CS1040 – Section 3 - Lab Exercise 2

PrintJob.java

```
public class PrintJob{
    private TextFile textFile;
    private String fileType;

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

    public TextFile getContent() {
        return textFile;
    }

    public String getFileType() {
        return fileType;
    }
}
```

Computer.java

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

public class Computer implements Runnable {
    private SharedQueue queue;
    private int ID;
    private String fileName;

    public Computer(int ID,String fileName,SharedQueue queue) {
        this.ID = ID;
        this.fileName=fileName;
        this.queue=queue;
    }


    @Override
```

```
    public void run() {
        TextFile textFile=readAFile(fileName,"txt");

        PrintJob job=new PrintJob(textFile, "txt");

        queue.addPrintJob(job);

    }



    public TextFile readAFile(String fileName, String fileType) {
        try {
            if (fileName.endsWith(".txt")) {
                FileReader file = new FileReader(fileName);
                BufferedReader br = new BufferedReader(file);
                StringBuilder sb = new StringBuilder();
                String line;

                try {
                    while ((line = br.readLine()) != null) {
                        sb.append(line);
                        sb.append(System.lineSeparator());
                    }
                } catch (IOException e) {
                    e.printStackTrace();
                }
                TextFile textfile=new TextFile(sb.toString());
                return textfile;
            } else {
                throw new TypeNotSupportedException("Unsupported file type
added: " + fileName);
            }
        } catch (FileNotFoundException e) {
            System.err.println("file not found: " + e.getMessage());
        } catch (TypeNotSupportedException e) {
            System.err.println("Error: " + e.getMessage());
        }

        return null; // or return empty string, depending on your requirement
    }

}
```

Printer.java

```
public class Printer extends Thread {
    private int ID;
    private SharedQueue queue;

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

    @Override
    public void run() {
        processPrintJob();
    }

    public void processPrintJob() {
        PrintJob job = queue.getPrintJob();
        if (job != null) {
            try {
                if (job.getFileType().equals("txt")) {
                    System.out.println("Printing job on Printer ");
                    Thread.sleep(2000); // Simulate printing time
                } else {
                    throw new TypeNotSupportedException("Unsupported file type
printed: " + job.getFileType());
                }
            } catch (InterruptedException e) {
                Thread.currentThread().interrupt();
                System.err.println("Error during printing: " +
e.getMessage());
            } catch (TypeNotSupportedException e) {
                System.err.println("Error: " + e.getMessage());
            }
        }
    }

}
```

SharedQueue.java

```
import java.util.concurrent.ArrayBlockingQueue;
import java.util.concurrent.BlockingQueue;
import java.util.concurrent.LinkedBlockingQueue;

public class SharedQueue {
    private BlockingQueue<PrintJob> queue;

    public SharedQueue(int capacity) {
        queue = new ArrayBlockingQueue<>(capacity);
    }

    public void addPrintJob(PrintJob job) {
        while (queue.size() >= 5) {
            try {
                System.out.println("Queue is full, waiting for space...");
                Thread.sleep(1000); // Sleep for 1 second
            } catch (InterruptedException e) {
                Thread.currentThread().interrupt();
                System.err.println("Error while waiting for space in the
queue: " + e.getMessage());
            }
        }

        if (job != null) {
            try {
                queue.put(job);
                System.out.println("Successfully added printjob to the
queue");
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        } else {
            System.err.println("Attempted to add a null print job to the
queue.");
        }
    }

    public PrintJob getPrintJob() {
        try {
            return queue.take();
        } catch (InterruptedException e) {
            Thread.currentThread().interrupt();
```

```
            System.err.println("Error retrieving print job from the queue: " +
e.getMessage());
            return null;
        }
    }

}
```

TextFile.java

```
public class TextFile {
    private String content;

    public TextFile(String content){
        this.content=content;
    }
}
```

Main.java

```
public class Main {

    public static void main(String[] args) {
        SharedQueue queue = new SharedQueue(5);

        Thread thread1=new Thread(new Computer(1,"text1.txt",queue));
        Thread thread2=new Thread(new Computer(2,"text2.txt",queue));
        Thread thread3=new Thread(new Computer(2,"text3.pdf",queue));

        thread1.start();
        thread2.start();
        thread3.start();

        Printer printer1 = new Printer(queue,1);
        Printer printer2 = new Printer(queue,2);

        printer1.start();
        printer2.start();
    }

}
```

Group Name: Code Raptors

Group members:

- 220023U – Amarasinghe A.A.D.H.S.
- 220503R – Ranasinghe P.I.
- 220520P – Rathnayaka R.L.M.P.
- 220548H – Samarakoon E.S.P.A.