Program Construction - CS1040

Concurrency Lab Exercise 2

❖ Group Name: **CodegenX**

- 220049E-Aththanayaka.D.H
- 220075E-Botheju.W.P.B
- 220079U-Chamara.K.K
- 220165F-Fernando.K.A.E.M

➢ Introduction

- We are transforming the printing experience at ShinePrinters. Customers may handle print tasks seamlessly and effectively using our Java-designed networked solution. Our system, which has three PCs and two printers linked via a shared queue, is designed to process and handle papers quickly. A multi-threaded producer-consumer paradigm that is robust has been created in order to prevent consistency issues and maintain high throughput. Furthermore, we're advancing printing services by introducing a web interface for remote print job submission, supported by intelligent file type management to ensure compatibility and ease of use.

✓ **Source Files**

1. PrintJob Class

```
 public class PrintJob {
private String jobId;
private String fileType;

public PrintJob(String jobId, String fileType) {
    this.jobId = jobId;
    this.fileType = fileType;
}

// Getters
public String getJobId() {
    return jobId;
}

public String getFileType() {
```

```java
        return fileType;
    }

}

```

02. Computer Class

```java
import java.util.ArrayList;

public class Computer extends Thread {
    private final SharedQueue sharedQueue;
    private static ArrayList<PrintJob> job_list;
    private String compuetrID;

    public Computer(SharedQueue sharedQueue, String id, ArrayList<PrintJob> job_list) {
        this.sharedQueue = sharedQueue;
        this.compuetrID = id;
        this.job_list = job_list;
    }

    @Override
    public void run() {
        // Example: Creating a new print job. In practice, this would be more dynamic.
        while (!job_list.isEmpty()) {
            PrintJob job = job_list.remove(0);
            try {
                sharedQueue.enqueue(job, this.compuetrID);
                // Thread.sleep(100);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
            try {
                Thread.sleep(50);
            } catch (Exception e) {
                // TODO: handle exception
            }
        }

    }

}

```

03.Printer Class

```java
public class Printer extends Thread {
    private final SharedQueue sharedQueue;
    private int PrinterID;

    public Printer(SharedQueue sharedQueue, int ID) {
        this.sharedQueue = sharedQueue;
        this.PrinterID = ID;
    }

    public int GetID() {
        return PrinterID;
    }

    public static void checkType(String type) throws TypeNotSupportedException {
        if (type.equalsIgnoreCase("pdf")) {

        } else {
            throw new TypeNotSupportedException("This File is not support");
        }
    }

    @Override
    public void run() {
        while (!Thread.currentThread().isInterrupted()) {
            try {
                PrintJob job = sharedQueue.dequeue();
                try {
                    checkType(job.getFileType());
                    Thread.sleep(2000);
                    System.out.println("printer " + this.GetID() + " ---> Processing ---> " + job.getJobId());
                } catch (TypeNotSupportedException e) {
                    System.out.println(e);
                    System.out.println("!!!!" + job.getJobId() + " file is not supported" + "!!!!");
                    System.out.println();

                } catch (Exception ex) {
                    ex.printStackTrace();
                }

                // Add sleep to simulate job processing time

            } catch (InterruptedException e) {
                Thread.currentThread().interrupt(); // Properly handle interruption
                break;
            }
        }
    }
}

```

04. SharedQueue Class

```java
import java.util.LinkedList;
import java.util.Queue;

public class SharedQueue {
    public final Queue<PrintJob> queue = new LinkedList<>();
    private final int capacity = 5;

    public synchronized void enqueue(PrintJob job, String idnum) throws InterruptedException {
        while (queue.size() == capacity) {
            System.out.println("Queue is Full. Wait until printers are free...");
            wait();
        }
        System.out.println("Computer " + idnum + " ---> Enqueued ---> " + job.getJobId());
        queue.add(job);
        notifyAll();
    }

    public synchronized PrintJob dequeue() throws InterruptedException {
        while (queue.isEmpty()) {
            Thread.sleep(3000);
            System.out.println("waiting for order...");
        }
        PrintJob job = queue.poll();
        System.out.println("Dequeued " + job.getJobId());

        notifyAll();
        return job;
    }

    public Queue<PrintJob> GetQueue() {
        return this.queue;
    }
}

```

05. TypeNotSupportedException Class

```
public class TypeNotSupportedException extends Exception {
    public TypeNotSupportedException(String s) {
        super(s);
    }
}


```

06. Main Class

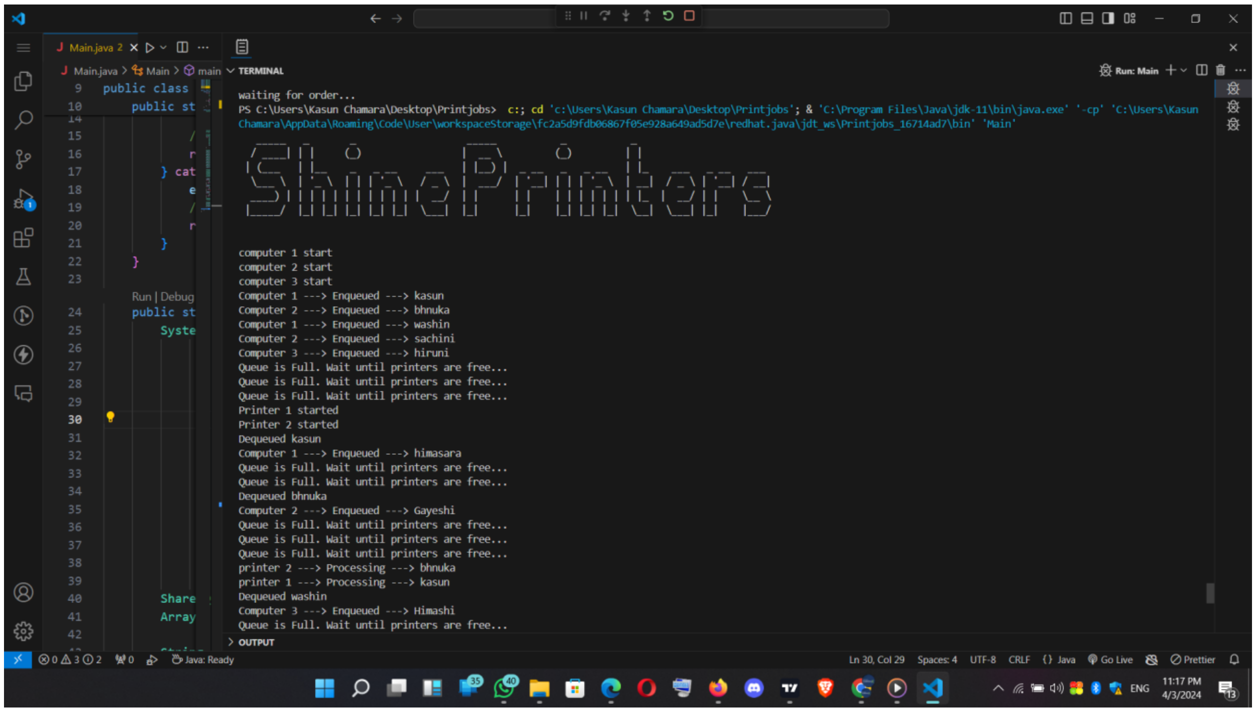
```
import java.util.ArrayList;
import java.util.List;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.io.IOException;
import java.util.stream.Collectors;
import java.util.stream.Stream;

public class Main {
    public static ArrayList<String> readLinesAsArrayList(String filePath) {
        try {
            // Read all lines from the file as a List
            List<String> lines = Files.readAllLines(Paths.get(filePath));

            // Convert List to ArrayList
            return new ArrayList<>(lines);
        } catch (IOException e) {
            e.printStackTrace();
            // Return an empty ArrayList in case of an error
            return new ArrayList<>();
        }
    }

    public static void main(String[] args) {
        System.out.println(
                "   _____   _       _                  _____           _           _
\r\n" + //
                        "  / ___| | |     ()                 |  __ \\         (_)         | |
\r\n"
                        + //
```

```java
                        " | (___   | |__    _   _ __     ___   | |) |  _ __   _   _ __   | |_ ___   _ __   ___ \r\n"
                        + //
                        "  \\___ \\  | '_ \\  | | | '_ \\   / _ \\  |  __/  | '| | | | ' \\  | _|  / _ \\ | '_| / __|\r\n"
                        + //
                        "  ___) | | | | | | | | | | | |  __/  | |      | |    | | | | | | | | |  _/ | |    \\_ \\\r\n"
                        + //
                        " |/  || || || || ||  \\|  ||      ||    || || ||  \\|  \\| || |_/\r\n"
                        + //
                        " \r\n"
                        + //
                        " ");
        SharedQueue sharedQueue = new SharedQueue();
        ArrayList<PrintJob> job_list = new ArrayList<>();

        String filePath = "textfile.txt";
        ArrayList<String> lines = readLinesAsArrayList(filePath);
        for (int i = 0; i < lines.size(); i++) {
            String filename = lines.get(i);
            String[] parts = filename.split("\\.");
            String s1 = parts[0];
            String s2 = parts[1];
            PrintJob obj = new PrintJob(s1, s2);
            job_list.add(obj);

        }

        PrintJob job1 = new PrintJob("Himashi", "pdf");
        job_list.add(job1);
        PrintJob job2 = new PrintJob("Gayeshi", "pdf");
        job_list.add(job2);
        PrintJob job3 = new PrintJob("Kavindya", "pdf");
        job_list.add(job3);
        PrintJob job4 = new PrintJob("Yashodara", "jpg");
        job_list.add(job4);
        PrintJob job5 = new PrintJob("Shakeena", "pdf");
        job_list.add(job5);
        PrintJob job6 = new PrintJob("Ravishna", "ser");
        job_list.add(job6);
        PrintJob job7 = new PrintJob("Nirasha", "pdf");
        job_list.add(job7);

        Thread computer1 = new Computer(sharedQueue, "1", job_list);
        System.out.println("computer 1 start");
        Thread computer2 = new Computer(sharedQueue, "2", job_list);
        System.out.println("computer 2 start");
        Thread computer3 = new Computer(sharedQueue, "3", job_list);
        System.out.println("computer 3 start");

        computer1.start();
        try {
            Thread.sleep(50);
        } catch (Exception e) {
        }
        computer2.start();
        try {
            Thread.sleep(50);
        } catch (Exception e) {
        }
        computer3.start();
        try {
            Thread.sleep(50);
        } catch (Exception e) {
        }
        Thread printer1 = new Printer(sharedQueue, 1);
        Thread printer2 = new Printer(sharedQueue, 2);

        System.out.println("Printer 1 started");
        printer1.start();
        System.out.println("Printer 2 started");
        printer2.start();

        // TODO: handle exception
    }
}

```

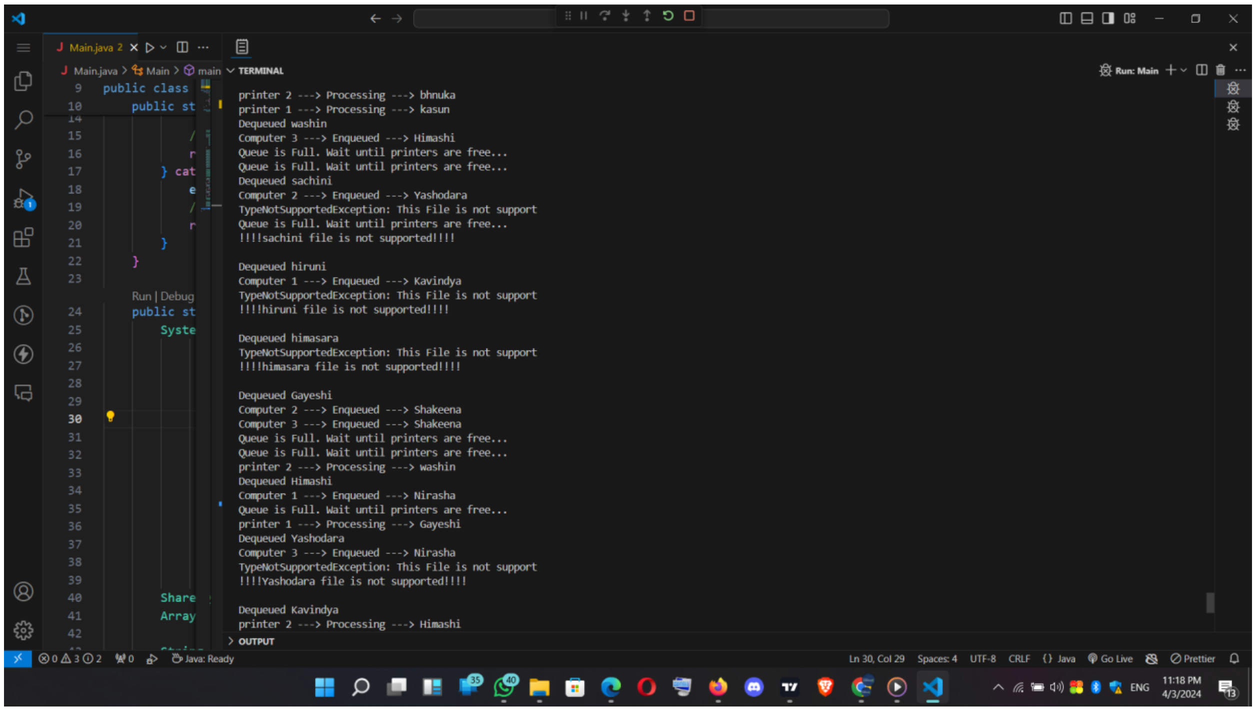
07. Text File

```
kasun.pdf
bhnuka.pdf
washin.pdf
sachini.jpg
hiruni.mp4
himasara.png

```

08. output Screenshot



```
waiting for order...
PS C:\Users\Kasun Chamara\Desktop\Printjobs>  c:; cd 'c:\Users\Kasun Chamara\Desktop\Printjobs'; & 'C:\Program Files\Java\jdk-11\bin\java.exe' '-cp' 'C:\Users\Kasun Chamara\AppData\Roaming\Code\User\workspaceStorage\fc2a5d9fdb06867f05e928a649ad5d7e\redhat.java\jdt_ws\Printjobs_16714ad7\bin' 'Main'
   _____ _     _              _____      _       _
  / ____| |   (_)            |  __ \    (_)     | |
 | (___ | |__  _ _ __   ___  | |__) | __ _ _ __ | |_ ___ _ __ ___
  \___ \| '_ \| | '_ \ / _ \ |  ___/ '__| | '_ \| __/ _ \ '__/ __|
  ____) | | | | | | | |  __/ | |   | |  | | | | | ||  __/ |  \__ \
 |_____/|_| |_|_|_| |_|\___| |_|   |_|  |_|_| |_|\__\___|_|  |___/


computer 1 start
computer 2 start
computer 3 start
Computer 1 ---> Enqueued ---> kasun
Computer 2 ---> Enqueued ---> bhnuka
Computer 1 ---> Enqueued ---> washin
Computer 2 ---> Enqueued ---> sachini
Computer 3 ---> Enqueued ---> hiruni
Queue is Full. Wait until printers are free...
Queue is Full. Wait until printers are free...
Queue is Full. Wait until printers are free...
Printer 1 started
Printer 2 started
Dequeued kasun
Computer 1 ---> Enqueued ---> himasara
Queue is Full. Wait until printers are free...
Queue is Full. Wait until printers are free...
Dequeued bhnuka
Computer 2 ---> Enqueued ---> Gayeshi
Queue is Full. Wait until printers are free...
Queue is Full. Wait until printers are free...
Queue is Full. Wait until printers are free...
printer 2 ---> Processing ---> bhnuka
printer 1 ---> Processing ---> kasun
Dequeued washin
Computer 3 ---> Enqueued ---> Himashi
Queue is Full. Wait until printers are free...
```



```
printer 2 ---> Processing ---> bhnuka
printer 1 ---> Processing ---> kasun
Dequeued washin
Computer 3 ---> Enqueued ---> Himashi
Queue is Full. Wait until printers are free...
Queue is Full. Wait until printers are free...
Dequeued sachini
Computer 2 ---> Enqueued ---> Yashodara
TypeNotSupportedException: This File is not support
Queue is Full. Wait until printers are free...
!!!!sachini file is not supported!!!!

Dequeued hiruni
Computer 1 ---> Enqueued ---> Kavindya
TypeNotSupportedException: This File is not support
!!!!hiruni file is not supported!!!!

Dequeued himasara
TypeNotSupportedException: This File is not support
!!!!himasara file is not supported!!!!

Dequeued Gayeshi
Computer 2 ---> Enqueued ---> Shakeena
Computer 3 ---> Enqueued ---> Shakeena
Queue is Full. Wait until printers are free...
Queue is Full. Wait until printers are free...
printer 2 ---> Processing ---> washin
Dequeued Himashi
Computer 1 ---> Enqueued ---> Nirasha
Queue is Full. Wait until printers are free...
printer 1 ---> Processing ---> Gayeshi
Dequeued Yashodara
Computer 3 ---> Enqueued ---> Nirasha
TypeNotSupportedException: This File is not support
!!!!Yashodara file is not supported!!!!

Dequeued Kavindya
printer 2 ---> Processing ---> Himashi
```