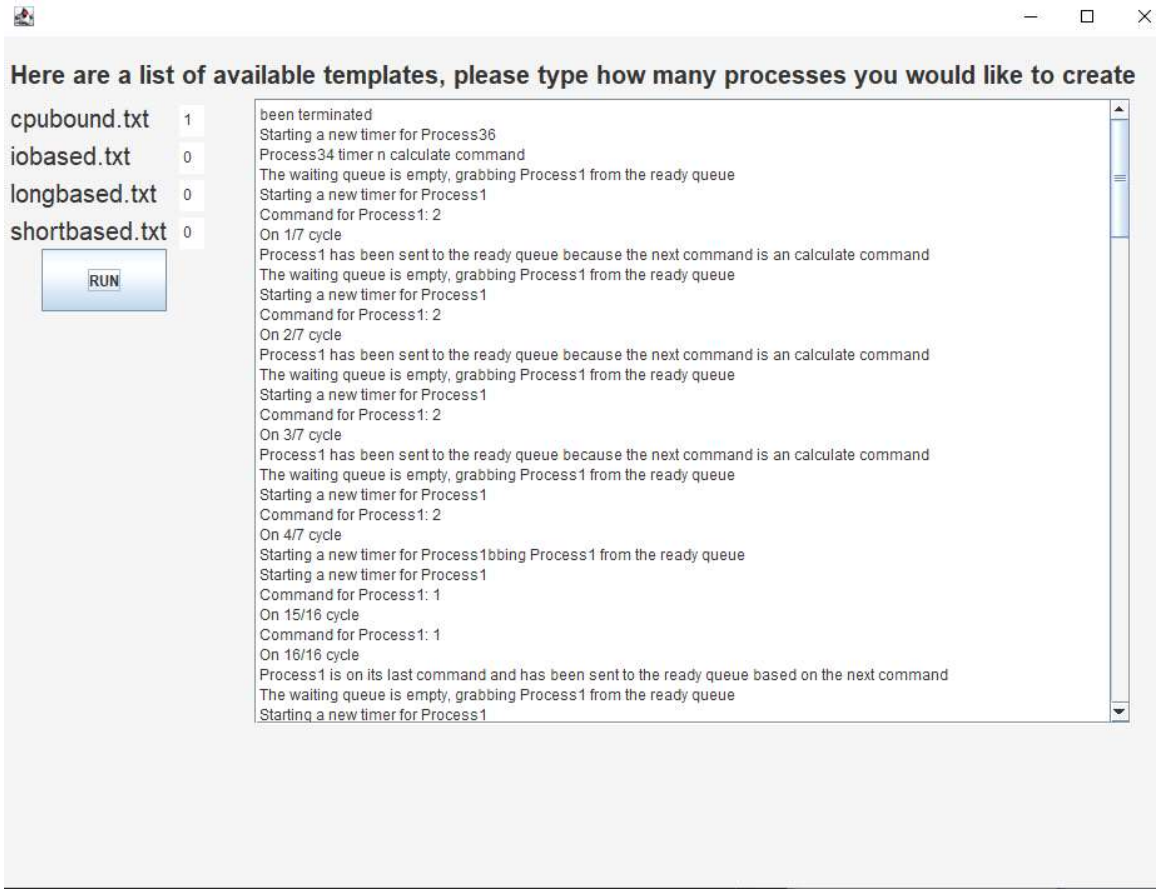
Github url:https://github.com/huhu72/Project-1
commit ID:cc13af1847b9328cec23b5ef42e1ebfd6d94e558

```
            Dispatcher.setPCBList(cpu.getComparePCBList());
            Dispatcher.setReadyQueue(cpu.getCompareQueue());

            int RRCycles = cpu.compareRR();
            cpu.compareQueue = new LinkedList<Process>();
            CPU.comparePCBList = new HashMap<>();
            try {
                p.createCompareProcesses();
            } catch (FileNotFoundException e1) {
                // TODO Auto-generated catch block
                e1.printStackTrace();
            }
            Dispatcher.setPCBList(cpu.getComparePCBList());
            Dispatcher.setReadyQueue(cpu.getCompareQueue());
            int PQCycles = cpu.comparePQ();
            if (RRCycles < PQCycles) {
                CPU.scheduler = "PQ";

            } else {
                CPU.scheduler = "RR";

            }
            cpu = new CPU();
            p = new Process(cpu);

            try {
                p.createProcessesPrompt(templateNums);
            } catch (FileNotFoundException e1) {

                e1.printStackTrace();
            }
            // statusThread.start();
            Dispatcher.setPCBList(cpu.getPCBList());
            Dispatcher.setReadyQueue(cpu.getJobQueue());

            Semaphore s1 = new Semaphore();
            Semaphore s2 = new Semaphore();
            cpu.setSemaphore(s1);
            cpu2.setSemaphore(s2);
            cpu.start();
            cpu2.start();
        } else {
            CPU.status = true;
            cpu.print();
```



- Added a **GUI** using JFrame. User can specify how many of each proccess they want from the provided templates in the input box. If the user doesnt add any values to it, it will automatically put 0 in the input so that its easier to collect data from it.
- Also added live feed in the GUI using the JFrameTextPane and JFrameScroll so that the users can scroll
- Multievel Program is created based off the old template except there are two semaphores and two CPUs with 4 threads each. The two CPUs have a semaphore of their own and access to it is still the same. Separating the semaphores prevent the second cpu from locking the first. Threads are still block from trying to aquire the semaphore if another thread currently holds it therefore **preventing a deadlock**

```java
//Implements multi level child parent relationship
    public Process fork() throws FileNotFoundException {
        int min = 1;
        int max = 500;
        int randomNum = (int) Math.floor(Math.random() * (max - min + 1) + min);
        createCommands("child.txt");
        Process childProcess = new Process("Process" + processCreationCounter, getCommands(),
                (this.pid + this.pidCounter), this.critStart, this.critEnd);
        childProcess.setParentPID(this.getPID());
        this.pidCounter++;
        this.processCreationCounter++;
        if (randomNum == 1) {
            Process grandChildProcess = fork();
            childProcess.setChildPID(grandChildProcess.getPID());
        }

        this.cpu.addToProcessQueue(childProcess);
        PCB childPCB = new PCB(childProcess);
        childPCB.setParentPID(childProcess.getParentPID());
        CPU.updatePCBList(childProcess, childPCB);
        return childProcess;
    }
```

```java
if (!semaphore.list.contains(pcb.getProcess())) {
    // Cascading termination for multilevel child parent relationship
    // If the process is out of commands to run
    if (pcb.programCounter.getCommandCounter() > commands.size()) {
        pcb.setState(STATE.EXIT);
        CPU.pcbList.put(pcb.getProcessPID(), pcb);
        // System.out.println(pcb.getChildPID());
        if (pcbList.get(pcb.getChildPID()) != null) {
            PCB childPCB = pcbList.get(pcb.getChildPID());
            childPCB.programCounter.setCounter(10000);
            childPCB.setState(STATE.EXIT);
            CPU.pcbList.put(childPCB.getProcessPID(), childPCB);
            if (pcbList.get(pcbList.get(pcb.getChildPID()).getChildPID()) != null) {
                PCB grandChildPCB = pcbList.get(childPCB.getChildPID());
                grandChildPCB.programCounter.setCounter(10000);
                grandChildPCB.setState(STATE.EXIT);
                CPU.pcbList.put(grandChildPCB.getProcessPID(), grandChildPCB);

                OS.info.setText(OS.info.getText() + "\n" + process.getProcessName() + ", its child "
                        + childPCB.getProcess().getProcessName() + ", and its grandchildren "
                        + grandChildPCB.getProcess().getProcessName() + " has been terminated");

                Process.memoryCount -= process.memory - childPCB.getProcess().memory
                        - grandChildPCB.getProcess().memory;
            } else {

                OS.info.setText(OS.info.getText() + "\n" + process.getProcessName() + " and its child "
                        + childPCB.getProcess().getProcessName() + "has been terminated");

                Process.memoryCount -= process.memory - childPCB.getProcess().memory;
            }
        } else {
            Process.memoryCount -= process.memory;

            OS.info.setText(OS.info.getText() + "\n" + process.getProcessName() + " been terminated");

        }
```

- **Multilevel parent relationship** is added to for by allowing the child process create a child as well.
- The cascading termination still holds based on the new conditions check for a child process, if a parent has a grandchild, the childs child, it terminate with the parent and grandparent

Requirements:

- Atleast java 8