1. Write an application to find out how many total characters can be held in a single StringBuilder before running out of memory, translate that number of characters to the number of bytes held by that StringBuilder before crashing.

Answer:
The demo Java code for this application is:

```
public class App1 {
    public static void main(String[] args) {
        // define a new StringBuilder object
        StringBuilder sb = new StringBuilder();

        // keep appending until error been caught
        try {
            while (true) {
                sb.append('a');
            }
        }
        catch (OutOfMemoryError e) {
            // check the size of the StringBuilder object
            long length = sb.length();
            long sizeBytes = length * 2;
            double sizeGB = length * 2.0 / 1073741824;
            System.out.println("Number of Characters in StringBuilder: " + length);
            System.out.println("Size of StringBuilder (Bytes): " + sizeBytes + "B");
            System.out.println("Size of StringBuilder (GB): " + sizeGB + "GB");
            e.printStackTrace();
        }
    }
}
```

And the output of this code on my computer is:

```
"C:\Program Files\Java\jdk1.8.0_221\bin\java.exe" ...
Number of Characters in StringBuilder: 1207959550
Size of StringBuilder (Bytes): 2415919100B
Size of StringBuilder (GB): 2.2499999962747097GB
java.lang.OutOfMemoryError: Java heap space
    at java.util.Arrays.copyOf(Arrays.java:3332)
    at java.lang.AbstractStringBuilder.ensureCapacityInternal(AbstractStringBuilder.java:124)
    at java.lang.AbstractStringBuilder.append(AbstractStringBuilder.java:649)
    at java.lang.StringBuilder.append(StringBuilder.java:202)
    at App1.main(App1.java:9)

Process finished with exit code 0
```

Hence, 1207959550 characters can be held in a single StringBuilder object before running out of memory, and this is 2.25GB size in memory.

2. Write an application to demonstrate using StringBuffer and StringBuilder. Write a little bit in the comment in the code, when to use each.

Answer:

We first compare the performance in single thread mode. The code of this single thread comparison is:

```
public class App2Single {
    public static void main(String[] args) {
        int N = 77777777;
        long t;
        // test the speed of StringBuilder in single thread
        {
            StringBuilder sb = new StringBuilder();
            t = System.currentTimeMillis();
            for (int i = N; i > 0 ; i--) {
                sb.append('a');
            }
            System.out.printf("Time for appending %d characters to StringBuilder: %d ms\n",
                    N, System.currentTimeMillis() - t);
        }
        // test the speed of StringBuffer in single thread
        {
            StringBuffer sb = new StringBuffer();
            t = System.currentTimeMillis();
            for (int i = N; i --> 0 ;) {
                sb.append('a');
            }
            System.out.printf("Time for appending %d characters to StringBuffer: %d ms\n",
                    N, System.currentTimeMillis() - t);
        }
    }
}
```

The output of this code on my computer is:

```
"C:\Program Files\Java\jdk1.8.0_221\bin\java.exe" ...
Time for appending 77777777 characters to StringBuilder: 331 ms
Time for appending 77777777 characters to StringBuffer: 1549 ms

Process finished with exit code 0
```

We can see the StringBuilder is more efficient than StringBuffer in single thread mode.

Then we do a multiple thread comparison, the code is:

```
import java.io.IOException;

public class App2Multi {
    private static StringBuilder builder = new StringBuilder();
    private static StringBuffer buffer = new StringBuffer();

    public static void main(String[] args) {
        // test the multi-threading performance for StringBuilder and StringBuffer
        testSeqMulti(builder);
        testSeqMulti(buffer);
    }

    // create a generic function for StringBuilder and StringBuffer
    private static <T extends Appendable & CharSequence> void testSeqMulti(T seq){
        // create a simple Runnable object: append 100 characters
        Runnable run = () -> {
            try {
                for (int i = 0; i < 100; i++) {
                    seq.append('a');
                }
            }
            catch (IOException e) {
                e.printStackTrace();
            }
        };

        // create 100 threads to run above Runnable object
        Thread[] td = new Thread[100];
        for (int i = 0; i < 100; i++) {
            td[i] = new Thread(run);
            td[i].start();
        }
        for (int i = 0; i < 100; i++) {
            try {
                td[i].join();
            }
            catch (InterruptedException ex) {
                ex.printStackTrace();
            }
        }

        // count the number of characters in the seq, expectation is 10000
        System.out.printf("Length of %s: %d, expect: 10000\n",
                seq.getClass().getSimpleName(), seq.length());
    }

}
```

The result is:

```
"C:\Program Files\Java\jdk1.8.0_221\bin\java.exe" ...
Length of StringBuilder: 9867, expect: 10000
Length of StringBuffer: 10000, expect: 10000
```

We can see the StringBuilder is incorrect under multi-thread mode. This is because the methods in StringBuilder is not synchronized while StringBuffer has synchronization.

3. Write an application to find out how many total characters can be held in a single StringBuilder before running out of memory, translate that number of characters to the number of bytes held by that StringBuilder before crashing.

Answer:
Duplicated with Problem 1.

4. Write an application to read a file with 10k lines of text, and output another file with the lines in sorted order.

Answer:

The code for the application is:

```
import java.io.*;
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;

public class App4 {
    public static void main(String[] args) throws IOException {
        FileReader fileReader = new FileReader("lorem.txt");
        BufferedReader bufferedReader = new BufferedReader(fileReader);
        List<String> lines = new ArrayList<>();
        String line = null;
        while ((line = bufferedReader.readLine()) != null) {
            lines.add(line);
        }
        bufferedReader.close();
        fileReader.close();
        Collections.sort(lines);

        FileWriter fileWriter = new FileWriter("lorem-sort.txt");
        for (String ln : lines) {
            fileWriter.write(ln);
            fileWriter.write('\n');
        }
        fileWriter.close();
    }
}
```

A snippet of the output file “lorem-sort.txt”:

```
A Morbi elit proin tincidunt!
A aliquam ante massa.
A ante sit venenatis?
A class suspendisse...
A iaculis duis.
A integre magnis sit.
Accumsan Eu cursus nostra et etiam luctus!
Accumsan Euismod consequat vel platea sodales augue enim: non iaculis suspendisse neque - posuere dapibus torquent.
Accumsan aliquam dapibus!
Accumsan augue aptent.
Accumsan commodo eros porttitor dui malesuada?
Accumsan cum Viverra metus egestas commodo odio dui lobortis.
Accumsan cum primis enim urna etiam.
Accumsan cum vel aenean.
Accumsan curabitur nascetur integre mauris morbi erat velit.
Accumsan cursus rhoncus...
Accumsan dignissim faucibus placerat et.
Accumsan eu Volutpat nascetur placerat tempor?
Accumsan eu lectus cursus primis leo mattis penatibus nam habitant tempus - ornare adipiscing et.
Accumsan euismod Sed curae.
Accumsan euismod consequat at lacinia fames ullamcorper urna - mi posuere.
Accumsan fringilla dictumst a blandit mi?
```

5. Write an application to read a file with 10k lines of text, and output another file with the lines in reverse sorted order.

Answer:

The code for the application is:

```
import java.io.*;
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;

public class App5 {
    public static void main(String[] args) throws IOException {
        FileReader fileReader = new FileReader("lorem.txt");
        BufferedReader bufferedReader = new BufferedReader(fileReader);
        List<String> lines = new ArrayList<>();
        String line = null;
        while ((line = bufferedReader.readLine()) != null) {
            lines.add(line);
        }
        bufferedReader.close();
        fileReader.close();
        lines.sort(Collections.reverseOrder());

        FileWriter fileWriter = new FileWriter("lorem-sort-reverse.txt");
        for (String ln : lines) {
            fileWriter.write(ln);
            fileWriter.write('\n');
        }
        fileWriter.close();
    }
}
```

A snippet of the output file “lorem-sort-reverse.txt”:

```
1  Vulputate urna erat ac!
2  Vulputate ultricies feugiat.
3  Vulputate sociis rhoncus...
4  Vulputate placerat cras porttitor dolor luctus.
5  Vulputate nam curae sit!
6  Volutpat viverra Elit mi odio tempus.
7  Volutpat vivamus Vel faucibus ut ornare arcu dolor...
8  Volutpat vel condimentum mi tempor?
9  Volutpat vehicula primis leo semper vulputate metus egestas habitant ornare.
10 Volutpat vehicula Sodales condimentum tempus nostra dui?
11 Volutpat tristique nam suspendisse nostra diam.
12 Volutpat risus vehicula sodales enim magnis penatibus sit ligula.
13 Volutpat risus amet neque.
14 Volutpat risus Leo vulputate fames class amet turpis.
15 Volutpat quisque facilisis faucibus pulvinar ac?
16 Volutpat primis nunc habitant nulla venenatis ligula, ac etiam...
17 Volutpat phasellus fermentum sed dis morbi potenti ullamcorper metus suspendisse amet, adipiscing venenatis diam malesuada...
18 Volutpat neque natoque.
19 Volutpat nascetur dis morbi porttitor nulla vitae.
20 Volutpat metus sociis natoque.
21 Volutpat integre etiam?
```

6. Write code to show exception handling including examples of checked, unchecked, and Error exceptions.

Answer:

The code for the application is:

```
import java.io.FileReader;

public class App6 {
    private static void handleException(Throwable th) {
        if (th instanceof Error) {
            System.out.println(th.getClass().getSimpleName() + " is an Error");
        }
        else if (th instanceof RuntimeException) {
            System.out.println(th.getClass().getSimpleName() + " is an unchecked exception");
        }
        else {
            System.out.println(th.getClass().getSimpleName() + " is a checked exception");
        }
    }
    public static void main(String[] args) {
        // an example of unchecked exception
        try {
            int a = Integer.parseInt("abc");
        }
        catch (Throwable t) {
            handleException(t);
        }

        // an example of checked exception
        try {
            FileReader reader = new FileReader("unreal.txt");
            reader.close();
        }
        catch (Throwable t) {
            handleException(t);
        }

        // an example of error
        try {
            int[] data = new int[1024*1024*1024];
        }
        catch (Throwable t) {
            handleException(t);
        }
    }
}
```

The output is:

```
"C:\Program Files\Java\jdk1.8.0_211\bin\java.exe" ..
NumberFormatException is an unchecked exception
FileNotFoundException is a checked exception
OutOfMemoryError is an Error
```

7. Write your own enum type. Describe when you would use it.

Answer:
The code for this problem ls:

```
public class App7 {
    enum Gender {
        MALE, FEMALE
    }

    public static void main(String[] args) {
        Gender g1 = Gender.FEMALE;
        Gender g2 = Gender.MALE;
        System.out.println(g1);
        System.out.println(g2);
    }
}
```

The output is:

```
"C:\Program Files\Java\jdk1.8.0_211\bin\java.exe" ..
FEMALE
MALE
```

This enum type can used for represent gender. It has two values, MALE and FEMALE. Many human resource systems contain this information for their employees. This kind of enum can be useful in those scenario.

Last one:
Write an application that uses the slf4j logging library directly (can also choose log4j if you want)

- Do the following:
    - configure the logging using an accepted department log statement format.
    - log at different logging levels (error, warn, info, debug), to see the effect of the default logging level setting.
    - turn on DEBUG in the logging config to show DEBUG output.
    - configure logging to go to **both** the console and a log file.

Answer:
The code is:

```
import org.apache.log4j.Logger;
import org.apache.log4j.PropertyConfigurator;

public class AppLog {
    static Logger LOGGER = Logger.getLogger(AppLog.class.getName());
    public static void main(String[] args) {
        PropertyConfigurator.configure("log4j.properties");

        LOGGER.info("This is an information message.");
        LOGGER.debug("This is a debug message.");
        LOGGER.warn("This is a warning message");
        LOGGER.error("This is an error message.");
        LOGGER.fatal("This is a fatal message");
    }
}
```

The log4j.properties configuration file is:

```
log4j.rootLogger=DEBUG,console,FILE
log4j.additivity.org.apache=true
#console
log4j.appender.console=org.apache.log4j.ConsoleAppender
log4j.appender.console.Threshold=DEBUG
log4j.appender.console.ImmediateFlush=true
log4j.appender.console.Target=System.out
log4j.appender.console.layout=org.apache.log4j.PatternLayout
log4j.appender.console.layout.ConversionPattern=%d{yyyy-MM-dd HH:mm:ss} [%p] %m%n
#console
log4j.appender.FILE=org.apache.log4j.FileAppender
log4j.appender.FILE.File=app.log
log4j.appender.FILE.ImmediateFlush=true
log4j.appender.FILE.Threshold=DEBUG
log4j.appender.FILE.Append=false
log4j.appender.FILE.layout=org.apache.log4j.PatternLayout
log4j.appender.FILE.layout.ConversionPattern=%d{yyyy-MM-dd HH:mm:ss} [%p] %m%n

log4j.logger.AppLog = debug
```

The output in the console is:

```
2019-08-24 04:06:42 [INFO] This is an information message.
2019-08-24 04:06:42 [DEBUG] This is a debug message.
2019-08-24 04:06:42 [WARN] This is a warning message
2019-08-24 04:06:42 [ERROR] This is an error message.
2019-08-24 04:06:42 [FATAL] This is a fatal message
```

The contents in the output log file is:

app.log

```
1  2019-08-24 04:06:42 [INFO] This is an information message.
2  2019-08-24 04:06:42 [DEBUG] This is a debug message.
3  2019-08-24 04:06:42 [WARN] This is a warning message
4  2019-08-24 04:06:42 [ERROR] This is an error message.
5  2019-08-24 04:06:42 [FATAL] This is a fatal message
6
```