Project Title – Java Programming Essentials –L1-Assignment

# Task to implement exception handling for given scenario:

**package** Sample;

**import** java.util.Scanner;

**public** **class** Student **extends** Exception

{

**private** **static** **final** **long** *serialVersionUID* = 1L;

**private** String name;

**private** **double** rollno;

**private** String stuclass;

**private** **static** **double** *marks*;

**private** **static** **double** *result*;

**public** **double** getMarks(**double** marks)

{

**return** marks;

}

**public** **void** setMarks(**double** mark1,**double** mark2,**double** mark3) {

**double** marks=mark1+mark2+mark3;

**this**.*marks* = marks;

}

**public** String getName()

{

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** **double** getRollno() {

**return** rollno;

}

**public** **void** setRollno(**double** rollno) {

**this**.rollno = rollno;

}

**public** String getStuclass() {

**return** stuclass;

}

**public** **void** setStuclass(String stuclass) {

**this**.stuclass = stuclass;

}

**public** **static** **double** getResult(**double** marks)

{

**double** avg=(marks)/3;

**return** avg;

}

**public** **void** setResult(**double** avg)

{

**this**.*result*=avg;

}

**public** **static** **void** main(String[] args)

{

Student student = **new** Student();

Scanner scanner = **new** Scanner(System.*in*);

System.*out*.println("Enter Name: ");

String name = scanner.nextLine();

student.setName(name);

System.*out*.println("Enter Student Class(Section A,B,C): ");

String stuclass = scanner.nextLine();

student.setStuclass(stuclass);

System.*out*.println("Enter roll number: ");

**int** rollNumber = scanner.nextInt();

student.setRollno(rollNumber);

System.*out*.println("Enter three subject marks: ");

**double** mark1=scanner.nextInt();

**double** mark2=scanner.nextInt();

**double** mark3=scanner.nextInt();

student.setMarks(mark1,mark2,mark3);

**try**

{

**if** (*result*< 50 || *marks*< 50) {

}

} **catch**(Exception ex){

System.*out*.println("Exception Message: "+ex.getMessage());

ex.printStackTrace();

}

System.*out*.println("Name of the student:" +name + "\n" +"ClassName:" +stuclass + "\n" +"RollNumber:" +rollNumber +"\n" +"Subject1:" +mark1 +"\n" +"Subject2:" +mark2 +"\n" +"Subject3:" +mark3 +"\n" +"Addition of marks:" +*marks* +"\n" +"Result:" +*getResult*(*marks*));

}

}

1. Task to implement Java Threads by for given scenario:

**package** Sample;

**import** java.util.ArrayList;

**import** java.util.List;

**public** **class** MyThreadJoin {

**public** **static** List<String> *names* = **new** ArrayList<String>();

**public** **static** **void** main(String a[]){

List<SampleThread> list = **new** ArrayList<SampleThread>();

**for**(**int** i=0;i<5;i++){

SampleThread s = **new** SampleThread();

list.add(s);

s.start();

}

**for**(SampleThread st:list){

**try**{

st.join();

} **catch** (Exception ex){}

}

System.*out*.println(*names*);

}

}

**class** SampleThread **extends** Thread{

**public** **void** run(){

**for**(**int** i=0; i<10; i++){

**try**{

Thread.*sleep*(10);

} **catch**(Exception ex){}

}

MyThreadJoin.*names*.add(getName());

}

}

Output: [Thread-1, Thread-3, Thread-0, Thread-4, Thread-2]

# 3. Task to implement HTML tag remover Java utility :

**package** Sample;

**class** MyHtmlTagRemover {

**public** **static** **void** main(String a[]){

String text = "<B>I don't want this to be bold<\\B>";

System.*out*.println(text);

text = text.replaceAll("\\<.\*?\\>", "");

System.*out*.println(text);

}

}

Output: I don't want this to be bold

# 4.Task to implement given class relationship and related functionalities:

**package** Sample;

**public** **class** MyWildcardEx {

**public** **static** **void** main(String a[]){

MyEmployeeUtil<CompAEmp> empA

= **new** MyEmployeeUtil<CompAEmp>(**new** CompAEmp("Rahul", 20000));

MyEmployeeUtil<CompBEmp> empB

= **new** MyEmployeeUtil<CompBEmp>(**new** CompBEmp("Swathi", 30000));

MyEmployeeUtil<CompAEmp> empC

= **new** MyEmployeeUtil<CompAEmp>(**new** CompAEmp("chinnu", 20000));

System.*out*.println("Is salary same? "+empA.isSalaryEqual(empB));

System.*out*.println("Is salary same? "+empA.isSalaryEqual(empC));

}

}

**class** MyEmployeeUtil<T **extends** Emp>{

**private** T emp;

**public** MyEmployeeUtil(T obj){

emp = obj;

}

**public** **int** getSalary(){

**return** emp.getSalary();

}

**public** **boolean** isSalaryEqual(MyEmployeeUtil<?> otherEmp){

**if**(emp.getSalary() == otherEmp.getSalary()){

**return** **true**;

}

**return** **false**;

}

}

**class** Emp{

**private** String name;

**private** **int** salary;

**public** Emp(String name, **int** sal){

**this**.name = name;

**this**.salary = sal;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** **int** getSalary() {

**return** salary;

}

**public** **void** setSalary(**int** salary) {

**this**.salary = salary;

}

}

**class** CompAEmp **extends** Emp{

**public** CompAEmp(String nm, **int** sal){

**super**(nm, sal);

}

}

**class** CompBEmp **extends** Emp{

**public** CompBEmp(String nm, **int** sal){

**super**(nm, sal);

}

}

output:

Is salary same? false

Is salary same? True

# 5. Task to learn compressing and un-compressing file(text) using java.util.zip.GZIPOutputStream API:

**package** Sample;

**public** **class** MyFileGZipExamp {

**public** **void** doGzip(String filePath){

}

**public** **void** doUnGzipFile(String filePath) {

}

**public** **static** **void** main(String a[]){

MyFileGZipExamp mfg = **new** MyFileGZipExamp();

mfg.doGzip("C:/Users/sw325873/Downloads/core\_java\_assignments.txt");

mfg.doUnGzipFile("C:/Users/sw325873/Downloads/core-java-l1-training\_0 (1).txt");

}

}

# 6. Task to learn formatting a String using Java Regular Expressions:

**package** Sample;

**import** java.io.FileInputStream;

**import** java.io.FileOutputStream;

**import** java.io.IOException;

**import** java.util.zip.GZIPInputStream;

**import** java.util.zip.GZIPOutputStream;

**public** **class** MyFileGZipExamp{

**public** **void** doGzip(String filePath){

FileOutputStream fos = **null**;

GZIPOutputStream gos = **null**;

FileInputStream fis = **null**;

**try** {

fos = **new** FileOutputStream("C:/Users/t566621/Downloads/fs-core\_java-training.gzip");

gos = **new** GZIPOutputStream(fos);

fis = **new** FileInputStream(filePath);

**byte**[] tmp = **new** **byte**[4\*1024];

**int** size = 0;

**while** ((size = fis.read(tmp)) != -1) {

gos.write(tmp, 0, size);

}

gos.finish();

System.*out*.println("Done with GZip");

} **catch** (IOException e) {

} **finally**{

**try**{

**if**(fis != **null**) fis.close();

**if**(gos != **null**) gos.close();

} **catch**(Exception ex){}

}

}

**public** **void** doUnGzipFile(String filePath) {

GZIPInputStream gis = **null**;

FileInputStream fis = **null**;

FileOutputStream fos = **null**;

**try** {

fis = **new** FileInputStream(filePath);

gis = **new** GZIPInputStream(fis);

fos = **new** FileOutputStream("C:/Users/t566621/Downloads/core\_java\_assignments.docx");

**byte**[] tmp = **new** **byte**[4\*1024];

**int** size = 0;

**while** ((size = gis.read(tmp)) > 0) {

fos.write(tmp, 0, size);

}

fos.flush();

System.*out*.println("Done with uncompressing GZip file.");

} **catch** (IOException ex) {

ex.printStackTrace();

} **finally** {

**try**{

**if**(gis != **null**) gis.close();

**if**(fos != **null**) fos.close();

} **catch**(Exception ex){}

}

}

**public** **static** **void** main(String a[]){

MyFileGZipExamp mfg = **new** MyFileGZipExamp();

mfg.doGzip("C:/Users/t566621/Downloads/core\_java\_assignments.docx");

mfg.doUnGzipFile("C:/Users/t566621/Downloads/fs-core\_java-training.gzip");

}

}

Output:

Done with GZip

Done with uncompressing GZip file.