CS1040 – Section 3 - Lab Exercise 2

Cybrox Team Members

Name	Index
Withanage W.I.N.	220731M
Jayasinghe M.M.S.	220263E
Hendalage D.S.D.	220222E
Ambepitiya D.C.H	220028N

Source File

```
import java.io.FileNotFoundException;
import java.util.Queue;
import java.util.LinkedList;
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;

public class Main {
    public static void main(String[] args){
        SharedQueue sharedQueue=new SharedQueue(5);
        //Hardcode the inputs for the running program
        Queue<Works> queue = new LinkedList<>();
        queue.offer(new Works("Sended","Essay1",4,"docx"));
        queue.offer(new Works("Sended","Essay2",4,"txt"));
        queue.offer(new Works("Sended","Essay3",4,"pdf"));
        queue.offer(new Works("Sended","Essay11",4,"txt"));
        queue.offer(new Works("Sended","Essay12",4,"txmt"));
        queue.offer(new Works("Sended","Essay13", 4, "txt"));

        //Making Computers threads
        Thread Computer1 = new Thread(new Computer(1,sharedQueue,queue),
"Printer2");
        Computer1.start();
        Thread Computer2 = new Thread(new Computer(2,sharedQueue,queue),
"Printer2");
        Computer2.start();
        Thread Computer3 = new Thread(new Computer(3,sharedQueue,queue),
"Printer2");
        Computer3.start();

        //Making Printer Threads
        new Thread(new Printer(1,sharedQueue),"Printer1").start();
        new Thread(new Printer(2,sharedQueue),"Printer2").start();
        /*
        if someone sent print job through online
        first get the text file
        next add the job directly to the SharedQueue object
         */
        String path="test.txt";//for an example
        TextFile mytxt=ReadAFile(path);
    }
    //we will use this if someone sent a print job get the text
    public static TextFile ReadAFile(String filePath){
```

```
        StringBuilder content=new StringBuilder();
        BufferedReader reader= null;
        try {
            reader = new BufferedReader(new FileReader(filePath));

            String line;
            while ((line = reader.readLine()) != null) {
                content.append(line);
                content.append("\n");
            }
            reader.close();
        }catch (Exception e){
            System.out.println(e.getMessage());
        }
        return new TextFile(content.toString());
    }
}
// This is not necessary for system.This use to hardcoded part only for
simulate.
class Works{
    private String workType; // Online sent document or computer created
one
    private String name;  //Name of the document
    private int numberOfCopies;  //Required number of copies.
    private String fileType;   // PDF,TXT,Doc type

    public Works(String workType,String name,int numberOfCopies,String
fileType){
        this.workType=workType;
        this.name=name;
        this.numberOfCopies=numberOfCopies;
        this.fileType=fileType;
    }

    public String getWorkType(){
        return workType;
    }

    public String getName(){
        return name;
    }

    public int getNumberOfCopies(){
        return numberOfCopies;
    }

    public String getFileType(){
        return fileType;
    }
}
//create job object to print for printers
class PrintJob{
    private String name;
    private int numberOfCopies;
    private String fileType;

    public PrintJob(String name,int numberOfCopies,String fileType){
        this.name=name;
        this.numberOfCopies=numberOfCopies;
        this.fileType=fileType;
    }
```

```
    public String getName(){
        return name;
    }

    public int getNumberOfCopies(){
        return numberOfCopies;
    }

    public String getFileType(){
        return fileType;
    }
}
//This is the path for send printing jobs to the printer.
class SharedQueue{
    private int size;
    private Queue<PrintJob> queue=new LinkedList<>();

    public SharedQueue(int size){
        this.size=size;
    }
    // Synchronized this methods for data protection.
    public synchronized void add(PrintJob printJob) throws
InterruptedException{
        while (queue.size()>=size){
            wait();
        }
        queue.offer(printJob);
        notifyAll();
    }
    // Synchronized this methods for data protection.
    public synchronized PrintJob get() throws InterruptedException{
        while(queue.isEmpty()){
            wait();
        }
        PrintJob printJob=queue.poll();
        notifyAll();
        return printJob;
    }
}

// Computers mainly do the creating print objects.
class Computer implements Runnable{
    private int computerNo;
    private SharedQueue sharedQueue;
    private Queue<Works> queue;

    public Computer(int computerNo,SharedQueue sharedQueue,Queue<Works>
queue){
        this.computerNo=computerNo;
        this.sharedQueue=sharedQueue;
        this.queue=queue;
    }

    @Override
    public void run(){
        while (true){
            try{
                Works tempWork;
                synchronized(queue){
                    while (queue.isEmpty()){
```

```
                        queue.wait();
                    }
                    tempWork=queue.poll();
                }
                //if file type don't match make an exception
                if (tempWork.getFileType().equals("pdf") ||
tempWork.getFileType().equals("txt") ||
tempWork.getFileType().equals("docx")) {
                    PrintJob myJob=new
PrintJob(tempWork.getName(),tempWork.getNumberOfCopies(),tempWork.getFileTy
pe());
                    sharedQueue.add(myJob);
                    System.out.println("Computer "+computerNo+" send
"+myJob.getName());
                    Thread.sleep(100);
                } else {
                    throw new TypeNotSupportedException("Invalid File
Type!!");
                }
            } catch (InterruptedException | TypeNotSupportedException e) {
                System.out.println(e.getMessage());
            }
        }
    }
}
//This is mainly printer class.This one print the given printjobs.
class Printer implements Runnable {
    private int printerNo;
    private SharedQueue sharedQueue;

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

    @Override
    public void run(){
        try{
            Thread.sleep(10);
        }catch(InterruptedException e){
        }
        while (true){
            try{
                PrintJob temp=sharedQueue.get();
                if (temp !=null) {
                    System.out.println("Printer "+printerNo + " printed
"+temp.getName());
                    Thread.sleep(100);
                }
                //if file type don't match make an exception
                if ((temp.getFileType().equals("pdf") ||
temp.getFileType().equals("txt") || temp.getFileType().equals("docx"))) {
                    throw new TypeNotSupportedException("Invaled File
Type!!");
                }

            }
            catch (InterruptedException e) {
                e.printStackTrace();
            }catch (TypeNotSupportedException e1){
```

```
                }
            }
        }
}
// File type error exception handling.
class TypeNotSupportedException extends Exception {
    public TypeNotSupportedException(String errorMessage) {
        super(errorMessage);
    }
}
//This is for online printing requirements.
class TextFile {
    private String content;
    public TextFile(String content) {
        this.content=content;
    }
    public String getContent() {
        return content;
    }
}
```

Example Output Screenshot