Assignment 4

1. What's the difference between final, finally? What is finalize()?
    - final
        - final with Variables : The value of variable cannot be changed once initialized.
        - final with Class : The class cannot be subclassed.
        - final with Method : The method cannot be overridden by a subclass.
        - Note : If a class is declared as final then by default all of the methods present in that class are automatically final but variables are not.
    - finally
        - The finally keyword is used in association with a try/catch block and guarantees that a section of code will be executed, even if an exception is thrown. The finally block will be executed after the try and catch blocks, but before control transfers back to its origin.
    - finalize
        - It is a method that the Garbage Collector always calls just before the deletion/destroying the object which is eligible for Garbage Collection, so as to perform clean-up activity. Clean-up activity means closing the resources associated with that object like Database Connection, Network Connection or we can say resource de-allocation. Remember it is not a reserved keyword.
        - Once the finalize method completes immediately Garbage Collector destroy that object.(overriding finalize method)
2. What's the difference between throw and throws?

    Throw is a keyword which is used to throw an exception explicitly in the program inside a function or inside a block of code. Throws is a keyword used in the method signature used to declare an exception which might get thrown by the function while executing the code.
3. What are the two types of exceptions?
        - There are mainly two types of exceptions: checked and unchecked. Here, an error is considered as the unchecked exception.
        - Error and RuntimeException classes are unchecked exceptions, everything else under throwable is checked.
        - Checked: it must specify the exception using throws keyword
        - Unchecked: allow without throws
        - Noted: If a client can reasonably be expected to recover from an exception, make it a checked exception. If a client cannot do anything to recover from the exception, make it an unchecked exception
4. What is error in java?
    - Error : An Error "indicates serious problems that a reasonable application should not try to catch."
    - Some of the examples of errors are Out of memory error or a System crash error.
5. Exception is object, true or false?

    True, Using "throw" to throw a exception object
6. Can a finally block exist with a try block but without a catch?

The finally block always executes when the try block exits. So you can use finally without catch but you must use try. The finally block always executes when the try block exits.

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

```
public static void main(String[] args) {

    BufferedReader br = null;
    String line;

    try {

       br = new BufferedReader(new FileReader("C:\\testing.txt"));
       while ((line = br.readLine()) != null) {
          System.out.println(line);
       }

    } catch (IOException e) {
       e.printStackTrace();
    } finally {
       try {
          if (br != null) br.close();
       } catch (IOException ex) {
          ex.printStackTrace();
       }
    }

 }
```

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, https://www.geeksforgeeks.org/string-in-switch-case-in-java/

But not in C++

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

- ArrayList is implemented as a resizable array. As more elements are added to ArrayList, its size is increased dynamically. It's elements can be accessed directly by using the get and set methods, since ArrayList is essentially an array.
- LinkedList is implemented as a double linked list. Its performance on add and remove is better than Arraylist, but worse on get and set methods.
- Vector is similar with ArrayList, but it is synchronized.

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

- HashMap is non synchronized. It is not-thread safe and can't be shared between many threads without proper synchronization code whereas Hashtable is synchronized. It is thread-safe and can be shared with many threads.

- HashMap allows one null key and multiple null values whereas Hashtable doesn't allow any null key or value.(Hashtable using object.hashCode to be key)
- HashMap is generally preferred over HashTable if thread synchronization is not needed

ConcurrentHashMap

- The underlined data structure for ConcurrentHashMap is Hashtable.
- ConcurrentHashMap class is thread-safe i.e. multiple threads can operate on a single object without any complications.
- At a time any number of threads are applicable for a read operation without locking
- This type of locking mechanism is known as Segment locking or bucket locking. Hence at a time, 16 update operations can be performed by threads.

12. What is static import?

we can access the static members of a class directly without class name or any object.(like namespace in c++)

```
// without static import
System.out.println(Math.sqrt(4));

// with static import
import static java.lang.Math.*;
System.out.println(sqrt(4));
```

13. What is static block?

static block can be used for static initializations of a class. This code inside static block is executed only once: the first time the class is loaded into memory.

14. Explain the keywords:

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

- default(java 1.8):Java 8 has introduced the concept of default methods which allow the interfaces to have methods with implementation
- Break: statement mainly used to terminate the enclosing loop such as while, do-while, for or switch statement
- Continue: statement mainly skip the rest of loop wherever continue is declared and execute the next iteration.
- synchronized: All synchronized blocks synchronized on the same object can only have one thread executing inside them at a time.(synchronized(sync_object){})
- strictfp: strictfp is a keyword in java used for restricting floating-point calculations and ensuring same result on every platform while performing operations in the floating-point variable. class or an interface is declared with strictfp modifier, then all methods declared in the class/interface
- transient: transient is a variables modifier used in serialization. At the time of serialization, if we don't want to save value of a particular variable in a file, then we use transient keyword.
- volatile:

        - Mutual Exclusion: It means that only one thread or process can execute a block of code (critical section) at a time.
        - Visibility: It means that changes made by one thread to shared data are visible to other threads.(thread won't have local copy of volatile variable)
        - Synchronized: mutual exclusion and visibility
        - Volatile: visibility
        - https://www.geeksforgeeks.org/volatile-keyword-in-java/
    - instanceOf:
        - instanceof is a keyword that is used for checking if a reference variable is containing a given type of object reference or not.
        - parent object is not an instance of Child
        - Child is an instance of Parent(Object)

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.

```
package com.antra.BlockingQueue;

import java.util.concurrent.*;
import java.io.File;  // Import the File class
import java.util.Scanner; // Import the Scanner class to read text files
import java.io.FileWriter;   // Import the FileWriter class

class Procuder extends Thread {
    BlockingQueue bq;
    Procuder(BlockingQueue src){
        bq = src;
    }
    public void run(){
        try {
             File myObj = new File(".\\com\\antra\\BlockingQueue\\input.txt");
                   Scanner myReader = new Scanner(myObj);
                   while (myReader.hasNextLine()) {
                     String data = myReader.nextLine();
                     //System.out.println(data);
                     bq.put(data);
                   }
                   myReader.close();
             bq.put("end");

        } catch (Exception e) {
          System.out.println("Something went wrong.");
        }
    }
}
```

```
class Consumer extends Thread {
    BlockingQueue bq;
    Consumer(BlockingQueue src){
        bq = src;
    }
    String parseData(String data) {
       String cleared = data.replaceAll("\\s+","");
       String[] nums = cleared.split("[+-]");
       String[] operator = cleared.split("\\d");
       int ans = Integer.valueOf(nums[0]);

       for(int i =1;i<operator.length;i++) {
             ans = (operator[i].equals("+"))?ans+Integer.valueOf(nums[i]):ans-
Integer.valueOf(nums[i]);
       }
       return data+" = "+String.valueOf(ans);
    }
    public void run() {
        try {
             FileWriter myWriter = new
FileWriter(".\\com\\antra\\BlockingQueue\\output.txt");
             while(true){
                var data = (String)bq.take();
                if(data=="end")
                    break;
                else if(!data.equals("")) {
                    myWriter.write(parseData(data));
                    myWriter.write("\n");
                }
            }
             myWriter.close();
        } catch (Exception e) {
          System.out.println(e);
        }

    }
}

public class BlockingQueue2 {
       public static void main(String[] args)
       {
             int capacity = 5;
             BlockingQueue bq = new ArrayBlockingQueue(capacity);

             Procuder procuder = new Procuder(bq);
             Consumer consumer = new Consumer(bq);
             procuder.start();
             consumer.start();
             //System.out.println("queue contains " + bq);
       }
}
```