Assignment 4

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

   Final is a keyword, it can modify variable, method, and class.

   Variable : after assigned a value, it can't be changed

   Method : this method can't be override

   Class : this class can't be extend

   Finally is a keyword when used with try catch block, the finally block can always be executed before return to the call function

   Finalize() : This is a method to mark the object is going to be delete

2. What's the difference between throw and throws?

   Throw is used in method in order to throw one exception, throws appears on the signature of the method, in order to throw the exception to the call stack

3. What are the two types of exceptions?

   Checked exception : can be checked during compile time, should be handled

   Unchecked exception : happened during runtime

4. What is error in java?

   Error is a subclass of Throwable, it should not be handled, like stack over flow exception

5. Exception is object, true or false?

   true

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

   yes

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

```
static String readFirstLineFromFileWithFinallyBlock(String path) throws IOException
 {
    BufferedReader br = new BufferedReader(new FileReader(path));
    try {
       return br.readLine();
    } finally {
       br.close();
   }
  }
```

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

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

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

   Yes, it will call equals method

10. What's the difference between ArrayList, LinkedList and vector?

    ArrayList is implented by an array, it occupies consecutive space, and it is efficient to retrieve data when you know the index

    LinkedList occupies fragment of heap memory, it is efficient when you add or delete one element at head

    Vector is a thread safe class, all the characteristic is like ArrayList

11. What's the difference between hashTable and hashMap?

    HashTable is thread safe but not efficient

    hashMap is not thread safe but efficient

12. What is static import?

In Java, static import concept is introduced in 1.5 version. With the help of static import, we can access the static members of a class directly without class name or any object. For Example: we always use sqrt() method of Math class by using Math class i.e. Math.sqrt(), but by using static import we can access sqrt() method directly

13. What is static block?

Static block initiate the variables in the block during loading class phase of JVM

14. Explain the keywords:

default(java 1.8), break, continue, synchronized, strictfp, transient, volatile, instanceOf

Default keyword is used in functional interface to declare this method can be implemented

Break : to stop a loop

Continue : continue to next iteration of a loop

Synchronized : keep the block thread safe

Strictfp : to restrict the precision and rouding of floating calculation

Transient : avoid that variable to be serialized

Volatile : to keep thread safe, the variable can't only be read and modify on the main cache

instanceOf : one class is the other class or subclass

15. 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 RandW {
   //this is the common memory place two threads operate
   static String s;
   static int ind;
   public static void main(String[] args) throws IOException {
      RandW a = new RandW();
      int total = 0;
      Reader in = null;
      //get the total lines
      try {
         in = new FileReader("C:\\Users\\guoha\\Downloads\\input.txt");
         BufferedReader bf = new BufferedReader(in);
         while (bf.readLine() != null) {
            total++;
         }
      } catch (FileNotFoundException e) {
         e.printStackTrace();
      } catch (IOException e) {
         e.printStackTrace();
      } finally {
         if (in != null) {
            in.close();
```

```
        }
    }
    int finalTotal = total;
    //put the writer into the thread, it will keep running until all the lines have been
written into the file
    Thread t = new Thread() {
        @Override
        public void run() {
            while (ind < finalTotal || (ind == finalTotal && s != null)) {
                try {
                    a.write("C:\\Users\\guoha\\Downloads\\output.txt");
                    System.out.println("1");
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
        }
    };
    t.start();
    while (ind < total) {
        a.read("C:\\Users\\guoha\\Downloads\\input.txt");
        System.out.println("2");
    }

}
public synchronized  void write(String path) throws IOException {
    if (s == null) {
        return;
    }
    Writer out = null;
    try {
        out = new FileWriter(path);
        BufferedWriter bw = new BufferedWriter(out);
        bw.write(s);
        s = null;
    } catch (FileNotFoundException e) {
        e.printStackTrace();
    } finally {
        if (out != null) {
            out.close();
        }
    }
}
```

```
    public synchronized void read(String path) throws IOException {
        if (s != null) {
            return;
        }
        Reader in = null;
        try {
            in = new FileReader(path);
            BufferedReader bf = new BufferedReader(in);
            int i = ind;
            while (i > 0) {
                bf.readLine();
                i--;
            }
            s = bf.readLine();
            if (s != null) {
                ind++;
            }
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        } finally {
            if (in != null) {
                in.close();
            }
        }
    }
}
```