Assignment 4

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

final is the keyword and access modifier which is used to apply restrictions on a class, method or variable.
finally is the 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 just before object is garbage collected.

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

The throw keyword is used inside a function. It is used when it is required to throw an Exception logically.
The throws keyword is used in the function signature. It is used when the function has some statements that can lead to exceptions.

3. What are the two types of exceptions?

Checked exception and Unchecked exception.

4. What is error in java?

An 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. Generally, it has been noticed that most of the occurring errors are abnormal conditions and cannot be resolved by normal conditions.

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 {

    FileReader fr = new FileReader(path);
    BufferedReader br = new BufferedReader(fr);
    try {
        return br.readLine();
    } finally {
        br.close();
        fr.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

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

ArrayList	LinkedList	Vector
ArrayList internally uses a dynamic array to store its elements.	LinkedList uses Doubly Linked List to store its elements.	vector is similar with arraylist to use dynamic array, but it is synchronized.
ArrayList is slow as array manipulation is slower.	LinkedList is faster being node based as not much bit shifting required.	Vector is slower than arrayList
ArrayList implements only List.	LinkedList implements List as well as Queue.	Vector implements a dynamic array that means it can grow or shrink as required.
ArrayList is faster in storing and accessing data.	LinkedList is faster in manipulation of data.	

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

HashMap	HashTable
No method is synchronized.	Every method is synchronized.
Not thread-safe.	Thread-safe.
Threads are not required to wait and hence relatively performance is high.	It increases the waiting time of the thread and hence performance is low.
Null is allowed for both key and value.	Null is not allowed for both key and value.

12. What is static import?

Static import is a feature introduced in the Java programming language that allows members (fields and methods) which have been scoped within their container class as public and static, to be used in Java code without specifying the class in which the field has been defined.

13. What is static block?

A static block is a set of instructions that is run only once when a class is loaded into memory. A static block is also called a static initialization block. This is because it is an option for initializing or setting up the class at run-time.

14. Explain the keywords:

default(java 1.8): The default keyword specifies some code to run if there is no case match in the switch.

break: It is used to terminate loops and switch statements in java.

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: A piece of logic marked with synchronized becomes a synchronized block, allowing only one thread to execute at any given time.

strictfp: trictfp is used for restricting floating-point calculations and ensuring same result on every platform while performing operations in the floating-point variable.
transient: transient marks a member variable not to be serialized when it is persisted to streams of bytes.
Volatile: Volatile keyword is used to modify the value of a variable by different threads. It is also used to make classes thread safe.
instanceOf: The instanceof keyword checks whether an object is an instance of a specific class or an interface.
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.

ReaderThread

```
import java.io.BufferedReader;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
import java.util.concurrent.BlockingQueue;

public class ReaderThread implements Runnable{

   protected BlockingQueue<String> blockingQueue = null;

   public ReaderThread(BlockingQueue<String> blockingQueue){
      this.blockingQueue = blockingQueue;
   }

   @Override
   public void run() {
      BufferedReader br = null;
       try {
                br = new BufferedReader(new FileReader(new File("input.txt")));
                String buffer =null;
                while((buffer=br.readLine())!=null){
                     blockingQueue.put(buffer);
```

```
                }
                blockingQueue.put("EOF");    //When end of file has been reached

            } catch (FileNotFoundException e) {

                e.printStackTrace();
            } catch (IOException e) {

                e.printStackTrace();
            } catch(InterruptedException e){

            }finally{
                try {
                    br.close();
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
    }
}
```

WriteThread

```
import java.io.File;
import java.io.FileNotFoundException;
import java.io.PrintWriter;
import java.util.concurrent.BlockingQueue;

public class WriterThread implements Runnable{
    protected BlockingQueue<String> blockingQueue = null;
    public WriterThread(BlockingQueue<String> blockingQueue){
        this.blockingQueue = blockingQueue;
    }
    @Override
    public void run() {
        PrintWriter writer = null;
        try {
        BufferedReader reader = null;


        File file = new File("input.txt");
        StringBuffer sb = new StringBuffer();
        reader = new BufferedReader(new FileReader("input.txt"));
        writer = new BufferedWriter(new FileWriter(file));
```

```
        String str=reader.readLine();


        while(!str.isEmpty())
        {

            int sum=0;
            char last = '+';
            for(int i=0;i<str.length();i=i+2) {
                if(Character.isDigit(str.charAt(i))){
                    if(last == '+') {
                        sum = sum+(str.charAt(i)-'0');
                    }else {
                        sum = sum-(str.charAt(i)-'0');
                    }

                }else {
                    last=str.charAt(i);
                }
            }
            sb.append(" = ").append(sum);
            writer.append(" = "+sum);

            str = reader.readLine();
            str = reader.readLine();
        }

        } catch (FileNotFoundException e) {

            e.printStackTrace();
        } catch(InterruptedException e){

        }finally{
            reader.close();
            writer.close();
        }
        }
}
```

Launcher Class

```
import java.util.concurrent.ArrayBlockingQueue;
import java.util.concurrent.BlockingQueue;

public class Launcher {

   public static void main(String[] args) {

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

      ReaderThread reader = new ReaderThread(queue);
      WriterThread writer = new WriterThread(queue);

      new Thread(reader).start();
      new Thread(writer).start();

   }

}
```