Academic Year: 2023-24 **Semester: II** **Class: FYMCA**

Course Code: MC506 **Course Name: Java Programming**

Name: Durgesh Dilip Mandge **UCID : 2023510032**

Experiment No. 5

Date: 05-02-2024

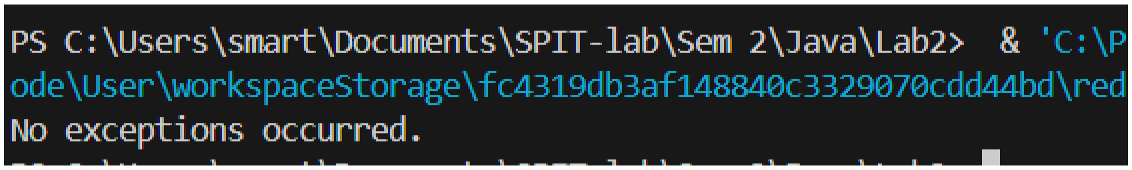
Aim: Fundamentals of Java Programming
CO Mapping – CO 1

Exception Handeling

```
class SpecialCharacterException extends Exception {
    public SpecialCharacterException(String message) {
        super(message);
    }
}

public class Main {
    public static void main(String[] args) {
        int a = 10;  // This is our reference number
        try {
            if (isPrime(a)) {
                throw new IllegalArgumentException("The given number is
prime.");
            }
            containsSpecialCharacter(Integer.toString(a));  // Convert 'a'
to string to check for special characters
            System.out.println("No exceptions occurred.");
        } catch (IllegalArgumentException iae) {
            System.out.println("Exception occurred due to prime number: "
+ iae.getMessage());
        } catch (SpecialCharacterException sce) {
            System.out.println("Exception occurred due to special
character: " + sce.getMessage());
```

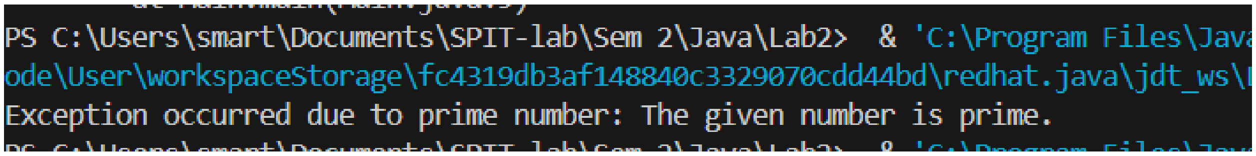
```
        } catch (Exception e) {
            System.out.println("An unexpected exception occurred: " +
e.getMessage());
        }
    }

    public static boolean isPrime(int number) {
        if (number < 2) {
            return false;
        }
        for (int i = 2; i <= Math.sqrt(number); i++) {
            if (number % i == 0) {
                return false;
            }
        }
        return true;
    }

    public static void containsSpecialCharacter(String str) throws
SpecialCharacterException {
        String specialCharacters = "!@#$%^&*()-+";
        for (int i = 0; i < str.length(); i++) {
            if
(specialCharacters.contains(Character.toString(str.charAt(i)))) {
                throw new SpecialCharacterException("String contains a
special character.");
            }
        }
    }
}
```

Output:



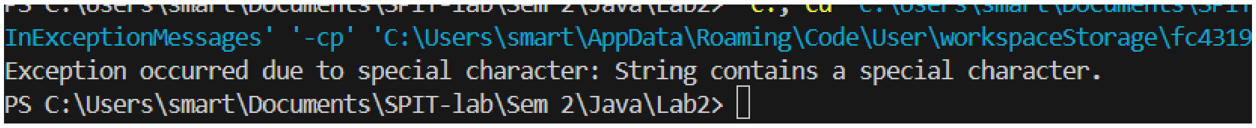
```
PS C:\Users\smart\Documents\SPIT-lab\Sem 2\Java\Lab2>  & 'C:\P
ode\User\workspaceStorage\fc4319db3af148840c3329070cdd44bd\red
No exceptions occurred.
```

For a=7:



```
PS C:\Users\smart\Documents\SPIT-lab\Sem 2\Java\Lab2>  & 'C:\Program Files\Java
ode\User\workspaceStorage\fc4319db3af148840c3329070cdd44bd\redhat.java\jdt_ws\
Exception occurred due to prime number: The given number is prime.
```

For string “-+”



```
InExceptionMessages' '-cp' 'C:\Users\smart\AppData\Roaming\Code\User\workspaceStorage\fc4319
Exception occurred due to special character: String contains a special character.
PS C:\Users\smart\Documents\SPIT-lab\Sem 2\Java\Lab2>
```

Multithreading

● WAP based on Multithreading which should do the following: Display the number of threads in use, Count the number of threads.

● Create 5 child threads. Join() function terminates the thread sequentially. Demonstrate join(). Check the status of liveness using isAlive().

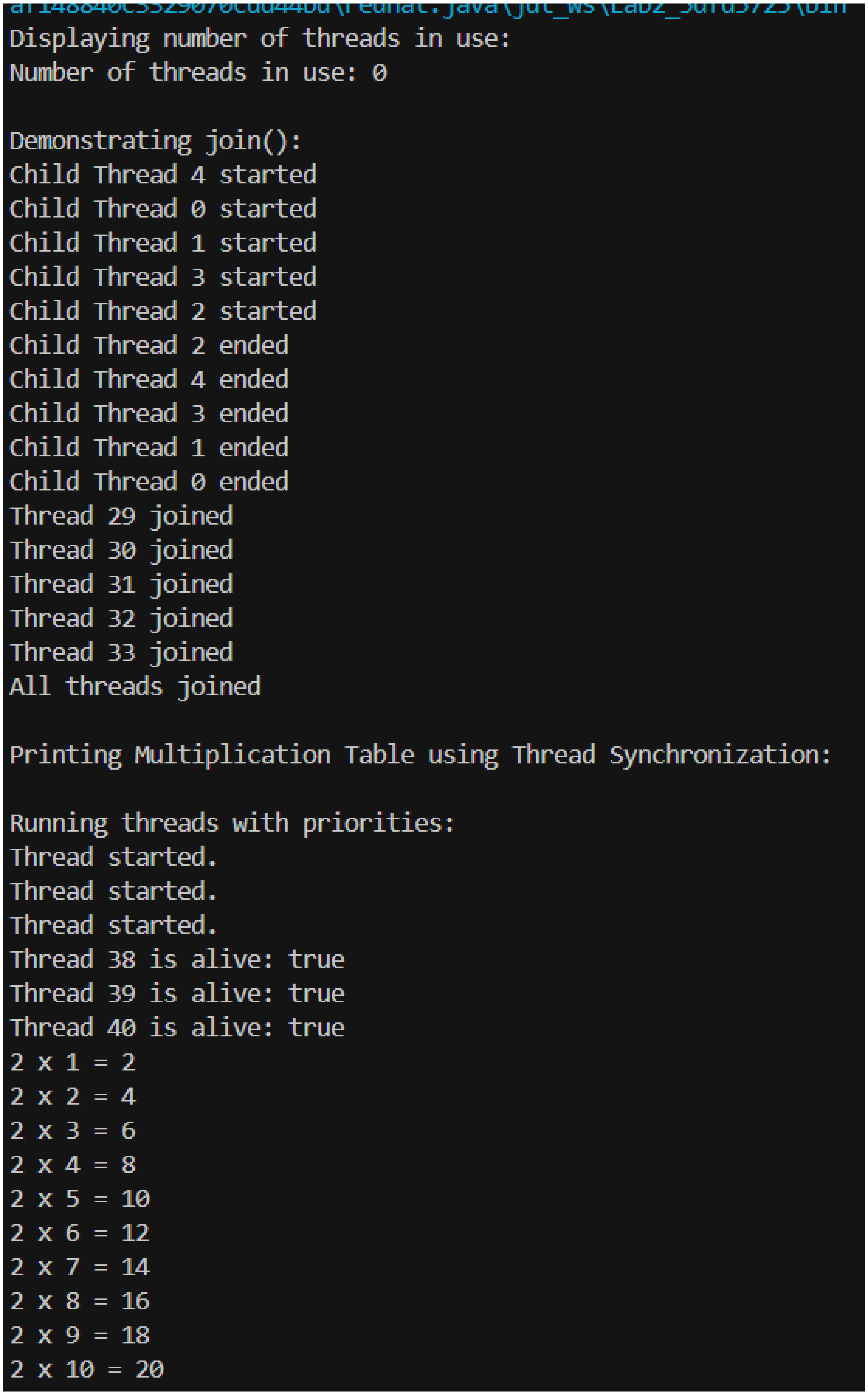
● W.A.P to print Multiplication Table using Thread Synchronization.

● Create 3 threads. Assign priority as per click operation provided in the running function. Assign first with max value. Check the status of the thread using isAlive(). Terminate the thread with proper message

Code:

```java
import java.util.concurrent.locks.Lock;
import java.util.concurrent.locks.ReentrantLock;

class ThreadCounter {
    private static int threadCount = 0;
    private static final Lock lock = new ReentrantLock();

    public static void displayThreadCount() {
        lock.lock();
        try {
            System.out.println("Number of threads in use: " + threadCount);
        } finally {
            lock.unlock();
        }
    }
    public static void countThread() {
        lock.lock();
        try {
            threadCount++;
        } finally {
            lock.unlock();
        }
    }
}

class ChildThread extends Thread {
    private int threadNum;

    public ChildThread(int threadNum) {
        this.threadNum = threadNum;
    }

    public void run() {
        System.out.println("Child Thread " + threadNum + " started");
        ThreadCounter.countThread();
        try {
            Thread.sleep(2000);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        System.out.println("Child Thread " + threadNum + " ended");
    }
}

public class ThreadMain {
    public static void main(String[] args) {
        System.out.println("Displaying number of threads in use:");
        ThreadCounter.displayThreadCount();
        System.out.println();

        System.out.println("Demonstrating join():");
        joinDemo();
        System.out.println();
        System.out.println("Printing Multiplication Table using Thread 
Synchronization:");
        printMultiplicationTable();
        System.out.println();

        System.out.println("Running threads with priorities:");
        runThreadsWithPriority();
    }

    public static void joinDemo() {
        Thread[] threads = new Thread[5];
        for (int i = 0; i < threads.length; i++) {
            threads[i] = new ChildThread(i);
            threads[i].start();
        }

        try {
            for (Thread thread : threads) {
                thread.join();
                System.out.println("Thread " + thread.getId() + " joined");
            }
        } catch (InterruptedException e) {
            e.printStackTrace();
        }

        System.out.println("All threads joined");
    }

    public static void printMultiplicationTable() {
        Lock lock = new ReentrantLock();
        for (int i = 2; i <= 5; i++) {
            final int table = i;
            new Thread(() -> {
                lock.lock();
                try {
                    for (int j = 1; j <= 10; j++) {
                        System.out.println(table + " x " + j + " = " + (table 
* j));
                    }
                } finally {
                    lock.unlock();
                }
            }).start();
        }
    }

    public static void runThreadsWithPriority() {
        Runnable runningFunction = () -> {
            System.out.println("Thread started.");
            while (Thread.currentThread().isAlive()) {
                try {
                    Thread.sleep(1000);
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
                System.out.println("Click detected.");
            }
            System.out.println("Thread terminated.");
        };

        Thread[] threads = new Thread[3];
        for (int i = 0; i < threads.length; i++) {
            threads[i] = new Thread(runningFunction);
            threads[i].setDaemon(true);
            threads[i].start();
        }

        threads[0].setPriority(Thread.MAX_PRIORITY);
        threads[1].setPriority(Thread.NORM_PRIORITY);
        threads[2].setPriority(Thread.MIN_PRIORITY);

        for (Thread thread : threads) {
            System.out.println("Thread " + thread.getId() + " is alive: " +
thread.isAlive());
        }

        for (Thread thread : threads) {
            try {
                thread.join();
                System.out.println("Thread " + thread.getId() + "
terminated.");
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}
```

Output



```
af148840c3329070cdd44bd\redhat.java\jdt_ws\Lab2_3dfd3723\bin
Displaying number of threads in use:
Number of threads in use: 0

Demonstrating join():
Child Thread 4 started
Child Thread 0 started
Child Thread 1 started
Child Thread 3 started
Child Thread 2 started
Child Thread 2 ended
Child Thread 4 ended
Child Thread 3 ended
Child Thread 1 ended
Child Thread 0 ended
Thread 29 joined
Thread 30 joined
Thread 31 joined
Thread 32 joined
Thread 33 joined
All threads joined

Printing Multiplication Table using Thread Synchronization:

Running threads with priorities:
Thread started.
Thread started.
Thread started.
Thread 38 is alive: true
Thread 39 is alive: true
Thread 40 is alive: true
2 x 1 = 2
2 x 2 = 4
2 x 3 = 6
2 x 4 = 8
2 x 5 = 10
2 x 6 = 12
2 x 7 = 14
2 x 8 = 16
2 x 9 = 18
2 x 10 = 20
```

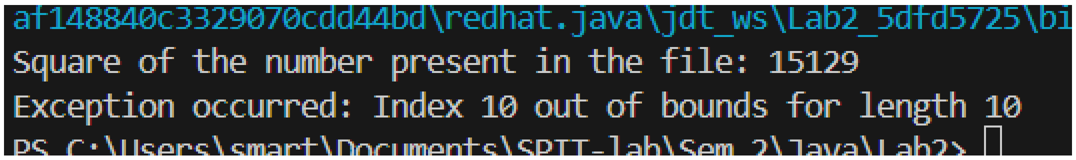
File Handling :

● Open the file "even.txt" read the number present in the file. Show the square of the number as output. If the number is not present show the exception.

● Build an array out of index for the data stored for integer numbers from 1 to 10 by accessing the 11th element from the index

```java
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;

public class File {
    public static void main(String[] args) {
        try {
            readAndSquareNumber("even.txt");
        } catch (IOException e) {
            System.out.println("Exception occurred: " + e.getMessage());
        }

        buildArrayOutOfIndex();
    }

    public static void readAndSquareNumber(String fileName) throws IOException
{
        try (BufferedReader reader = new BufferedReader(new
FileReader(fileName))) {
            String line = reader.readLine();
            if (line == null) {
                throw new IOException("File is empty");
            }
            int number = Integer.parseInt(line);
            System.out.println("Square of the number present in the file: " +
(number * number));
        } catch (NumberFormatException e) {
            throw new IOException("Invalid number format in the file");
        }
    }

    public static void buildArrayOutOfIndex() {
        int[] array = new int[10];
        for (int i = 1; i <= 10; i++) {
            array[i - 1] = i;
        }
        try {
            // Accessing the 11th element from the index (which doesn't exist)
            int value = array[10];
            System.out.println("Value at index 10: " + value);
        } catch (ArrayIndexOutOfBoundsException e) {
            System.out.println("Exception occurred: " + e.getMessage());
        }
    }
}
```

Output



```
af148840c3329070cdd44bd\redhat.java\jdt_ws\Lab2_5dfd5725\bi
Square of the number present in the file: 15129
Exception occurred: Index 10 out of bounds for length 10
PS C:\Users\smart\Documents\SPIT-lab\Sem 2\Java\Lab2>
```

OBSERVATIONS:

Through the Java exercises I learnt a breadth of topics, from basic syntax to advanced object oriented principles. Each question posed unique challenges, refining coding skills and deepening understanding of Java concepts. Through troubleshooting and experimentation, practical experience in writing efficient Java code was gained. These exercises established a solid foundation for tackling real-world programming tasks and instilled confidence for exploring complex Java projects in the future.