**Assignment 4**

1. What’s the difference between final, finally? What is finalize()?

the [**final**](https://www.javatpoint.com/final-keyword) is an access modifier,

[**finally**](https://www.javatpoint.com/finally-block-in-exception-handling) is the block in Exception Handling

[**finalize**](https://www.javatpoint.com/java-object-finalize-method) is the method of object class.

1. What’s the difference between throw and throws?

Throws VS throw:

-Use behind method and following by name of Exception/ Use in a method and following by the name of exception instance.

-throw an exception which should be handled by caller/throw an exception which should be handled by catch.

-represent the possibility of occurring exception / must occur exception.

1. What are the two types of exceptions?

Diagram

Description automatically generated

Exceptions:

Exception can be handled by programs.

1)Runtime Exception:

Examples for RuntimeException are illegal cast operation,for example NullPointerException, IndexOutOfBoundsException. These are uncheck exception. Compiler do not check them. We should avoid this error logically.

2)Compile-time Exception:

There are exceptions out of RuntimeException. These exceptions are checked by the compiler during the compilation process to confirm whether the exception is handled by the programmer or not. If not, then the system displays a compilation error. For example , IOException、SQLException. We need to use try catch or throws to address them.

Uncheck Exceptions: includes runtime exception and errors. Compiler don’t check them. We should modify program to address it.

Check Exceptions: These exceptions are checked by the compiler during the compilation process to confirm whether the exception is handled by the programmer or not. If not, then the system displays a compilation error.

1. What is error in java?

error represent serious problems which should be removed before compiling and executing. For example , JVM error -> stackoverflow and outofmemory.

1. Exception is object, true or false?



Exception is a class which inherit Throwable.

1. Can a finally block exist with a try block but without a catch?

Text

Description automatically generated

Yes.

1. From java 1.7, give an example of the try-resource feature.

Graphical user interface, text

Description automatically generated

1. What will happen to the Exception object after exception handling?

 The Exception object **will be garbage collected in the next garbage collection**.

1. Can we use String as a condition in switch(str){} clause?

Yes.

Graphical user interface, text, application

Description automatically generated

1. What’s the difference between ArrayList, LinkedList and vector?

ArrayList has high efficiency of random access.

LinkedList has higher efficiency for insertion and deletion.

Vector: is the thread-safe version of ArrayList.

1. What’s the difference between hashTable and hashMap?

**HashMap is not thread-safe but Hashtable is thread-safe.**

1. What is static import?

With the help of static import, we can access the static members of a class directly without class name or any object.

Text

Description automatically generated

1. What is static block?

a static block is **a set of instructions that is run only once when a class is loaded into memory.**

1. Explain the keywords:

default(java 1.8): A Java default keyword is **an access modifier**. The default access modifier is accessible within the package only.

Break: The break keyword is used to break out a for loop, a while loop or a switch block.

Continue: The continue keyword is used to end the current iteration in a for loop (or a while loop), and continues to the next iteration.

Synchronized: specifies that methods can only be accessed by one thread at a time

Strictfp: strictfp is used to ensure that floating points operations give the same result on any platform

Transient: The transient keyword in Java is **used to avoid serialization**.

Volatile: The volatile keyword flushes the changes directly to the main memory instead of the CPU cache

InstanceOf: Checks whether an object is an instance of a specific class or an interface

1. Create a program including two threads – thread read and thread write. Input file ->Thread read -> Calculate -> buffered area

Buffered area -> Thread write -> output file Detailed description is in assignment4.txt file. Sample input.txt file.

Attached files are input.txt and a more detailed description file.

public class ReadThread implements Runnable{  
  
 protected BlockingQueue<String> blockingQueue = null;  
  
 public ReadThread(BlockingQueue<String> blockingQueue){  
 this.blockingQueue = blockingQueue;  
 }  
 private String Calulate(String str)  
 {  
 int rst = 0;  
 char lastOper = ' ';  
 for(int i = 0; i < str.length(); i++)  
 {  
 char cur = str.charAt(i);  
 if(cur<='9'&& cur>='0')  
 {  
 int num = cur - '0';  
 if(lastOper != ' ')  
 {  
 if(lastOper == '+')  
 {  
 rst += num;  
 }  
 else if(lastOper == '-')  
 {  
 rst -= num;  
 }  
 }  
 else  
 {  
 rst += num;  
 }  
 }  
 else if(cur == '+' && cur == '-')  
 {  
 lastOper = cur;  
 }  
 }  
 StringBuilder sb = new StringBuilder(str);  
 sb.append(' ').append('=').append(' ').append(rst);  
 return sb.toString();  
 }  
 @Override  
 public void run() {  
 BufferedReader br = null;  
 try {  
 br = new BufferedReader(new FileReader(".\\src\\main\\java\\LMSAssignment\\Assignment4\\input.txt"));  
 String buffer =null;  
 while((buffer=br.readLine())!=null){  
 if (!buffer.equals(""))  
 {  
 blockingQueue.put(Calulate(buffer));  
 }  
 else  
 {  
 blockingQueue.put(buffer);  
 }  
  
 }  
 blockingQueue.put("EOF");  
  
 } catch (FileNotFoundException e) {  
  
 e.printStackTrace();  
 } catch (IOException e) {  
  
 e.printStackTrace();  
 } catch(InterruptedException e){  
  
 }finally{  
 try {  
 br.close();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
  
  
 }  
  
  
  
}

public class WriteThread implements Runnable{  
  
 protected BlockingQueue<String> blockingQueue = null;  
  
 public WriteThread(BlockingQueue<String> blockingQueue){  
 this.blockingQueue = blockingQueue;  
 }  
  
 @Override  
 public void run() {  
 FileWriter writer = null;  
  
 try {  
 writer = new FileWriter(".\\src\\main\\java\\LMSAssignment\\Assignment4\\output.txt");  
  
 while(true){  
 String buffer = blockingQueue.take();  
 //Check whether end of file has been reached  
 if(buffer.equals("EOF")){  
 break;  
 }  
 writer.write(buffer);  
 writer.write('\n');  
 }  
  
  
 } catch (FileNotFoundException e) {  
  
 e.printStackTrace();  
 } catch(InterruptedException e){  
  
 } catch (IOException e) {  
 e.printStackTrace();  
 } finally{  
 try {  
 writer.close();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
  
 }  
  
}

public class Main {  
 public static void main(String[] args) {  
 BlockingQueue<String> queue = new ArrayBlockingQueue<String>(1024);  
  
 ReadThread reader = new ReadThread(queue);  
 WriteThread writer = new WriteThread(queue);  
  
 new Thread(reader).start();  
 new Thread(writer).start();  
 }  
}