Assignment 4

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

Final is the keyword and access modifier use in a class method or variable. Finally is a block in java exception handling to execute the important code whether the exception occurs or not. finalize() is the method in java which is used to perform clean up processing before object is garbage collected.

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

Throw keyword is use inside a function when required to throw an exception logically, it can only propagate the unchecked exceptions. Throws is used in the function signature when the function has some statement can lead to exceptions, and it can only propagate checked exception.

1. What are the two types of exceptions?

Checked and unchecked.

1. What is error in java?

Error is a subclass of Throwable that tells that something serious problem is existing and a reasonable java application should not try to catch that error.

1. Exception is object, true or false?

True.

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

Yes.

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

try (BufferedReader br = new BufferedReader(new FileReader(path)))

{}

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.

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

ArrayList is essentially an array. A linkedlist is implemented as a double linked list. Its performance on add and remopve is better than arraylist but worse on get and set methods. Vector is similar with arraylist, but it is synchronized, which mean vector is thread safe and arraylist is not. LinkedList is also a queue.

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

The main difference is that hashmap is not thread-safe but hashtable is thread-safe.

1. What is static import?

Static import allows programmer to access the static members of a class directly without the class name or any object.

1. What is static block?

The static block is a block of statement inside a java class that will be executed when a class is first loaded into JVM.

1. Explain the keywords: default(java 1.8), break, continue, synchronized, strictfp, transient, volatile, instanceOf

In java8, default was added to allow interface to implement default method without affecting classes that already implemented the interface.

Break is used to terminate loop.

Continue is used to skip the rest code of current iteration in a loop.

Synchronized keyword only allowing a single thread to access the shared data or resources at a particular point of time.

Strictfp is a modifier that stands for strict floating-point. It is used in java for restricting floating-point calculation and ensuring the same result on every platform while performing operations in the floating-point variable.

Transient in java is used to indicate that a field should not be part of the serialization process.

Volatile keyword is used to modify the value of a variable by different threads. The volatile keyword does not cache the value of the variable and always read the variable from memory.

instanceOf is a binary operator used to test if an object is of a given type.

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

**import** java.io.\*;

**import** java.util.concurrent.ArrayBlockingQueue;

**import** java.util.concurrent.BlockingQueue;

**class** ThreadRead **extends** Thread

{

**protected** BlockingQueue<String> blockingQueue = **null**;

**private** String path;

**public** ThreadRead(String path, BlockingQueue<String> blockingQueue)

{

**super**("ReadThread");

**this**.path = path;

**this**.blockingQueue = blockingQueue;

}

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

{

**try**

{ InputStream doc = **new** FileInputStream(path);

StringBuilder sb = **new** StringBuilder();

BufferedReader br = **new** BufferedReader(**new** InputStreamReader(doc));

String line;

**while** ((line = br.readLine()) != **null**)

{

String[] str = line.split(" ");

**if**(str.length==1) **continue**;

**int** sum = Integer.*parseInt*((str[0]));

**for** (**int** i=1; i<str.length;i=i+2)

{

**if**(str[i].equals("+"))

{

sum += Integer.*parseInt*((str[i+1]));

}

**else** sum -= Integer.*parseInt*((str[i+1]));

}

line += " = " + sum;

blockingQueue.put(line);

}

blockingQueue.put("EOF");

}

**catch** (FileNotFoundException e)

{

e.printStackTrace();

}

**catch** (IOException e)

{

e.printStackTrace();

} **catch** (InterruptedException e) {

e.printStackTrace();

}

}

}

**class** ThreadWrite **extends** Thread

{

**protected** BlockingQueue<String> blockingQueue = **null**;

String path;

**public** ThreadWrite(String path, BlockingQueue<String> blockingQueue)

{

**this**.blockingQueue = blockingQueue;

**this**.path = path;

}

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

{

PrintWriter writer = **null**;

**try**

{

writer = **new** PrintWriter(**new** File(path));

**while**(**true**)

{

String buffer = blockingQueue.take();

//Check whether end of file has been reached

**if**(buffer.equals("EOF"))

{

**break**;

}

writer.println(buffer);

}

}

**catch** (FileNotFoundException e)

{

e.printStackTrace();

}

**catch**(InterruptedException e)

{

}

**finally**

{

writer.close();

}

}

}

**public** **class** Assignment4

{

**static** BlockingQueue<String> *queue* = **new** ArrayBlockingQueue<String>(1024);

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

{

ThreadRead tr = **new** ThreadRead("input.txt", *queue*);

ThreadWrite tw = **new** ThreadWrite("assignment4.txt", *queue*);

tr.start();

tw.start();

}

}