Day 5

In day 5 session we will take one json file from user and we will write all information of console like -
“Visited urls , user given web link , user given word , most recommended url , process time ” into json file .

Task Of Day

I. Write console output information into json file. (file should be json)
II. Logics
    A. Take json file name from user to store console output into json file.
    B. Implementation of validation on json file name.
    C. Short description of simple.json package .
    D. implementation logic to write console information into json file.
III. Information should be
    1- Date
    2- All visited urls
    3- Process Time
    4- Most recommended utr
    5- User given string
    6- User given web Url

Story 1 - Take json file name from user

Task 1 - json file name from user .

1- Create “writeOutputInFile” method in SystemInput.java file of processors folder .

Description - “this method will take “URLCrawler” class object in argument . because we are using only one object of URLCrawler in whole project so at time of write value of console into file i will use same object”

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

2- Create Scanner class object and take one string value as a file name from user.

```
System.out.println("\n");
logger.info("Please Enter Your Json File Name :- ");
Scanner scan = new Scanner(System.in);
String fileName= scan.nextLine();
```

3- Create “FileValidator” class object . In further session i will create this class to validate user given json file name . but at this time you can assume that we have create this file .

Description - In further session we will create FileValidator. Class which will have one validator() method to validate user given json file name and if all validation will true then it return (true) but in case of validation fail it will return (false)

```
// Calling NumberOfPagesValidator class to Check all validations of
number of pages to search.
 FileValidator validatorObj=new FileValidator();
  boolean checkValidator=validatorObj.validator(fileName);
   if(checkValidator)
      {
       urlCrawlerObj.writeOutputInFile(fileName);
       systemInputObj.exits();
      }else{
  systemInputObj.writeOutputInFile(urlCrawlerObj); // urlCrawlerObj
   }
```

Description- In case of validation false we are calling again writeOutputInFile() to take new json file name from user and In case of validation (true) we are calling (writeOutputInFile()) method of URLCrawler class for further process .

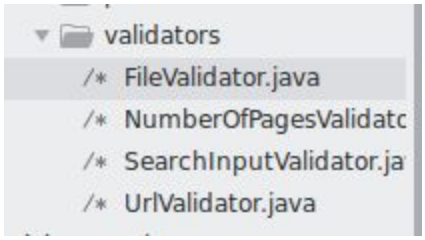
4- “After getting permission from user to write information into json file . call “writeOutputInFile” file from “numberOfPages” method into SystemInput class .
“Add below code into “numberOfPages” method .

```
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)){
systemInputObj.writeOutputInFile(urlCrawlerObj);
  }else{
 // Implementation of exit code
   System.out.println("\n");
logger.info(message.getProperty("notWriteInFileMessage"));
   systemInputObj.exits();
  }
```

Story 2 - Implementation of validation on user given json file name

Tasks -

1- Create “FileValidator” java file with FileValidator class name into “validators” folder



```
public class FileValidator{
}
```

2- imports require package , as well as create package “com.ncu.validators” into this class .

```
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 java.io.File;

import org.apache.log4j.Logger;
```

3- Create constructor to initialization of "Logger" for write information into logger file , "Configuration" for get custom message form "exceptions.properties" file and "configConstants" to get constants value from "constants.properties" file
“ Add below code into “FileValidator” class . ”

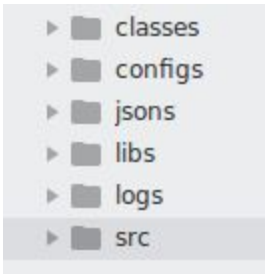
```
public FileValidator()
{
// initialization of GetLogger to get logger object
GetLogger loggerObject=new GetLogger();
Logger logger=loggerObject.loggerValue("FileValidator");
this.logger=logger;

// 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;
}
```

4- Create “jsons” folder in parallel of src folder to store application create json file



5 - Configure the json file path into constructor of FileValidator class

```
public FileValidator()
{

  // calling jsonFilePath method to get path of json file .
  String filePath=propertyObject.jsonFilePath();
  this.filePath=filePath;
}
```

6- Create a validator() method which validate all validation of json file name

Description- “this method will return true if all validation is true and if any exception will generate the this method will return false”

```
public boolean validator(String fileName)
     {
      }
```

7- Create emplty FileName() method into FileValidator class and call it from validator() method to validate user given file name should not empty

```
public boolean validator(String fileName)
 {

 Try{
FileValidator validatorObj=new FileValidator();

// Generate "EmptyFileNameException" Exception if user gives blank
space to as a file name.
 validatorObj.emplty FileName(fileName);
    }catch(){
  }
/* Generate "EmptyFileNameException" Exception if user gives blank
space as a file name  */



private void emplty FileName(String fileName) throws
EmptyFileNameException {
      if (fileName == null || fileName.trim().isEmpty()) {
       throw new EmptyFileNameException("");
      }
}
```

8- emplty FileName method is generating EmptyFileNameException so we should catch it into catch block

```
catch(EmptyFileNameException e){
logger.error("\n"+e+message.getProperty("emptyFileNameMessage")+"\n")
;
                return false;
      }
```

9 - add “emptyFileNameMessage” custom message into “exceptions.properties” file -

```
emptyFileNameMessage="Oops.. Empty File Name Is Not Acceptable ...!"
```

10 - Create a custom “EmptyFileNameException” exception into “exceptions” folder

```
package com.ncu.exceptions;

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

11- Create fileFormat() method into FileValidator class and call it from validator() method to validate user given file should have (dot value (.)) . and call it from validator() method . -

```
public boolean validator(String fileName)
{
   try{
  FileValidator validatorObj=new FileValidator();
// Generate "EmptyFileNameException" Exception if user gives blank
space to as a file name.
     validatorObj.emplty FileName(fileName);
// Generate "FileFormatException" Exception if user does not . before
extension
     validatorObj.fileFormat(fileName);
   }catch(){
}
/* Generate "EmptyFileNameException" Exception if user gives blank
space as a file name  */
private void emplty FileName(String fileName) throws
EmptyFileNameException {
```

```
   if (fileName == null || fileName.trim().isEmpty()) {
     throw new EmptyFileNameException("");
     }
  }
/* Generate "FileFormatException" Exception if user does not . before
extension */
private void fileFormat(String fileName) throws FileFormatException{
    String fileFormatGet = constants.getProperty("setDot");
    Pattern patternObject = Pattern.compile(fileFormatGet);
    Matcher matcherObject = patternObject.matcher(fileName);
    Boolean value = matcherObject.find();
     if(!value){
       throw new FileFormatException("");
     }
}
```

12- fileFormat method is generating FileFormatException so we should catch it into catch block

```
 // To catch FileFormatException .
catch(FileFormatException e){
logger.error("\n"+e+message.getProperty("extensionFormatMessage")+"\n
");
     return false;
}
```

13 - add “extensionFormatMessage” custom message into “exceptions.properties” file -

```
extensionFormatMessage="Oops.. Extension Is Messing .You Should Also
Give .json Extension ...!"
```

14 - Create a custom “FileFormatException” exception into “exceptions” folder

```
package com.ncu.exceptions;

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

15 - Create checkDotFormat() method into FileValidator class and call it from validator() method to validate user given file name should have extension after file name

```
public boolean validator(String fileName)
     {

 Try{
  FileValidator validatorObj=new FileValidator();

// Generate "FileFormatException" Exception if user does not . before
extension
   validatorObj.checkDotFormat(fileName);
         }catch(){
}
/* Generate "InvalidFileException" Exception if user will not give
dot (.) in file */
     private void checkDotFormat(String fileName) throws
InvalidFileException {
           String [] haveDot= fileName.split("\\.");
           if (haveDot.length<=1) {
                 throw new InvalidFileException("");
```

```
      }
}
```

16- checkDotFormat method is generating InvalidFileException so we should catch it into catch block

```
// To catch InvalidFileException
catch(InvalidFileException e){
logger.error("\n
\n"+e+message.getProperty("invalidFileExtension")+"\n");
      return false;
}
```

17- add “invalidFileExtension” custom message into “exceptions.properties” file -

```
invalidFileExtension="Oops.. This is not json file ! This System
Accept Only json file ...!"
```

18 - Create a custom “InvalidFileException” exception into “exceptions” folder

```
package com.ncu.exceptions;

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

19- Create checkExtension() method into FileValidator class and call it from validator() method to validate user given file extension should have (json)

```
public boolean validator(String fileName)
{
```

```
  try{
FileValidator validatorObj=new FileValidator();
// Generate "InvalidFileException" Exception if user will give other
then json file
  validatorObj.checkExtension(fileName);
}catch(){
}
/* Generate "InvalidFileException" Exception if user will give other
the json url */
private void checkExtension(String fileName) throws
InvalidFileException {
String name = fileName.split("\\.")[0];
String currentExtension = fileName.split("\\.")[1];
String getExtenstion = constants.getProperty("setFileExtension");
   if(!getExtenstion.equals(currentExtension)){
     throw new InvalidFileException("");
     }
}
```

20- fileFormat method is generating FileFormatException so we should catch it into catch block

```
 // To catch FileFormatException .
catch(FileFormatException e){
logger.error("\n"+e+message.getProperty("extensionFormatMessage")+"\n
");
    return false;
}
```

21- add “extensionFormatMessage” custom message into “exceptions.properties” file

```
extensionFormatMessage="Oops.. Extension Is Messing .You Should Also
Give .json Extension ...!"
```

22 - Create a custom “FileFormatException” exception into “exceptions”

folder

```
package com.ncu.exceptions;

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

23- Create checkSpecialCharacter() method into FileValidator class and call it from validator() method to validate user given file name should have not have any special characters .

```
public boolean validator(String fileName)
{
   try{
  FileValidator validatorObj=new FileValidator();
// Generate "SpecialCharacterException" Exception will give any
special character in file name.
validatorObj.checkSpecialCharacter(fileName);
   }catch(){
}
/* Generate "SpecialCharacterException" Exception will give any
special character in filename */
private void checkSpecialCharacter(String fileName) throws
SpecialCharacterException{
      String name = fileName.split("\\.")[0];
      String regexValue = constants.getProperty("fileRegex");
      Pattern  patternGet = Pattern.compile("["+regexValue+"]");
      Matcher matcherObj = patternGet.matcher(name);
      boolean check = matcherObj.find();
      if (check == true){
       throw new SpecialCharacterException("");
      }
  }
}
```

24- checkSpecialCharacter method is generating SpecialCharacterException so we should catch it into catch block

```
// To catch SpecialCharacterException
catch(SpecialCharacterException e){
logger.error("\n"+e+message.getProperty("specialcharacterMessage")+"\
n");
      return false;
}
```

24 - add “specialcharacterMessage” custom message into “exceptions.properties” file -

```
specialcharacterMessage="Oops.. You have given special characters
into file name . This System does not take Special characters ...!"
```

26 - Create a custom “SpecialCharacterException” exception into “exceptions” folder

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

27 -Add below code to check user given file name is already exist into jsons folder
Description- “if user given file name is already into jsons folder of application then it will display message to change file name .”

“Add below code into validator() method “FileValidator.java” file .

```
Boolean fileExists= new File(filePath+fileName).exists();
  if(fileExists)
  {
  System.out.println("\n");
logger.error(message.getProperty("fileExistMessage")+"\n");
     return false;
  }
```

28 -Add below code to check user given file name is already exist into jsons folder

Story 3 - Implementation of logic to write information into json file .

Tasks -

1- create “writeOutputInFile()” method into URLCrawler.java file . this method take file name as a argument value .

“Add below code into URLCrawler class of “URLCrawler.java” file .

```
  // write value into json file
public void writeOutputInFile(String fileName){
String outputFolderPath=System.getProperty("user.dir")+
File.separator+"jsons/";
try (BufferedWriter bw = new BufferedWriter(new
FileWriter(outputFolderPath+fileName))){
            JSONObject ObjWite = new JSONObject();
            ObjWite.put("beforeProccessTime",beforeStart);
            ObjWite.put("afterProccessTime",afterStart);
            ObjWite.put("timeDifference",diffInMilisecond);
            ObjWite.put("searchWord",searchWord);
            ObjWite.put("SearchinPath",jsonArryObject);
            ObjWite.put("Most_Recommended_Url",recomendUrl);
            bw.write(ObjWite.toJSONString());
            bw.flush();
```

```
logger.info("\n(*.*) Successfully Write This Information Into
"+fileName+" Json File\n");
logger.info("You Can Get "+fileName+" On Path -: "+outputFolderPath);
          }catch (IOException e) {
            e.printStackTrace();
            System.out.println(e);
          }
        }
```

Story 3 - Implementation of logic to write information into json file .

Tasks -