m%n
# configuration to print on console
log4j.appender.stdout=org.apache.log4j.ConsoleAppender
log4j.appender.stdout.Target=System.out
log4j.appender.stdout.layout=org.apache.log4j.PatternLayout
log4j.appender.stdout.layout.ConversionPattern=%d{yyyy-MM-dd HH:mm:ss} %-5p
%c{1}:%L - %m%n
```

Description of log4j.properties file

- *log4j.appender.***file**=*org.apache.log4j.***RollingFileAppender**
- log4j.appender.**stdout**=org.apache.log4j.**ConsoleAppender**

These will define appender types: That means they will specify where we want to store application logs. RollingFileAppender will print all logs in a file, and ConsoleAppender will print all logs in the console.

- log4j.appender.file.File=**D:\\log\\logging.log**

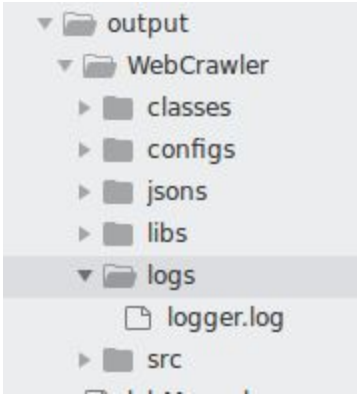
That specifies the log file location.

- log4j.appender.**file.layout**=org.apache.log4j.PatternLayout
- log4j.appender.**file.layout**.ConversionPattern=%d{yyyy-MM-dd HH:mm:ss} %-5p %c{1}:%L - %m%n

Story 4 - Implementation of logger in Crawler

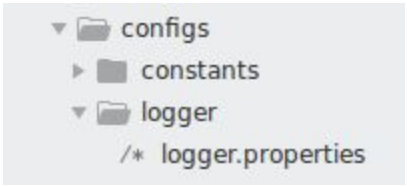
Task 1 - Create log file

1- Create logger.log file into logs folder to store log output



Task 2 - Create logger properties file

1- Create logger.properties file into logget folder of configs folder . this file user to configure logger into your application



2- Append below code into "logger.properties" file

2.1 - Log4j ConsoleAppender – Logging to console

```
# Root logger option
log4j.rootLogger=DEBUG, stdout ,file

# Redirect log messages to console
log4j.appender.stdout=org.apache.log4j.ConsoleAppender
log4j.appender.stdout.Target=System.out
log4j.appender.stdout.layout=org.apache.log4j.PatternLayout
log4j.appender.stdout.layout.ConversionPattern=%d{yyyy-MM-dd HH:mm:ss} %-5p
%c{1}:%L - %m%n
```

2.1 -Log4j RollingFileAppender – Logging to file

```
# Redirect log messages to a log file, support file rolling.
log4j.appender.file=org.apache.log4j.RollingFileAppender
log4j.appender.file.File=/home/kls103/Desktop/NCU/Exercises/day1/output/Web
Crawler/logs/logger.log
log4j.appender.file.layout=org.apache.log4j.PatternLayout
log4j.appender.file.layout.ConversionPattern=%d{yyyy-MM-dd HH:mm:ss} %-5p
%c{1}:%L - %m%n
Copy
```

Note -

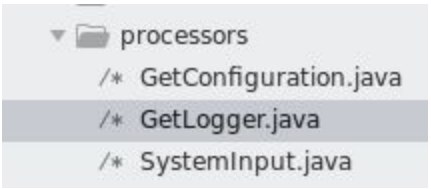
“ Change file (log4j.appender.file.File) according to your system path”

```
log4j.appender.file.File=/home/kls103/Desktop/NCU/Exercises/day1/output/WebCr
awler/logs/logger.log
```

Task 3 - Configure logger properties file into project

Description - “we will not write logger properties configuration code into each file of application . instead of this i will create one class with method which will have code of logger properties configuration and it i will return object of logger . so wheneven we will require of logger object then simply we will call this method”

1- create GetLogger java file into “processors” folder



2- create GetLogger class with “loggerValue” method into GetLogger java file which return logger object into

```
public class GetLogger
{
// loggerValue method initialization logger and return one logger
object
      public Logger loggerValue(String className)
      {
       }
}
```

3- configure “logger.properties” file and return object of logger

```
public Logger loggerValue(String className)
      {
String log4jConfigFile = System.getProperty("user.dir")+
File.separator + "configs/logger/logger.properties";
Logger logger = Logger.getLogger(className+".class");
PropertyConfigurator.configure(log4jConfigFile);
  return logger;
      }
```

4- full source code of “GetLogger.java” file . add below codes into your project

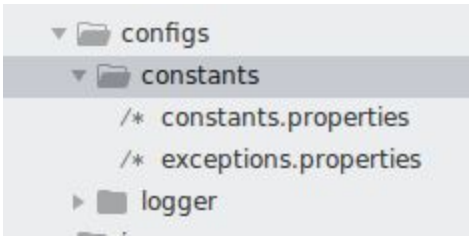
```
package com.ncu.processors;
import java.io.File;
import org.apache.log4j.BasicConfigurator;
import org.apache.log4j.Logger;
import org.apache.log4j.PropertyConfigurator;

public class GetLogger
{

      public Logger loggerValue(String className)
      {
String log4jConfigFile = System.getProperty("user.dir")+
File.separator + "configs/logger/logger.properties";
Logger logger = Logger.getLogger(className+".class");
PropertyConfigurator.configure(log4jConfigFile);
    return logger;
      }

}
```

Story 5 - Implementation of constants folder

Task 1 - Create constants folder

1- Create constants folder inside configs folder and this folder will contain two file

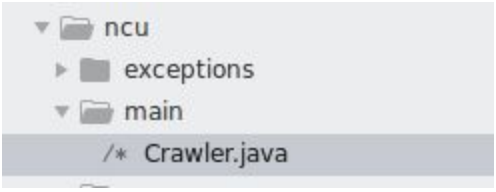
I. **constants.properties** - " To store all constants value which we will use in this application "
II. **exceptions.properties** - " To store all custom exception which we will use to store custom exception "



Story 6 - Implementation of main folder In our Application

Task 1 - Create Crawler.java file

1- Create Crawler.java file into main folder with class



2- Create “Crawler” class with main method into Crawler.java file
3- Create package “com.ncu.main” to store .class file in manner of source code structure

```
package com.ncu.main;

public class Crawler
{
// This main method will start all processes from here .
     public static void main(String[] args)
      {
       }
}
```

4- import com.ncu.processors.*;
“I am importing processors which will have some java file in future tutorials for taking input and configuration of logger and configs”

5- create GetLogger class object and call loggerValue method to get logger object into this file

```
public static void main(String[] args)
     {
 // Installation of logger object
GetLogger loggerObject=new GetLogger();
Logger logger=loggerObject.loggerValue("Crawler");
      }
```

6- create SystemInput class object and call inputUrl method of SystemInput class

Note - "System Input" class we will implement into further session to take inputs from user . input Url method we will use to take url as a input

```
 // Creating object of SystemInput to take inputs from user.
    SystemInput inputObject=new SystemInput();
    inputObject.inputUrl();
```

6- full source code of "Crawler.java" file . add below code into your file

```
package com.ncu.main;
import com.ncu.processors.*;
import org.apache.log4j.Logger;

public class Crawler
{
// This main method will start all processes from here .
public static void main(String[] args)
{
// Installation of logger object
GetLogger loggerObject=new GetLogger();
Logger logger=loggerObject.loggerValue("Crawler");
logger.info("=======================================");
logger.info("||        Welcome NCU Students       ||");
logger.info("=======================================");
```

```
// Creating object of SystemInput to take inputs from user.
SystemInput inputObject=new SystemInput();
inputObject.inputUrl();
}
}
```

Story 6.1// configConstants method initialization Properties object and retrun one Properties class object

public Properties configConstants()

{

Properties propertieValue=null;

try{

Properties propertieObj = new Properties();

String configPath = System.getProperty("user.dir")+ File.separator + "configs/constants/constants.properties";

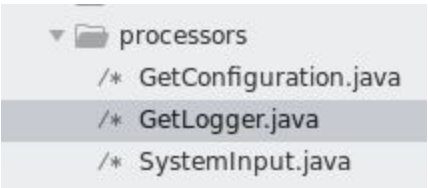
InputStream input = new FileInputStream(configPath);

```
            propertieObj.load(input);

            propertieValue=propertieObj;

        }catch(Exception e)

        {

            e.printStackTrace();

        }

        return propertieValue;

    }// configConstants method initialization Properties object
and retrun one Properties class object

    public Properties configConstants()

    {

        Properties propertieValue=null;
```

```
        try{

            Properties propertieObj = new Properties();

            String configPath =
System.getProperty("user.dir")+ File.separator +
"configs/constants/constants.properties";

            InputStream input = new
FileInputStream(configPath);

            propertieObj.load(input);

            propertieValue=propertieObj;

        }catch(Exception e)

        {

            e.printStackTrace();

        }

        return propertieValue;
```

} - Implementation of “GetConfiguration.java” .

1- Create GetConfiguration.java file into processors folder to take input from user



Define - “create GetConfiguration.java file to configure constants for get configs value and messages”

2- create “GetConfiguration” class into GetConfiguration.java file

```
package com.ncu.processors;
import java.io.File;
import java.util.Properties;
import java.io.FileInputStream;
import java.io.InputStream;
public class GetConfiguration
{
}
```

3- create “configMessages” method inside GetConfiguration class .

Define - “use this method to configure “exceptions.properties” file to get messages”

```
public Properties configMessages()
  {
     Properties propertieValue=null;
      try{
       Properties propertieObj = new Properties();
String configPath = System.getProperty("user.dir")+ File.separator +
"configs/constants/exceptions.properties";
       InputStream input = new FileInputStream(configPath);
       propertieObj.load(input);
       propertieValue=propertieObj;
            }catch(Exception e)
            {
                  e.printStackTrace();
            }
            return propertieValue;
      }
```

3- create “configConstants” method inside GetConfiguration class .
Define - “use this method to configure “constants.properties” file to get constant values ”

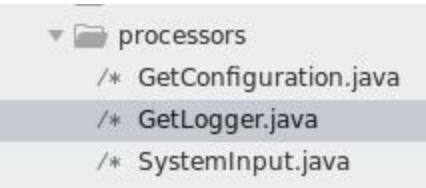
```
// configConstants method initialization Properties object and retrn
one Properties class object
      public Properties configConstants()
      {
            Properties propertieValue=null;
            try{
                  Properties propertieObj = new Properties();
                  String configPath = System.getProperty("user.dir")+
File.separator + "configs/constants/constants.properties";
                  InputStream input = new FileInputStream(configPath);
                  propertieObj.load(input);
                  propertieValue=propertieObj;
            }catch(Exception e)
            {
                  e.printStackTrace();
            }
```

```
            return propertieValue;
      }
```

Story 7 - Create SystemInput.java file to take input from user .

Task 1 - Create SystemInput.java file

1- Create SystemInput.java file into processors folder to take input from user



2- Create “SystemInput” class into SystemInput.java

```
public class SystemInput
    {
    }
```

3- Make constructor to initialization of logger and constants file object creation time

```
public class SystemInput
{
  //Creating constructor to Initialization Logger and Configs
  public SystemInput()
  {
  }
}
```

4- Initialization of logger and Properties files to get objects which we will use in further sessions

```
public class SystemInput
{
  //Creating constructor to Initialization Logger and Configs
      public SystemInput()
      {
            // initialization of GetLogger to get logger object
            GetLogger loggerObject=new GetLogger();
            Logger logger=loggerObject.loggerValue("SystemInput");
            this.logger=logger;

            // calling configConstants method to get constant values
            Properties constants=propertyObject.configConstants();
            this.constants=constants;
      }
}
```

Story 8 - take Url as a input from user into SystemInput file .

Task 1 - take url as a input from user .

1- create inputUrl method to take Url as String input from user into SystemInput java file

```
public void inputUrl()
  {
  }
```

2- create Scanner class object and take string as a input

```
/* This method will take one url from user and it will called to
  UrlValidator to Check Validation of Url */
  public void inputUrl()
  {
// Creating object of current class object to use in this method
only.
```

```
     SystemInput systemInputObj=new SystemInput();
    // Taking Url as a input from user.
     System.out.println("\n");
     logger.info("Please Enter Your Url :-");
     Scanner scan = new Scanner(System.in);
     String urlName = scan.nextLine();
  /* Calling searchValue only for Testing Purpose this line will
delete further session  */
     systemInputObj.searchValue(urlName);

  }
```

Note -

systemInputObj.searchValue(urlName) line is only for Testing Purpose it will delete further session

Story 8 - take Search Word as a input from user .

Task 1 - take Search as a input from user .

1- create searchValue() method to take Search word as a String input from user into SystemInput java file . and this method will take one string value (url) .

```
public void searchValue(String urlName)  {
  }
```

2- create Scanner class object and take string as a input

```
// This method will take Search Word from user , which user want to
search on url
```

```
  public void searchValue(String urlName)
  {
// Creating object of current class object to use in this method
only.
     SystemInput systemInputObj=new SystemInput();
    // Taking Url as a input from user.
     System.out.println("\n");
     logger.info("Enter Your String Which You Want to Search :- ");
     Scanner scan = new Scanner(System.in);
     String searchValue= scan.nextLine();
/* Calling searchValue only for Testing Purpose this line will delete
further session  */
    systemInputObj.numberOfPages(urlName,searchValue);
  }
```

Note -

systemInputObj.numberOfPages(urlName,searchValue) line is only for Testing Purpose it will delete further session

Story 9 - take number of pages user want to Search .

Task 1 - take number of pages from user .

1- create numberOfPages() method to take number of pages as integer input from user into SystemInput java file . and this method will take two string value (url) and search word .

```
public void numberOfPages(String urlName,String searchValue)
  {
  }
```

2- create Scanner class object and take integer as a input

```
public void numberOfPages(String urlName,String searchValue)
 {
// Creating object of current class object to use in this method
only.
     SystemInput systemInputObj=new SystemInput();
 // Taking number of pages as a input from user.
      System.out.println("\n");
      logger.info("How Many Pages Do You Want To Search :- ");
      Scanner scan = new Scanner(System.in);
       int numberOfPage= scan.nextInt();

/* Calling searchValue only for Testing Purpose this line will delete
further session  */
    systemInputObj.writeOutputInFile();
  }
}
```

Note -

systemInputObj.writeOutputInFile() line is only for Testing Purpose it will delete further session

Story 10 - take json file name to write console information into file

Task 1 - file name from user .

1- create writeOutputInFile() method to take file name as string input from user into SystemInput java file .

```
 public void writeOutputInFile()
  {
  }
```

2- create Scanner class object and take string as a input

```
// writeOutputInFile Method to write output into json file .// URLCrawler
urlCrawlerObj
  public void writeOutputInFile()
  {
// Creating object of current class object to use in this method only.
      SystemInput systemInputObj=new SystemInput();
    // Taking Url as a input from user.
      System.out.println("\n");
      logger.info("Please Enter Your Json File Name :- ");
      Scanner scan = new Scanner(System.in);
      String fileName= scan.nextLine();

/* Calling searchValue only for Testing Purpose this line will delete
further session  */
      systemInputObj.exits();
  }
```

Note -

systemInputObj.exits() line is only for Testing Purpose it will delete further session

Story 10 - Ask user to exits from system after complete process .

Task 1 - exits method into SystemInput file .

1- create exits() method to take permission from user to exit from application .

```
 public void exits()
  {
  }
```

2- create Scanner class object and take string as a input from user .

Description - “ application will ask to user for

```
// Implementation of Exit from System
public void exits(){
// Creating object of current class object to use in this method
only.
SystemInput systemInputObj=new SystemInput();
// Ask user to exit from system or Continue process.
  System.out.println("\n");
  logger.info(message.getProperty("exitMessage"));
   Scanner exitObject = new Scanner(System.in);
   String userPermisson= exitObject.nextLine();
   String setPermisson = constants.getProperty("endProcess");
      if(setPermisson.equalsIgnoreCase(userPermisson)){
            logger.info("Thanks for Using Our System .......!");
            System.exit(0);
      }else{
            // Continue our process Again.
            systemInputObj.inputUrl();
            }
      }
```

3- add below code into exceptions.properties of constants folder .

```
validIntegerInput="Oops.. Mismatch Input Not Please Give Value In
Integer format..!"
exitMessage="Do You Want To Exist From System... Please Write (exit)
or Any Other Key To Continue Process ... !";
```

4- add below code into constants.properties of constants folder .

```
endProcess=exit
```

Story 11 - Start taking input from user by calling inputUrl .

Task 1- call inputUrl method

1- call inputUrl() from main method of Crawler.java file to start taking value from user in application .

```
public class Crawler
{
 // This main method will start all processes from here .
     public static void main(String[] args)
     {
 // Creating object of SystemInput to take inputs from user.
     SystemInput inputObject=new SystemInput();
     inputObject.inputUrl();
     }
 }
```

Story 12 - Full Source code of SystemInput.java file

```
package com.ncu.processors;

import java.util.Scanner;
```

```java
import java.util.Properties;
import java.util.InputMismatchException;

import org.apache.log4j.Logger;
public class SystemInput
{
      Logger logger;
      Properties message,constants;

      //Creating constructor to Initialization Logger and Configs
      public SystemInput()
      {
            // initialization of GetLogger to get logger object
            GetLogger loggerObject=new GetLogger();
            Logger logger=loggerObject.loggerValue("SystemInput");
            this.logger=logger;

            // initialization of configMessage to get messages
            GetConfiguration propertyObject=new GetConfiguration();
            Properties message=propertyObject.configMessages();
            this.message=message;

            // calling configConstants method to get constant values
            Properties constants=propertyObject.configConstants();
            this.constants=constants;
      }

  /* This method will take one url from user and it will called to
  UrlValidator to Check validation of Url */
  public void inputUrl()
  {
// Creating object of current class object to use in this method
only.
      SystemInput systemInputObj=new SystemInput();
    // Taking Url as a input from user.
      System.out.println("\n");
      logger.info("Please Enter Your Url :-");
      Scanner scan = new Scanner(System.in);
```

```
      String urlName = scan.nextLine();

/* Calling searchValue only for Testing Purpose this line will delete
further session  */
    systemInputObj.searchValue(urlName);

  }
// This method will take Search Word from user , which user want to
search on url
  public void searchValue(String urlName)
  {
// Creating object of current class object to use in this method
only.
      SystemInput systemInputObj=new SystemInput();
      // Taking Url as a input from user.
      System.out.println("\n");
      logger.info("Enter Your String Which You Want to Search :- ");
      Scanner scan = new Scanner(System.in);
      String searchValue= scan.nextLine();
 /* Calling searchValue only for Testing Purpose this line will
delete further session  */
    systemInputObj.numberOfPages(urlName,searchValue);

  }


// This method will take how number of pages do you want to search.
  public void numberOfPages(String urlName,String searchValue)
  {
// Creating object of current class object to use in this method
only.
      SystemInput systemInputObj=new SystemInput();
// Using Try Catch if to catch exception if user give string value as
number of page to search
      try{
            // Taking number of pages as a input from user.
            System.out.println("\n");
            logger.info("How Many Pages Do You Want To Search :- ");
```

```
            Scanner scan = new Scanner(System.in);
            int numberOfPage= scan.nextInt();
/* Calling searchValue only for Testing Purpose this line will delete
further session  */
            systemInputObj.writeOutputInFile();

      }catch(InputMismatchException e){
  logger.error("\n"+e+message.getProperty("validIntegerInput")+"\n");
  systemInputObj.numberOfPages(urlName,searchValue);
      }
  }
// writeOutputInFile Method to write output into json file .//
URLCrawler urlCrawlerObj
  public void writeOutputInFile()
  {
// Creating object of current class object to use in this method
only.
      SystemInput systemInputObj=new SystemInput();
    // Taking Url as a input from user.
      System.out.println("\n");
      logger.info("Please Enter Your Json File Name :- ");
      Scanner scan = new Scanner(System.in);
      String fileName= scan.nextLine();
/* Calling searchValue only for Testing Purpose this line will delete
further session  */
     systemInputObj.exits();
      }
// Implementation of Exit from System
      public void exits(){
// Creating object of current class object to use in this method
only.
            SystemInput systemInputObj=new SystemInput();
     // Ask user to exit from system or Continue process.
            System.out.println("\n");
            logger.info(message.getProperty("exitMessage"));
            Scanner exitObject = new Scanner(System.in);
            String userPermisson= exitObject.nextLine();
            String setPermisson = constants.getProperty("endProcess");
```

```
            if(setPermisson.equalsIgnoreCase(userPermisson)){
             logger.info("Thanks for Using Our System .......!");
             System.exit(0);
            }else{
            // Continue our process Again.
              systemInputObj.inputUrl();
            }
      }  }
```

Story 13 - Full Source code of Crawler.java file

```
package com.ncu.main;
import com.ncu.processors.*;
import org.apache.log4j.Logger;
public class Crawler
{
  // This main method will start all processes from here .
      public static void main(String[] args)
      {
     // Installation of logger object
            GetLogger loggerObject=new GetLogger();
            Logger logger=loggerObject.loggerValue("Crawler");

            logger.info("=======================================");
            logger.info("||        Welcome NCU Students       ||");
            logger.info("=======================================");

     // Creating object of SystemInput to take inputs from user.
            SystemInput inputObject=new SystemInput();
            inputObject.inputUrl();

      }
}
```

Story 13 - Full Source code of GetConfiguration.java file

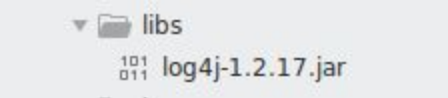
```
package com.ncu.processors;

import java.io.File;
import java.util.Properties;
import java.io.FileInputStream;
import java.io.InputStream;

public class GetConfiguration
{
// configMessages method initialization Properties object and retrun one
Properties class object
	public Properties configMessages()
	{
		Properties propertieValue=null;
		try{
			Properties propertieObj = new Properties();
			String configPath = System.getProperty("user.dir")+
File.separator + "configs/constants/exceptions.properties";
			InputStream input = new FileInputStream(configPath);
			propertieObj.load(input);
			propertieValue=propertieObj;
		}catch(Exception e)
		{
			e.printStackTrace();
		}
		return propertieValue;
	}

// configConstants method initialization Properties object and retrun one
Properties class object
	public Properties configConstants()
	{
		Properties propertieValue=null;
		try{
			Properties propertieObj = new Properties();
			String configPath = System.getProperty("user.dir")+
File.separator + "configs/constants/constants.properties";
			InputStream input = new FileInputStream(configPath);
			propertieObj.load(input);
			propertieValue=propertieObj;
		}catch(Exception e)
		{
```

```
                    e.printStackTrace();
                }
                return propertieValue;
        }

}
```

Story 12 - Compile and run your application of day 1

Task 1 - Compile code

2- add logger jar file into libs folder

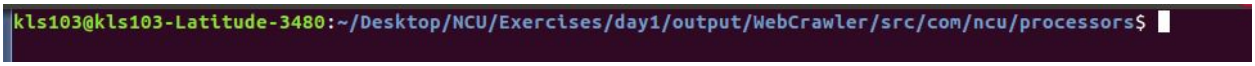


Note -

I. **All paths will be of your own computer path . this compile command will not run for your computer .**
II. **before compile GetLogger you must create classes folder**

2- compile processors java files

“Open terminal into processors folder of project”



3-compile GetLogger.java file of processors folder .

```
javac -cp
".:/home/kls103/Desktop/NCU/Exercises/day1/output/WebCrawler/libs/log4j-1.2.17.jar:
/home/kls103/Desktop/NCU/Exercises/day1/output/WebCrawler/classes" -d
"/home/kls103/Desktop/NCU/Exercises/day1/output/WebCrawler/classes"  GetLogger.java
```

Note -

“I am setting classpath of (libs) folder . bcz it have external jar files”

To download jar file of log plz follow below links -

“http://www.java2s.com/Code/Jar/l/Downloadlog4j1217jar.htm”

“And setting classpath of classes folder .bcz it have our own packages”

".:/home/kls103/Desktop/NCU/Exercises/day1/output/WebCrawler/libs/log4j-1.2.17.jar:/home/kls103/Desktop/NCU/Exercises/day1/output/WebCrawler/classes"

2- for send our compiled .class file into classes folder

-d "/home/kls103/Desktop/NCU/Exercises/day1/output/WebCrawler/classes"

5- compile GetConfiguration.java file of processors folder .

```
javac -cp
".:/home/kls103/Desktop/NCU/Exercises/day1/output/WebCrawler/libs/log4j-1.2.17.jar:
/home/kls103/Desktop/NCU/Exercises/day1/output/WebCrawler/classes" -d
"/home/kls103/Desktop/NCU/Exercises/day1/output/WebCrawler/classes"
GetConfiguration.java
```

6- compile SystemInput.java file of processors folder .

```
javac -cp
".:/home/kls103/Desktop/NCU/Exercises/day1/output/WebCrawler/libs/log4j-1.2.17.jar:
/home/kls103/Desktop/NCU/Exercises/day1/output/WebCrawler/classes" -d
"/home/kls103/Desktop/NCU/Exercises/day1/output/WebCrawler/classes"
SystemInput.java
```

7- compile Crawler.java file of main folder .

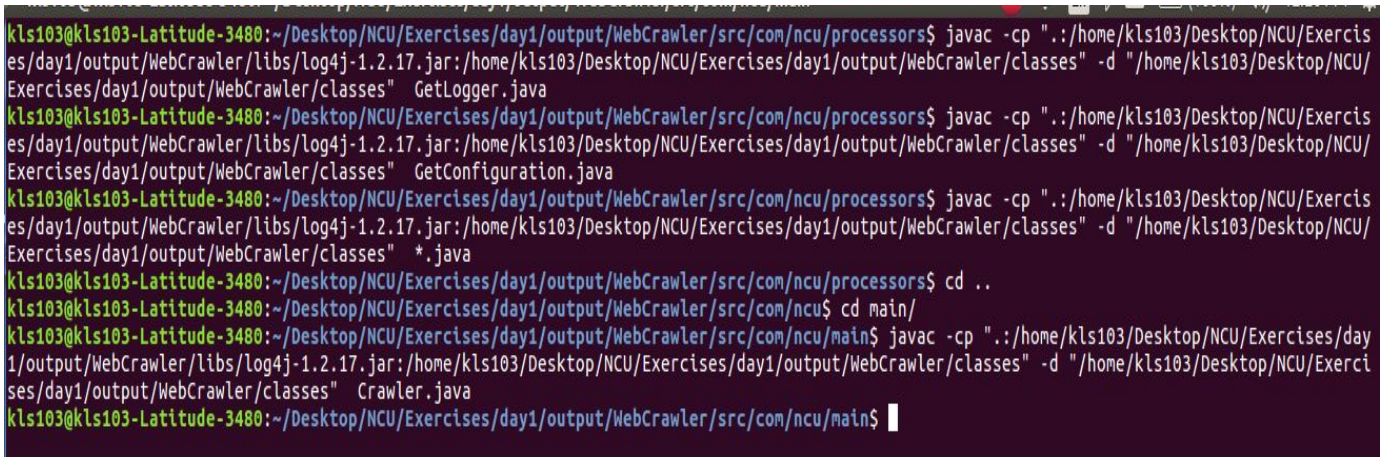
7.1 - open terminal into main folder

```
kls103@kls103-Latitude-3480:~/Desktop/NCU/Exercises/day1/output/WebCrawler/src/com/ncu/main$
```

7.2 - compile Crawler.java file of main folder

```
javac -cp
".:/home/kls103/Desktop/NCU/Exercises/day1/output/WebCrawler/libs/log4j-1.2.17.jar:
/home/kls103/Desktop/NCU/Exercises/day1/output/WebCrawler/classes" -d
"/home/kls103/Desktop/NCU/Exercises/day1/output/WebCrawler/classes"  Crawler.java
```

8- whole process of compile



Task 2 - Run code

1 - open terminal in root of application “ WebCrawler”

2- Run your application

```
java -cp
".:/home/kls103/Desktop/NCU/Exercises/day1/output/WebCrawler/libs/log4j-1.2.17.jar:
/home/kls103/Desktop/NCU/Exercises/day1/output/WebCrawler/classes"
com.ncu.main.Crawler
```

Note -

“Your should give package name at run time like -

“com.ncu.main” is package of Crawler .class file”

Story 13- Output of day 2