Module 4 - Java
Exceptions, Apache Log4j, Regex

Advanced Java Certification Training

Akram M'Tir

1. A class method expects a list of prime integers. If any of the element in the list is not prime then a user defined checked exception for thrown.

- Write the class that expects a list of prime integers. Make appropriate choices of non-access modifiers.
- Write the user defined exception.

```java
public class PrimeInteger {

    public static void verifyPrime(int intArr[]) throws NotPrimeIntException {
        for(int i =0;i<intArr.length; i++) {
            if(!isPrime(intArr[i]))
                throw new NotPrimeIntException(intArr[i] + " at index " + i +
                        " is not a Prime number.");
        }
    }

    private static boolean isPrime(int n) {
        for(int i =2;i<=n/2;i++)
            if(n % i == 0) return false;
        return true;
    }

    public static void main(String[] args) {
        try {
            int arrInt[] = {11,13,17,13,29,31,37,41,43,53,59,61,67};
            PrimeInteger.verifyPrime(arrInt);
            System.out.println(Arrays.toString(arrInt));
            System.out.println("Array with Prime numbers Verified");
        }catch (NotPrimeIntException e) {
            e.printStackTrace();
        }
        catch (Exception e) {
            e.printStackTrace();
```

Problems | Javadoc | Declaration | Console | Debug

```
<terminated> PrimeInteger [Java Application] /usr/lib/jvm/java-1.8.0-openjdk-1.8.0.144-0.b01.el7_4.x8
[11, 13, 17, 13, 29, 31, 37, 41, 43, 53, 59, 61, 67]
Array with Prime numbers Verified
```

```java
public class PrimeInteger {

    public static void verifyPrime(int intArr[]) throws NotPrimeIntException {
        for(int i =0;i<intArr.length; i++) {
            if(!isPrime(intArr[i]))
                throw new NotPrimeIntException(intArr[i] + " at index " + i +
                        " is not a Prime number.");
        }
    }

    private static boolean isPrime(int n) {
        for(int i =2;i<=n/2;i++)
            if(n % i == 0) return false;
        return true;
    }

    public static void main(String[] args) {
        try {
            int arrInt[] = {6,11,13,17,13,29,31,37,41,43,53,59,61,67};
            PrimeInteger.verifyPrime(arrInt);
            System.out.println(Arrays.toString(arrInt));
            System.out.println("Array with Prime numbers Verified");
        }catch (NotPrimeIntException e) {
            e.printStackTrace();
        }
        catch (Exception e) {
            e.printStackTrace();
```

Problems @ Javadoc Declaration Console Debug

```
<terminated> PrimeInteger [Java Application] /usr/lib/jvm/java-1.8.0-openjdk-1.8.0.144-0.b01.el7_4.x86
module4.NotPrimeIntException: 6 at index 0 is not a Prime number.
        at module4.PrimeInteger.verifyPrime(PrimeInteger.java:10)
        at module4.PrimeInteger.main(PrimeInteger.java:24)
```

```java
package module4;

public class NotPrimeIntException extends Exception {

    private static final long serialVersionUID = -5807915051996596234L;

    public NotPrimeIntException(String message) {
        super(message);
    }

    @Override
    public String getMessage() {
        return super.getMessage();
    }

    @Override
    public String toString() {
        return super.toString();
    }

}
```

2. In the module4.Log4jDemo set the logging level to DEBUG and see what happens.

```java
3  import org.apache.log4j.Logger;
4
5  // Demonstrating the Log4j framework
6  public final class Log4jDemo {
7      // Loading the Log4j framework and initializing the framework
8      // with the values from log4j.properties files.
9      private static final Logger log4j =
10             Logger.getLogger(Log4jDemo.class.getName());
11
12     private static int sum(int intArr[]) {
13         log4j.trace("Entering sum(...)");
14         int val=0;
15         for (int i=0;i<intArr.length; i++) {
16             val += intArr[i];
17             log4j.debug("val= " + val);
18         }
19         log4j.trace("Returning from sum(...)");
20         return val;
21     }
22
23     public static void main(String[] args) {
24         log4j.info("Entering main(...)");
25         int intArr[] = {1,2,3,4,5,6,7,8,9,10};
26         int sum = Log4jDemo.sum(intArr);
27         log4j.debug("Final sum = " + sum);
28         log4j.info("Exiting main(...)");
29     }
30 }
```

Problems | Javadoc | Declaration | Console | Debug

```
<terminated> Log4jDemo [Java Application] /usr/lib/jvm/java-1.8.0-openjdk-1.8.0.144-0.b0
2018-05-06 14:15:55 INFO  module4.Log4jDemo:24 - Entering main(...)
2018-05-06 14:15:55 DEBUG module4.Log4jDemo:17 - val= 1
2018-05-06 14:15:55 DEBUG module4.Log4jDemo:17 - val= 3
2018-05-06 14:15:55 DEBUG module4.Log4jDemo:17 - val= 6
2018-05-06 14:15:55 DEBUG module4.Log4jDemo:17 - val= 10
2018-05-06 14:15:55 DEBUG module4.Log4jDemo:17 - val= 15
2018-05-06 14:15:55 DEBUG module4.Log4jDemo:17 - val= 21
2018-05-06 14:15:55 DEBUG module4.Log4jDemo:17 - val= 28
2018-05-06 14:15:55 DEBUG module4.Log4jDemo:17 - val= 36
2018-05-06 14:15:55 DEBUG module4.Log4jDemo:17 - val= 45
2018-05-06 14:15:55 DEBUG module4.Log4jDemo:17 - val= 55
2018-05-06 14:15:55 DEBUG module4.Log4jDemo:27 - Final sum = 55
2018-05-06 14:15:55 INFO  module4.Log4jDemo:28 - Exiting main(...)
```

```
1 # 1- Root logger option (provides core logging services)
2 #log4j.rootLogger=ALL, console, file
3 log4j.rootLogger=DEBUG, console, file
4 #log4j.rootLogger=INFO, console, file
5 #log4j.rootLogger=ERROR, console, file
6
7 # 2- Appender: destination where the logs are written. Redirect log messages to console
8 log4j.appender.console=org.apache.log4j.ConsoleAppender
9 log4j.appender.console.Target=System.out
10 # 3- Layout provides various layouts and formats like text-files, XML or HTML.
11 log4j.appender.console.layout=org.apache.log4j.PatternLayout
12 log4j.appender.console.layout.ConversionPattern=%d{yyyy-MM-dd HH:mm:ss} %-5p %c:%L - %m%n
13
14 # 2- Redirect log messages to a log file with support file rolling.
15 # If you don't want file rolling then change the class name to FileAppender.
16 log4j.appender.file=org.apache.log4j.RollingFileAppender
17 # The log file will be stored at the root folder i.e. Module_4_Assignments
18 log4j.appender.file.File=module4_log4j.log
19 log4j.appender.file.MaxFileSize=10KB
20 log4j.appender.file.MaxBackupIndex=2
21 log4j.appender.file.layout=org.apache.log4j.PatternLayout
22 log4j.appender.file.layout.ConversionPattern=%d{yyyy-MM-dd HH:mm:ss} %-5p %c:%L - %m%n
```

```
Open ▾                                   module4_log4j.log [Read-Only]
                                         ~/eclipse-workspace2/module4
40 2018-05-06 14:15:55 INFO  module4.Log4jDemo:24 - Entering main(...)
41 2018-05-06 14:15:55 DEBUG module4.Log4jDemo:17 - val= 1
42 2018-05-06 14:15:55 DEBUG module4.Log4jDemo:17 - val= 3
43 2018-05-06 14:15:55 DEBUG module4.Log4jDemo:17 - val= 6
44 2018-05-06 14:15:55 DEBUG module4.Log4jDemo:17 - val= 10
45 2018-05-06 14:15:55 DEBUG module4.Log4jDemo:17 - val= 15
46 2018-05-06 14:15:55 DEBUG module4.Log4jDemo:17 - val= 21
47 2018-05-06 14:15:55 DEBUG module4.Log4jDemo:17 - val= 28
48 2018-05-06 14:15:55 DEBUG module4.Log4jDemo:17 - val= 36
49 2018-05-06 14:15:55 DEBUG module4.Log4jDemo:17 - val= 45
50 2018-05-06 14:15:55 DEBUG module4.Log4jDemo:17 - val= 55
51 2018-05-06 14:15:55 DEBUG module4.Log4jDemo:27 - Final sum = 55
52 2018-05-06 14:15:55 INFO  module4.Log4jDemo:28 - Exiting main(...)
```

3. In the module4.Log4jDemo set the appender to a non-rolling file only (should not have console and rolling file).

```
1 # 1- Root logger option (provides core logging services)
2 #log4j.rootLogger=ALL, console, file
3 #log4j.rootLogger=DEBUG, console, file
4 log4j.rootLogger=DEBUG, file
5 #log4j.rootLogger=INFO, console, file
6 #log4j.rootLogger=ERROR, console, file
7
8 # 2- Appender: destination where the logs are written. Redirect log messages to console
9 #log4j.appender.console=org.apache.log4j.ConsoleAppender
10 #log4j.appender.console.Target=System.out
11 # 3- Layout provides various layouts and formats like text-files, XML or HTML.
12 #log4j.appender.console.layout=org.apache.log4j.PatternLayout
13 #log4j.appender.console.layout.ConversionPattern=%d{yyyy-MM-dd HH:mm:ss} %-5p %c:%L - %m%n
14
15 # 2- Redirect log messages to a log file with support file rolling.
16 # If you don't want file rolling then change the class name to FileAppender.
17 #log4j.appender.file=org.apache.log4j.RollingFileAppender
18 log4j.appender.file=org.apache.log4j.FileAppender
19 # The log file will be stored at the root folder i.e. Module_4_Assignments
20 log4j.appender.file.File=module4_log4j.log
21 #log4j.appender.file.MaxFileSize=10KB
22 #log4j.appender.file.MaxBackupIndex=2
23 log4j.appender.file.layout=org.apache.log4j.PatternLayout
24 log4j.appender.file.layout.ConversionPattern=%d{yyyy-MM-dd HH:mm:ss} %-5p %c:%L - %m%n
```

```
Open ▾                                   module4_log4j.log [Read-Only]   Save
                                         ~/eclipse-workspace2/module4
1 2018-05-06 14:23:46 INFO  module4.Log4jDemo:24 - Entering main(...)
2 2018-05-06 14:23:46 DEBUG module4.Log4jDemo:17 - val= 1
3 2018-05-06 14:23:46 DEBUG module4.Log4jDemo:17 - val= 3
4 2018-05-06 14:23:46 DEBUG module4.Log4jDemo:17 - val= 6
5 2018-05-06 14:23:46 DEBUG module4.Log4jDemo:17 - val= 10
6 2018-05-06 14:23:46 DEBUG module4.Log4jDemo:17 - val= 15
7 2018-05-06 14:23:46 DEBUG module4.Log4jDemo:17 - val= 21
8 2018-05-06 14:23:46 DEBUG module4.Log4jDemo:17 - val= 28
9 2018-05-06 14:23:46 DEBUG module4.Log4jDemo:17 - val= 36
10 2018-05-06 14:23:46 DEBUG module4.Log4jDemo:17 - val= 45
11 2018-05-06 14:23:46 DEBUG module4.Log4jDemo:17 - val= 55
12 2018-05-06 14:23:46 DEBUG module4.Log4jDemo:27 - Final sum = 55
13 2018-05-06 14:23:46 INFO  module4.Log4jDemo:28 - Exiting main(...)
```

4. Write a class RegexHelper where
The inputs will be the

- Regex expression
- Input string to parse

The inputs will be fetched from the standard input (console).
Use the try-with-resources block that we learnt in this module.
Handle the ParseSyntaxException .

```java

import java.util.Scanner;
import java.util.regex.Matcher;
import java.util.regex.Pattern;
import java.util.regex.PatternSyntaxException;

public final class RegexHelper {
    public static void main(String[] args) {
        // try-with-resources block. Will be implicitly/automatically closed for us
        // as long as these resources implement the autocloseable interface.
        try(Scanner sc = new Scanner(System.in)){
            System.out.println("Enter Regex: ");
            Pattern pattern = Pattern.compile(sc.nextLine());
            System.out.println("Enter Regex: ");
            Matcher matcher = pattern.matcher(sc.nextLine());

            boolean found = false;
            while(matcher.find()) {
                System.out.println("Found text " + matcher.group() +
                        " starting at index=" + matcher.start() +
                        " and ending at index=" + matcher.end() );
                found = true;
            }

            if(!found)
                System.out.println("No match found.");
    // Handling unchecked exception (indicate a syntax error in a regular expression pattern)
        }catch (PatternSyntaxException e) {
            e.printStackTrace();
        }
    }
```

Problems @ Javadoc Declaration Console Debug

```
<terminated> RegexHelper [Java Application] /usr/lib/jvm/java-1.8.0-openjdk-1.8.0.144-0.b01.el7_4.x86_64/jre/bin/java (Ma
Enter Regex:
(Bob)|(\sA[a-z]*)
Enter Regex:
Is Bob back Bob bob? Is Alice awake? Tell Alice to get rAady? We Are going for A ride.
Found text Bob starting at index=3 and ending at index=6
Found text Bob starting at index=12 and ending at index=15
Found text  Alice starting at index=23 and ending at index=29
Found text  Alice starting at index=41 and ending at index=47
Found text  Are starting at index=64 and ending at index=68
Found text  A starting at index=78 and ending at index=80
```