Assignment 4

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

The final keyword can be used with class method and variable. A final class cannot be inherited, a final method cannot be overridden and a final variable cannot be reassigned.

The finally keyword is used to create a block of code that follows a try block. A finally block of code always executes, whether or not an exception has occurred. Using a finally block allows you to run any cleanup-type statements that you just wish to execute, despite what happens within the protected code.

The finalize() method is used just before object is destroyed and can be called just prior to object creation.

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

The throw keyword is used to throw an exception explicitly. It can throw only one exception at a time. The throws keyword can be used to declare multiple exceptions, separated by a comma. Whichever exception occurs, if matched with the declared ones, is thrown automatically then.

3. What are the two types of exceptions?

There are two types of exceptions: checked exception and unchecked exception. In this guide, we will discuss them. The main difference between checked and unchecked exception is that the checked exceptions are checked at compile-time while unchecked exceptions are checked at runtime.

4. What is error in java?

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.

5. Exception is object, true or false?

True

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

Yes, It is possible to have a try block without a catch block by using a final block. As we know, a final block will always execute even there is an exception occurred in a try block, except System.

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

Prior to Java SE 7, you can use a finally block to ensure that a resource is closed regardless of whether the try statement completes normally or abruptly. The following example uses a finally block instead of a try-with-resources statement:

```
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?

It is recommended to use String values in a switch statement if the data you are dealing with is also Strings.

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

main difference is their implementation which causes different performance for different operations. 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. arraylist is a better choice if your program is thread-safe. vector and arraylist require space as more elements are added. vector each time doubles its array size, while arraylist grow 50% of its size each time. linkedlist, however, also implements queue interface which adds more methods than arraylist and vector, such as offer(), peek(), poll(), etc. note: the default initial capacity of an arraylist is pretty small. it is a good habit to construct the arraylist with a higher initial capacity. this can avoid the resizing cost.

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.

HashMap is generally preferred over HashTable if thread synchronization is not needed.

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.

13. What is static block?

In a Java class, 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), break, continue, synchronized,
strictfp, transient, volatile, instanceOf

Break: The Java break statement is used to break loop or switch statement. It breaks
the current flow of the program at specified condition. In case of inner loop, it breaks

only inner loop.
Continue: The Java continue statement is used to continue the loop.
Synchronized: Java synchronized keyword is used to specify the critical sections or
methods in multithreaded code.
Strictfp: Java strictfp is used to restrict the floating-point calculations to ensure
portability.
Transient: Java transient keyword is used in serialization. If you define any data
member as transient, it will not be serialized.
Volatile: Java volatile keyword is used to indicate that a variable may change
asynchronously.
Instanceof: Java instanceof keyword is used to test whether the object is an instance
of the specified class or implements 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.

```
package Day4;

import java.io.*;

public class ThreadAssignment extends Thread {
   //To create producer and consumer as threads
   //Shared variable
   public static int x = 0;//checks if all lines are read
   public static int y = 0;//variable to switch between threads based upon its value
   public static String line;
   public static String s;

   public ThreadAssignment(String threadName) {     //Constuctor
      super(threadName);       //Call to constructor of Thread class
   }

   public void run() {

      while (x != -1)
      {
         if (Thread.currentThread().getName().contains("Reader")) {
```

```java
if (x != -1&&j==0)
{
    j=1;
    String fileName = "/Users/simengfeng/Downloads/Assignment4/input.txt";

    try {
        // FileReader reads text files in the default encoding.
        FileReader fileReader =
                new FileReader(fileName);

        // Always wrap FileReader in BufferedReader.
        BufferedReader bufferedReader =
                new BufferedReader(fileReader);

        for (int check = 0; check <= x; check++) {

            line = bufferedReader.readLine();
            if(line != null) {
                //if(line.length() > 0) {
                    int sum = getSum(line);
                    line += " " + sum;
                //}
            }
        }
        if (line == null) {
            x = -1;
        } else {
            System.out.println(line);
            x++;
        }



        // Always close files.
        bufferedReader.close();
    } catch (FileNotFoundException ex) {
        System.out.println(
                "Unable to open file '"
                        + fileName + "'");
    } catch (IOException ex) {
        System.out.println(
                "Error reading file '"
```

```java
                                    + fileName + "'");
                    // Or we could just do this:
                    // ex.printStackTrace();

                }
            }

            yield();
        }else if (Thread.currentThread().getName().contains("writer")) {
                if (x != -1 && line != null && j == 1) {
                    j = 0;

                    String fileName = "/Users/simengfeng/Downloads/Assignment4/output.txt";

                    try {
                        // Assume default encoding.
                        FileWriter fileWriter =
                                new FileWriter(fileName, true);

                        // Always wrap FileWriter in BufferedWriter.
                        BufferedWriter bufferedWriter =
                                new BufferedWriter(fileWriter);

                        // Note that write() does not automatically
                        // append a newline character.
                        if(!line.contains("0")) {
                            bufferedWriter.write(line);
                            //bufferedWriter.write(" " + s);
                            bufferedWriter.newLine();
                        }
                        //System.out.println("y");
                        // Always close files.
                        bufferedWriter.close();
                    } catch (IOException ex) {
                        System.out.println(
                                "Error writing to file '"
                                        + fileName + "'");
                        // Or we could just do this:
                        // ex.printStackTrace();
                    }
                }
                Thread.yield();
            } else {
            }
        }
    }

public static void main(String[] args) {

    ThreadAssignment reader = new ThreadAssignment("Reader");
    ThreadAssignment writer = new ThreadAssignment("writer");

    reader.start();
    writer.start();



}

public static int getSum(String line){
    int sum = 0;
    String operation = "+";
    line.trim();
    String[] arr = line.split(" ");
    for (String m : arr) {
        if (m.equals("+")) {
            operation = "+";
        } else if (m.equals("-")) {
            operation = "-";
        }
        if (isNumeric(m)) {
            if (operation.equals("+")) {
                sum += Integer.parseInt(m);
            } else {
                sum -= Integer.parseInt(m);
            }
        }
    }
    return sum;
}

public static boolean isNumeric(String string) {
    int intValue;
```

```java
        if(string == null || string.equals("")) {
            return false;
        }

        try {
            intValue = Integer.parseInt(string);
            return true;
        } catch (NumberFormatException e) {
        }
        return false;
    }
}
```