Day 2

In day 2 session we will apply validation on user given input value . In case of validation fail it will return false other then true .

Task Of Day

I. Implementation of validation on Input url
II. Implementation of validation on Input Search Word
III. Implementation of validation on Input Number of Pages

Story 1 - Implementation of validation on user given url

Task 1 - Url should not be blank

“ Create UrlValidator.java into validators folder to validate all validation On url”

1- Create UrlValidator class into UrlValidator.java file

```
package com.ncu.validators;
import com.ncu.exceptions.*;
import com.ncu.processors.GetLogger;
import com.ncu.processors.GetConfiguration;
import java.util.Properties;
import java.util.regex.Matcher;
import java.util.regex.Pattern;
import org.apache.log4j.Logger;
public class UrlValidator{
```

```
 Logger logger;
 Properties message,constants;
}
```

2- create constructor to initialization of logger , configMessage and configConstants files

```
public UrlValidator()
      {
      // initialization of GetLogger to get logger object
      GetLogger loggerObject=new GetLogger();
      Logger logger=loggerObject.loggerValue("UrlValidator");
      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;
}
```

3- create validator() method into UrlValidator.java file where we will validate all url validation .

Description - “this method will validate all validation of user given url . if all validation will true then this method will return true otherwise false”

```
public boolean validator(String urlName)
{
try{
    }catch(){
   }
}
```

4- create empltyUrlMethod() method into UrlValidator.java to validate user give url should not blank

```
/* Generate "EmptyUrlException" Exception if gives blank space as a
Url  */
private void empltyUrlMethod(String urlName) throws EmptyUrlException
{
  if (urlName == null || urlName.trim().isEmpty()) {
        throw new EmptyUrlException("");
      }
}
```

5- call this empltyUrlMethod() method from validator() with url value

```
  //Generate "EmptyUrlException" Exception if gives blank space as a
Url
       validatorObj.empltyUrlMethod(urlName);
```

6- catch EmptyUrlException generated by empltyUrlMethod() method into catch block

```
public boolean validator(String urlName)
{
   try{
      validatorObj.empltyUrlMethod(urlName);

      }
//All Exception will taken in this section
catch(EmptyUrlException e){
logger.error("\n"+e+message.getProperty("emptyUrlMessage")+"\n");
      return false;
      }
catch(Exception e){
logger.error("\n"+e+"\n");
      return false;
      }
   return true;
}
```

7- add custom message into “exceptions.properties” for “emptyUrlMessage”

```
 emptyUrlMessage="Oops.. Empty Url Is Not Acceptable ...!"
```

8- empltyUrlMethod() method throws EmptyUrlException so you have to create EmptyUrlException custom exception into exceptions folder .

8.1- create EmptyUrlException java file into “exceptions” folder

8.2- add below code into EmptyUrlException.java file

```
package com.ncu.exceptions;

public class EmptyUrlException extends Exception{
 public EmptyUrlException(String s){
  super(s);
 }
}
```

Task 2 - Url should follow “urlRegex”

1- add “Regex” value into constants.properties file of constants folder .

```
urlRegex =\\b(https?|ftp|file)://[-a-zA-Z0-9+&@#/%?=~_|!:,.;]*[-a-zA-Z0-9+&@#/%=~_|
```

2- Create invalidUrlMethod() method into UrlValidator.java to apply regex on url .

```
/* Generate "InvalidUrlException" Exception if user give wrong Url */
private void invalidUrlMethod(String urlName) throws
InvalidUrlException {
String regexValue = constants.getProperty("urlRegex");
Pattern patObject = Pattern.compile(regexValue);
Matcher matcher = patObject.matcher(urlName);
   boolean value = matcher.matches();
   if(!value)
   {
      throw new InvalidUrlException("");
   }
}
```

3- call this invalidUrlMethod() method from validator() with url value

```
//Generate "InvalidUrlException" Exception if user give wrong Url
   validatorObj.invalidUrlMethod(urlName);
```

4- catch InvalidUrlException generated by invalidUrlMethod() method into catch block

```
 public boolean validator(String urlName)
{
 try{
      UrlValidator validatorObj = new UrlValidator();
//Generate "InvalidUrlException" Exception if user give wrong Url
       validatorObj.invalidUrlMethod(urlName);
            }
//All Exception will taken in this section
catch(EmptyUrlException e){
logger.error("\n"+e+message.getProperty("emptyUrlMessage")+"\n");
       return false;
   }
catch(InvalidUrlException e){
this.logger.error("\n"+e+message.getProperty("invalidUrlMessage")+"\n
");
      return false;
            }
catch(Exception e){
     logger.error("\n"+e+"\n");
      return false;
      }
  return true;
}
```

5- add custom message into “exceptions.properties” for“invalidUrlMessage”

```
invalidUrlMessage="Oops.. Given Url Is Not Valid Plz Try With Other
Url ...!"
```

6- invalidUrlMethod() method throws InvalidUrlException so you should create InvalidUrlException exception into exceptions folder .

6.1 - create InvalidUrlException java file into “exceptions” folder

6.2- add below code into InvalidUrlException.java file

```
 package com.ncu.exceptions;
 public class InvalidUrlException extends Exception{
 public InvalidUrlException(String s){
  super(s);
 }
}
```

Task 3 - Call “validator()” method of UrlValidator class from SystemInput.java

1- after take (Web URL) as a input into inputUrl method of SystemInput.java file . call validator method of UrlValidator.java file to apply validation
2- add below code into “SystemInput.java” file of processors

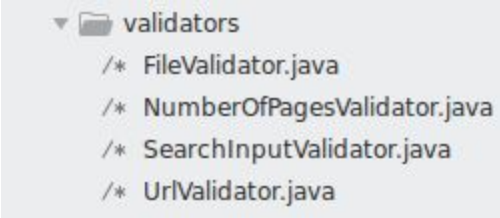
```
  public void inputUrl()
  {
// Creating object of current class object to use in this method only
    SystemInput systemInputObj=new SystemInput();
    // Taking Url as a input from user.
```

```
    System.out.println("\n");
    logger.info("Please Enter Your Url :-");
    Scanner scan = new Scanner(System.in);
    String urlName = scan.nextLine();
    // Calling UrlValidator classes to Check all validations of url.
    UrlValidator validatorObj=new UrlValidator();
    boolean checkValidator=validatorObj.validator(urlName);
    if(checkValidator){
      systemInputObj.searchValue(urlName);
    }else{
      systemInputObj.inputUrl();
    }
  }
```

Story 2 - Implementation of limit on search .

Task 1 - user can give maximum 20 page to search

“ Create NumberOfPagesValidator.java into validators folder to validate all validation On number of pages to search”



1- Create NumberOfPagesValidator class into NumberOfPagesValidator.java file

```java
package com.ncu.validators;
import com.ncu.exceptions.*;
import com.ncu.processors.GetLogger;
import com.ncu.processors.GetConfiguration;
import java.util.Properties;
import org.apache.log4j.Logger;
public class NumberOfPagesValidator{
 Logger logger;
 Properties message,constants;
}
```

2- create constructor to initialization of logger , configMessage and configConstants files

```java
 public NumberOfPagesValidator()
      {
// initialization of GetLogger to get logger object
GetLogger loggerObject=new GetLogger();
Logger logger=loggerObject.loggerValue("NumberOfPagesValidator");
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;
      }
```

3- create validator() method into NumberOfPagesValidator.java file where we will validate all (limit value) validation .

Description - “this method will validate all validation of user given value as a limit of search . if all validation will true then this method will return true otherwise false”

```
public boolean validator(int numberOfPage)
{
try{
    }catch(){
   }
}
```

4- add “limit” value into constants.properties file of constants folder .

```
maxPageNumber=20
```

5- Create maxNumberOfPages() method into NumberOfPagesValidator.java to apply limit on search .

```
/* Generate "MaxPageNumberException" Exception if user give more then
20 page number to search */
private void maxNumberOfPages(int numberOfPage) throws
MaxPageNumberException {
String pageNumber = constants.getProperty("maxPageNumber");
  int getLength=Integer.parseInt(pageNumber);
      if(numberOfPage>getLength){
        throw new MaxPageNumberException("");
      }
}
```

3- call this maxNumberOfPages() method from validator() with limit value

```
// Generate "MaxNumberOfPages" Exception if user give more then 20
length of string to search.
  validatorObj.maxNumberOfPages(numberOfPage);
```

4- catch MaxPageNumberException generated by maxNumberOfPages() method into catch block

```
public boolean validator(int numberOfPage)
      {
```

```
  try{
NumberOfPagesValidator validatorObj=new NumberOfPagesValidator();
// Generate "MaxNumberOfPages" Exception if user give more then 20 length of string
to search.
       validatorObj.maxNumberOfPages(numberOfPage);
       }
catch(MaxPageNumberException e){
logger.error("\n"+e+message.getProperty("maxPageNumberMessage")+"\n");
       return false;
       }
     catch(Exception e){
        logger.error("\n"+e+"\n");
        return false;
       }
  return true;
}
```

5- add custom message into “exceptions.properties” for “maxPageNumberMessage”

```
maxPageNumberMessage="Oops.. This System Accept Only Less Than 20
Pages Numbers To Search ..!"
```

6- maxNumberOfPages() method throws MaxPageNumberException so you should create MaxPageNumberException exception into exceptions folder.

6.1 - create InvalidUrlException java file into “exceptions” folder

6.2- add below code into MaxPageNumberException.java file

```
package com.ncu.exceptions;
public class MaxPageNumberException extends Exception{
 public MaxPageNumberException(String s){
  super(s);
 }
}
```

Task 2 - Call "validator()" method of SearchInputValidator class from SystemInput.java

1- after take (Limit of search) as a input into numberOfPages method of SystemInput.java file . call validator method of SearchInputValidator.java file to apply validation

2- add below code into "SystemInput.java" file of processors

```
 // 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 NumberOfPagesValidator classes to Check all validations of
number of pages to search.
      NumberOfPagesValidator pageObject=new NumberOfPagesValidator();
      boolean checkValidator=pageObject.validator(numberOfPage);
```

```
      if(checkValidator){
        SearchText searchTextObject =new SearchText();
        this.searchTextObject=searchTextObject;
        System.out.println("\n");
        logger.info("Process Started . . . . . . . . . . . . .");
        // Getting Permission to write output into json file .
        System.out.println("\n");
        logger.info("Do You Want to Write this Information Into Json
File Plz Press y/n:- ");
        // Getting config permission value
        Scanner scanObj = new Scanner(System.in);
        String permission= scanObj.nextLine();
        String getPermisson = constants.getProperty("permission");
        if(getPermisson.equalsIgnoreCase(permission)){
        }else{
          // Implementation of exit code
          System.out.println("\n");
          logger.info(message.getProperty("notWriteInFileMessage"));
          systemInputObj.exits();
        }
      }else{
        systemInputObj.numberOfPages(urlName,searchValue);
      }
    }catch(InputMismatchException e){

logger.error("\n"+e+message.getProperty("validIntegerInput")+"\n");
      systemInputObj.numberOfPages(urlName,searchValue);
    }
  }
```

Story 3 - Implementation of validation on Search Word .

Task 1 - User can not give blank space to search .

“ Create SearchInputValidator.java file into validators folder to validate all validation On user given search word ”

1- Create SearchInputValidator class into SearchInputValidator.java file

```
package com.ncu.validators;
import com.ncu.exceptions.*;
import com.ncu.processors.GetLogger;
import com.ncu.processors.GetConfiguration;
import java.util.Properties;
import java.util.regex.Matcher;
import java.util.regex.Pattern;
import org.apache.log4j.Logger;
public class SearchInputValidator{
Logger logger;
Properties message,constants;
}
```

2- create constructor to initialization of logger , configMessage and configConstants files

```
public SearchInputValidator()
      {
 // initialization of GetLogger to get logger object
      GetLogger loggerObject=new GetLogger();
      Logger logger=loggerObject.loggerValue("SearchInputValidator");
```

```
     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;
     }
```

3- create validator() method into SearchInputValidator.java file where we will validate on user given search word value and generate exception

```
public boolean validator(String searchValue)
     {
try{
    }catch(){
}
```

5- Create empltySearchInput() method into SearchInputValidator.java to validate user give search value should not empty .

```
/* Generate "empltySearchInput" Exception if user gives blank space
to search  */
private void empltySearchInput(String searchValue) throws
EmptySearchInputException {
   if (searchValue == null || searchValue.trim().isEmpty()) {
     throw new EmptySearchInputException("");
     }
}
```

6- call this empltySearchInput() method from validator() with search Value

```
// Generate "EmptySearchException" Exception if user gives blank
```

```
space to search .
   validatorObj.empltySearchInput(searchValue);
```

4- catch EmptySearchInputException generated by empltySearchInput() method into catch block

```
public boolean validator(String searchValue)
{

  try{
  SearchInputValidator validatorObj=new SearchInputValidator();

 // Generate "EmptySearchException" Exception if user gives blank space to search .
  validatorObj.empltySearchInput(searchValue);

       }
       catch(EmptySearchInputException e){

logger.error("\n"+e+message.getProperty("emptySearchMessage")+"\n");
        return false;
        }

       catch(Exception e){
       logger.error("\n"+e+"\n");
       return false;
    }
  return true;
}
```

5- add custom message into “exceptions.properties” for “emptySearchMessage”

```
emptySearchMessage="Oops.. Empty Search Is Not Acceptable ...!"
```

6- empltySearchInput() method throws EmptySearchInputException so you should create EmptySearchInputException exception into exceptions folder.

6.1 - create EmptySearchInputException java file into “exceptions” folder

6.2- add below code into EmptySearchInputException.java file

```
package com.ncu.exceptions;
public class EmptySearchInputException extends Exception{
 public EmptySearchInputException(String s){
  super(s);
 }
}
```

Task 2 - User can give limited length of string to search .

1- add “search string limit ” value into constants.properties file of constants folder .

```
searchLength=15
```

2- Create checkLength() method into SearchInputValidator.java to validate user give search value should not more then config define length .

```
/* Generate "MaxLengthException" Exception if user give more then 15
length of string */

private void checkLength(String searchValue) throws
```

```
MaxLengthException {
String lengthValue = constants.getProperty("searchLength");
      int getLength=Integer.parseInt(lengthValue);
      if(searchValue.length()>getLength){
         throw new MaxLengthException("");
      }
}
```

3- call this checkLength() method from validator() with searchValue

```
// Generate "MaxLengthException" Exception if user give more then 15
length of string to search.
      validatorObj.checkLength(searchValue);
```

4- catch MaxLengthException generated by checkLength() method into catch block

```
 public boolean validator(String searchValue)
  {
   try{
 SearchInputValidator validatorObj=new SearchInputValidator();
 // Generate "EmptySearchException" Exception if user gives blank
space to search .
     validatorObj.empltySearchInput(searchValue);
// Generate "MaxLengthException" Exception if user give more then 15
length of string to search.
     validatorObj.checkLength(searchValue);
      }
      catch(EmptySearchInputException e){
logger.error("\n"+e+message.getProperty("emptySearchMessage")+"\n");
        return false;
      }
      catch(MaxLengthException e){
this.logger.error("\n"+e+message.getProperty("searchLengthMessage")+"
```

```
\n");
      return false;
    }
    catch(Exception e){
        logger.error("\n"+e+"\n");
        return false;
    }
  return true;
}
```

5- add custom message into “exceptions.properties” for “searchLengthMessage”

```
searchLengthMessage="You have Given Long File Name .This System
Accept Only Less Than 15 Characters To File Name ..!"
```

6- checkLength() method throws MaxLengthException so you should create MaxLengthException exception into exceptions folder .

6.1 - create MaxLengthException java file into “exceptions” folder

6.2- add below code into MaxLengthException.java file

```
package com.ncu.exceptions;

public class MaxLengthException extends Exception{
```

```
 public MaxLengthException(String s){
  super(s);
 }
}
```

Task 3 - Call “validator()” method of SearchInputValidator class from SystemInput.java file

1- after take (Search String) as a input into search Value method of SystemInput.java file . call validator method of SearchInputValidator.java file to apply validation

2- add below code into “SystemInput.java” file of processors .

```
  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 UrlValidator classes to Check all validations of url.
    UrlValidator validatorObj=new UrlValidator();
    boolean checkValidator=validatorObj.validator(urlName);
    if(checkValidator){
      systemInputObj.searchValue(urlName);
    }else{
      systemInputObj.inputUrl();
    }
  }
```

Story 3 - Source code of files .

Task 1 - Source code of file "SystemInput.java" file of processors

```
/**
*  The SystemInput is class to take Url ,
*  search word , and page number count from user
*
* @author  knight Learning Solutions
* @version 1.0
* @since   2019-1-5
*/

package com.ncu.processors;

import com.ncu.validators.UrlValidator;
import com.ncu.validators.SearchInputValidator;
import com.ncu.validators.NumberOfPagesValidator;

import java.util.Scanner;
import java.util.Properties;
import java.util.InputMismatchException;

import org.apache.log4j.Logger;

public class SystemInput
{
  Logger logger;
  Properties message,constants;

  //Creating Contructor 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 Vaditation 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 UrlValidator classes to Check all validations of url.
    UrlValidator validatorObj=new UrlValidator();
    boolean checkValidator=validatorObj.validator(urlName);
    if(checkValidator){
      systemInputObj.searchValue(urlName);
    }else{
      systemInputObj.inputUrl();
    }
  }
  // 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 SearchInputValidator classes to Check all validations of
Search words.
    SearchInputValidator searchObject=new SearchInputValidator();
    boolean checkValidator=searchObject.validator(searchValue);
    if(checkValidator){
      systemInputObj.numberOfPages(urlName,searchValue);
    }else{
      systemInputObj.searchValue(urlName);
    }
  }
  // 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 NumberOfPagesValidator classes to Check all validations of
number of pages to search.
      NumberOfPagesValidator pageObject=new NumberOfPagesValidator();
      boolean checkValidator=pageObject.validator(numberOfPage);
      if(checkValidator){
        System.out.println("\n");
        logger.info("Process Started . . . . . . . . . . . . .");
        // Getting Perssion to write output into json file .
        System.out.println("\n");
        logger.info("Do You Want to Write this Information Into Json File
Plz Press y/n:- ");
        // Getting config permission value
        Scanner scanObj = new Scanner(System.in);
        String permission= scanObj.nextLine();
        String getPermisson = constants.getProperty("permission");
        if(getPermisson.equalsIgnoreCase(permission)){
        }else{
          // Implementation of exit code
          System.out.println("\n");
```

```
          logger.info(message.getProperty("notWriteInFileMessage"));
          systemInputObj.exits();
        }
      }else{
        systemInputObj.numberOfPages(urlName,searchValue);
      }
    }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();

  }

  // 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();
    }
  }
}
```

Task 2 - Source code of file "UrlValidator.java" file of validators

```
/**
*  The UrlValidator class will check all validations of url
*  List of validation -
*  1- EmptyUrlException - if User will give black space instead of url
*  2- InvalidUrlException - if User will give invalid Url.
*
* @author  knight Learning Solutions
* @version 1.0
* @since   2019-1-5
*/
package com.ncu.validators;
import com.ncu.exceptions.*;
import com.ncu.processors.GetLogger;
import com.ncu.processors.GetConfiguration;

import java.util.Properties;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

import org.apache.log4j.Logger;

public class UrlValidator{
      Logger logger;
      Properties message,constants;
      public UrlValidator()
      {
            // initialization of GetLogger to get logger object
            GetLogger loggerObject=new GetLogger();
            Logger logger=loggerObject.loggerValue("UrlValidator");
            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;
      }
      public boolean validator(String urlName)
      {
            try{
                  UrlValidator validatorObj = new UrlValidator();
      //Generate "EmptyUrlException" Exception if gives blank space as a
Url
                  validatorObj.empltyUrlMethod(urlName);
      //Generate "InvalidUrlException" Exception if user give wrong Url
                  validatorObj.invalidUrlMethod(urlName);
            }
      //All Excetion will taken in this section
            catch(EmptyUrlException e){
      logger.error("\n"+e+message.getProperty("emptyUrlMessage")+"\n");
                  return false;
            }
            catch(InvalidUrlException e){
this.logger.error("\n"+e+message.getProperty("invalidUrlMessage")+"\n");
                  return false;
            }
            catch(Exception e){
                  logger.error("\n"+e+"\n");
                  return false;
            }
            return true;
      }
      /* Generate "EmptyUrlException" Exception if gives blank space as a
Url  */
private void empltyUrlMethod(String urlName) throws EmptyUrlException {
            if (urlName == null || urlName.trim().isEmpty()) {
                  throw new EmptyUrlException("");
            }
      }

      /* Generate "InvalidUrlException" Exception if user give wrong Url */
```

```
      private void invalidUrlMethod(String urlName) throws
InvalidUrlException {
            String regexValue = constants.getProperty("urlRegex");
            Pattern patObject = Pattern.compile(regexValue);
            Matcher matcher = patObject.matcher(urlName);
            boolean value = matcher.matches();
            if(!value)
            {
                  throw new InvalidUrlException("");
            }
      }
}
```

Task 3 - Source code of file "SearchInputValidator.java" file of validators

```
/**
* Created SearchInputValidator class to check all validation of User given
Search Words.
* 1- EmptySearchInputException - if user will give blank space to search.
* 2- MaxLengthException- Generate "MaxLengthException" Exception if user
give more then 15 length of string
*
* @author  knight Learning Solutions
* @version 1.0
* @since   2019-1-5
*/
package com.ncu.validators;
import com.ncu.exceptions.*;
import com.ncu.processors.GetLogger;
import com.ncu.processors.GetConfiguration;
import java.util.Properties;
import java.util.regex.Matcher;
import java.util.regex.Pattern;
import org.apache.log4j.Logger;

public class SearchInputValidator{
      Logger logger;
```

```
      Properties message,constants;
      public SearchInputValidator()
      {
            // initialization of GetLogger to get logger object
            GetLogger loggerObject=new GetLogger();
            Logger logger=loggerObject.loggerValue("SearchInputValidator");
            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;
      }
      public boolean validator(String searchValue)
      {
      try{
       SearchInputValidator validatorObj=new SearchInputValidator();
      // Generate "EmptySearchException" Exception if user gives blank
space to search .
       validatorObj.empltySearchInput(searchValue);
     // Generate "MaxLengthException" Exception if user give more then 15
length of string to search.
        validatorObj.checkLength(searchValue);
            }
      catch(EmptySearchInputException e){
      logger.error("\n"+e+message.getProperty("emptySearchMessage")+"\n");
                  return false;
            }
      catch(MaxLengthException e){
this.logger.error("\n"+e+message.getProperty("searchLengthMessage")+"\n");
                  return false;
            }
      catch(Exception e){
                  logger.error("\n"+e+"\n");
                  return false;
            }
            return true;
      }
/* Generate "empltySearchInput" Exception if user gives blank space to
```

```
search  */
private void empltySearchInput(String searchValue) throws
EmptySearchInputException {
            if (searchValue == null || searchValue.trim().isEmpty()) {
                  throw new EmptySearchInputException("");
            }
      }
/* Generate "MaxLengthException" Exception if user give more then 15 length
of string */
private void checkLength(String searchValue) throws MaxLengthException {
            String lengthValue = constants.getProperty("searchLength");
            int getLength=Integer.parseInt(lengthValue);
            if(searchValue.length()>getLength){
                  throw new MaxLengthException("");
            }
      }

}
```

Task 4 - Source code of file “NumberOfPagesValidator.java” file of validators

```
/**
* Created NumberOfPagesValidator class to check all validation of User
given input as a
* number pages to search .
* 1- EmpltyNumberOfPages - Generate "empltyNumberOfPages" Exception if user
gives blank space to search.
* 2- MaxPageNumberException-Generate "MaxPageNumberException" Exception if
user give more then 20
*    page number to search
*
* @author  knight Learning Solutions
* @version 1.0
* @since   2019-1-5
*/
package com.ncu.validators;
import com.ncu.exceptions.*;
```

```java
import com.ncu.processors.GetLogger;
import com.ncu.processors.GetConfiguration;
import java.util.Properties;
import org.apache.log4j.Logger;
public class NumberOfPagesValidator{
      Logger logger;
      Properties message,constants;
      public NumberOfPagesValidator()
      {
            // initialization of GetLogger to get logger object
            GetLogger loggerObject=new GetLogger();
            Logger
logger=loggerObject.loggerValue("NumberOfPagesValidator");
            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;
      }
      public boolean validator(int numberOfPage)
      {

      try{
      NumberOfPagesValidator validatorObj=new NumberOfPagesValidator();
// Generate "MaxNumberOfPages" Exception if user give more then 20 length
of string to search.
       validatorObj.maxNumberOfPages(numberOfPage);
       }

      catch(MaxPageNumberException e){
logger.error("\n"+e+message.getProperty("maxPageNumberMessage")+"\n");
                  return false;
            }
            catch(Exception e){
                  logger.error("\n"+e+"\n");
                  return false;
            }
            return true;
      }
```

```
/* Generate "MaxPageNumberException" Exception if user give more then 20
page number to search */
      private void maxNumberOfPages(int numberOfPage) throws
MaxPageNumberException {
            String pageNumber = constants.getProperty("maxPageNumber");
            int getLength=Integer.parseInt(pageNumber);
            if(numberOfPage>getLength){
                  throw new MaxPageNumberException("");
            }
      }
}
```

Task 4 - Source code of file "exceptions.properties" file of constants

```
emptyUrlMessage="Oops.. Empty Url Is Not Acceptable ...!"
invalidUrlMessage="Oops.. Given Url Is Not Valid Plz Try With Other Url
...!"
emptySearchMessage="Oops.. Empty Search Is Not Acceptable ...!"
searchLengthMessage="You have Given Long File Name .This System Accept Only
Less Than 25 Characters To File Name ..!"
maxPageNumberMessage="Oops.. This System Accept Only Less Than 20 Pages
Numbers To Search ..!"
validIntegerInput="Oops.. Mismatch Input Not Please Give Value In Integer
Format..!"
emptyFileNameMessage="Oops.. Empty File Name Is Not Acceptable ...!"
extensionFormatMessage="Oops.. Extension Is Messing .You Should Also Give
.json Extension ...!"
invalidFileExtension="Oops.. This is not json file ! This System Accept
Only json file ...!"
specialcharacterMessage="Oops.. You have given special characters into file
name . This System does not take Special characters ...!"
fileExistMessage="Oops.. This File Is Already Exist Into Directory , Please
Try With Different Name...!"
notWriteInFileMessage="..................... Console Information Is Not
Written Into File ....................";
exitMessage="Do You Want To Exist From System... Please Write (exit) or Any
Other Key To Continue Process ... !"
```

Task 4 - Source code of file “constants.properties” file of constants

```
urlRegex
=\\b(https?|ftp|file)://[-a-zA-Z0-9+&@#/%?=~_|!:,.;]*[-a-zA-Z0-9+&@#/%=~_|]
searchLength=15
maxPageNumber=20
permission=y
setDot=[.]
setFileExtension=json
fileRegex="@#$%^&(,)_"
endProcess=exit
HTML_A_TAG_PATTERN=(?i)<a([^>]+)>(.+?)</a>
```

Story 4 - Compile and run your application of day 2

Task 1 - Compile code

1- compile all exceptions java files

“Open terminal into exceptions folder of project”

```
kls103@kls103-Latitude-3480:~/Desktop/NCU/Exercises/day2/output/WebCrawler/src/com/ncu/exceptions$
```

2-compile all java file of exceptions folder .

```
 javac  -d "/home/kls103/Desktop/NCU/Exercises/day2/output/WebCrawler/classes"
*.java
```

3- compile processors java files

“Open terminal into processors folder of project”

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

4-compile GetLogger.java file of processors folder .

```
javac -cp
".:/home/kls103/Desktop/NCU/Exercises/day2/output/WebCrawler/libs/log4j-1.2.17.jar:
/home/kls103/Desktop/NCU/Exercises/day2/output/WebCrawler/classes" -d
"/home/kls103/Desktop/NCU/Exercises/day2/output/WebCrawler/classes"  GetLogger.java
```

5- compile GetConfiguration.java file of processors folder .

```
javac -cp
".:/home/kls103/Desktop/NCU/Exercises/day2/output/WebCrawler/libs/log4j-1.2.17.jar:
/home/kls103/Desktop/NCU/Exercises/day2/output/WebCrawler/classes" -d
"/home/kls103/Desktop/NCU/Exercises/day2/output/WebCrawler/classes"
GetConfiguration.java
```

6- compile all java files of Validator folder

6.1- open terminal into validator folder .

```
kls103@kls103-Latitude-3480:~/Desktop/NCU/Exercises/day2/output/WebCrawler/src/com/ncu/validators$
```

6.2- compile validators file .

```
javac -cp
".:/home/kls103/Desktop/NCU/Exercises/day2/output/WebCrawler/libs/log4j-1.2.17.jar:
/home/kls103/Desktop/NCU/Exercises/day2/output/WebCrawler/classes" -d
"/home/kls103/Desktop/NCU/Exercises/day2/output/WebCrawler/classes"  *.java
```

7- compile all java file of processors folder .

```
javac -cp
".:/home/kls103/Desktop/NCU/Exercises/day2/output/WebCrawler/libs/log4j-1.2.17.jar:
/home/kls103/Desktop/NCU/Exercises/day2/output/WebCrawler/classes" -d
"/home/kls103/Desktop/NCU/Exercises/day2/output/WebCrawler/classes"  *.java
```

8- compile Crawler.java file of main folder .

8.1 - open terminal into main folder

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

8.2 - compile Crawler.java file of main folder

```
javac -cp
".:/home/kls103/Desktop/NCU/Exercises/day2/output/WebCrawler/libs/log4j-1.2.17.jar:
/home/kls103/Desktop/NCU/Exercises/day2/output/WebCrawler/classes" -d
"/home/kls103/Desktop/NCU/Exercises/day2/output/WebCrawler/classes"  Crawler.java
```

Task 2 - Run code

1 - open terminal in root of application “ WebCrawler”

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

2- Run your application

```
java -cp
".:/home/kls103/Desktop/NCU/Exercises/day2/output/WebCrawler/libs/log4j-1.2.17.jar:
/home/kls103/Desktop/NCU/Exercises/day2/output/WebCrawler/classes"
com.ncu.main.Crawler
```

Story 5- Output of day 2

```
kls103@kls103-Latitude-3480:~/Desktop/NCU/Exercises/day2/output/WebCrawler$ java -cp ".:/home/kls103/Desktop/NCU/Exercises/day2/output/WebCrawle
r/libs/log4j-1.2.17.jar:/home/kls103/Desktop/NCU/Exercises/day2/output/WebCrawler/classes" com.ncu.main.Crawler
2019-01-19 05:33:59 INFO  class:31 - ====================================
2019-01-19 05:33:59 INFO  class:32 - ||        Welcome NCU Students       ||
2019-01-19 05:33:59 INFO  class:33 - ====================================


2019-01-19 05:33:59 INFO  class:55 - Please Enter Your Url :-

2019-01-19 05:34:01 ERROR class:60 -
com.ncu.exceptions.EmptyUrlException: "Oops.. Empty Url Is Not Acceptable ...!"



2019-01-19 05:34:01 INFO  class:55 - Please Enter Your Url :-
SASDFSA
2019-01-19 05:34:05 ERROR class:64 -
com.ncu.exceptions.InvalidUrlException: "Oops.. Given Url Is Not Valid Plz Try With Other Url ...!"



2019-01-19 05:34:05 INFO  class:55 - Please Enter Your Url :-
```