DOCUMENTATIE

TEMA *2*

NUME STUDENT: Ivan Alexandru – Mihai
GRUPA: 30222

CUPRINS

1. Obiectivul temei

Obiectivul principal

Crearea si implementarea unei aplicatii de gestionare a cozilor care atribuie clinetii in cozi, astfel incat timpul de asteptare este minimizat.

Obiective secudare

- Analizarea problemei si identificarea necesitatilor
- Crearea unui design pentru aplicatia de simulare
- Implementarea aplicatiei de simulare
- Testarea simularilor aplicatiei

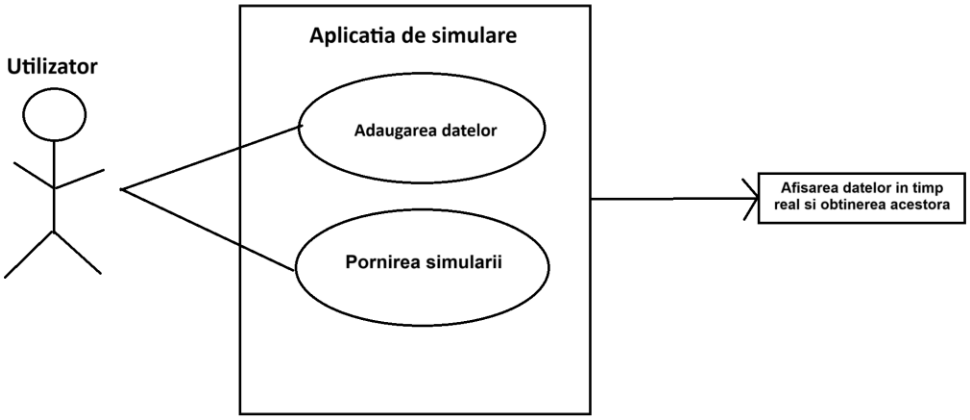
2. Analiza problemei, modelare, scenarii, cazuri de utilizare

Cerintele functionale ale temei:

- Crearea unei interfete grafice prin care utilizatorul poate interactiona cu aplicatia de simulare
- Posibilitatea introducerii datelor de catre utilizator
- Posibilitatea pornirii simularii
- Posibilitatea vizualizarii in timp real a progresului in cozi

Cerintele non-functionale ale temei:

- Aplicatia trebuie sa fie intuitiva si usor de folosit
- Performanta trebuie sa fie una ridicata, simularea oferind date exacte
- Sincronizarea metodelor



Cazuri de utilizare

- Odata cu rularea aplicatiei, utilizatorului i se va deschide o interfata aplicatiei de simulare
- Utilizatorul va putea introduce date precum: numarul de clienti, numarul de cozi, timpul maxim al simularii, timpul minim si maxim pentru sosire, timpul minim si maxim pentru procesarea comenzii si strategia
- Dupa ce s-au introdus date valide (numai numere intregi) se va afisa butonul start
- Se va deschide o noua interfata unde se va afisa continutul cozilor in timp real, precum si lista de asteptare
- Intr-un fisier txt se va salva progresul simularii si timpul mediu de asteptare

3. Proiectare

Diagrama pachete

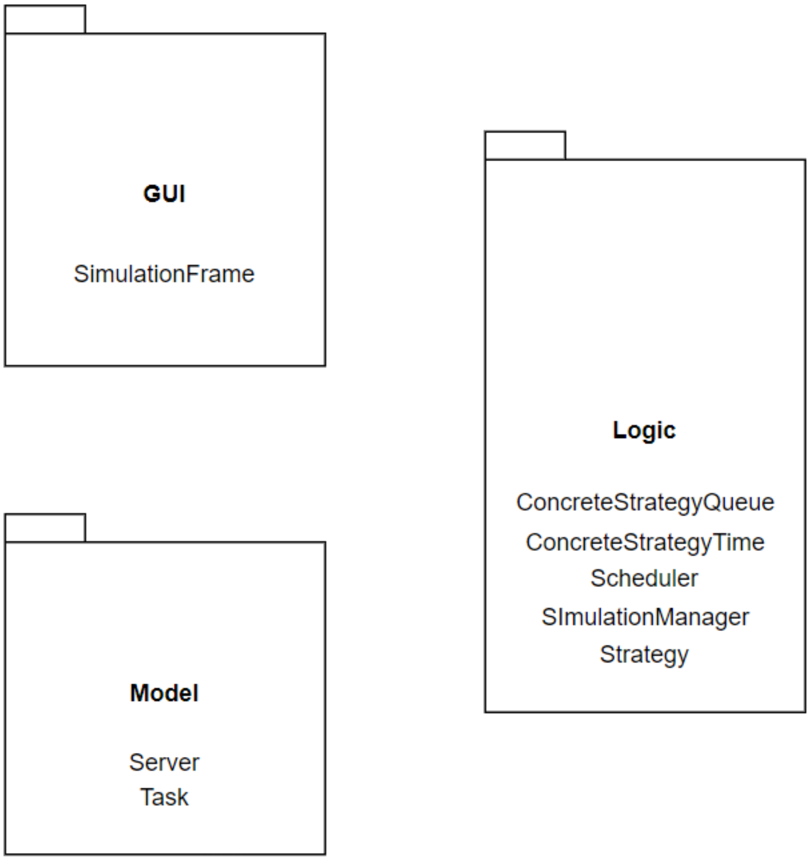
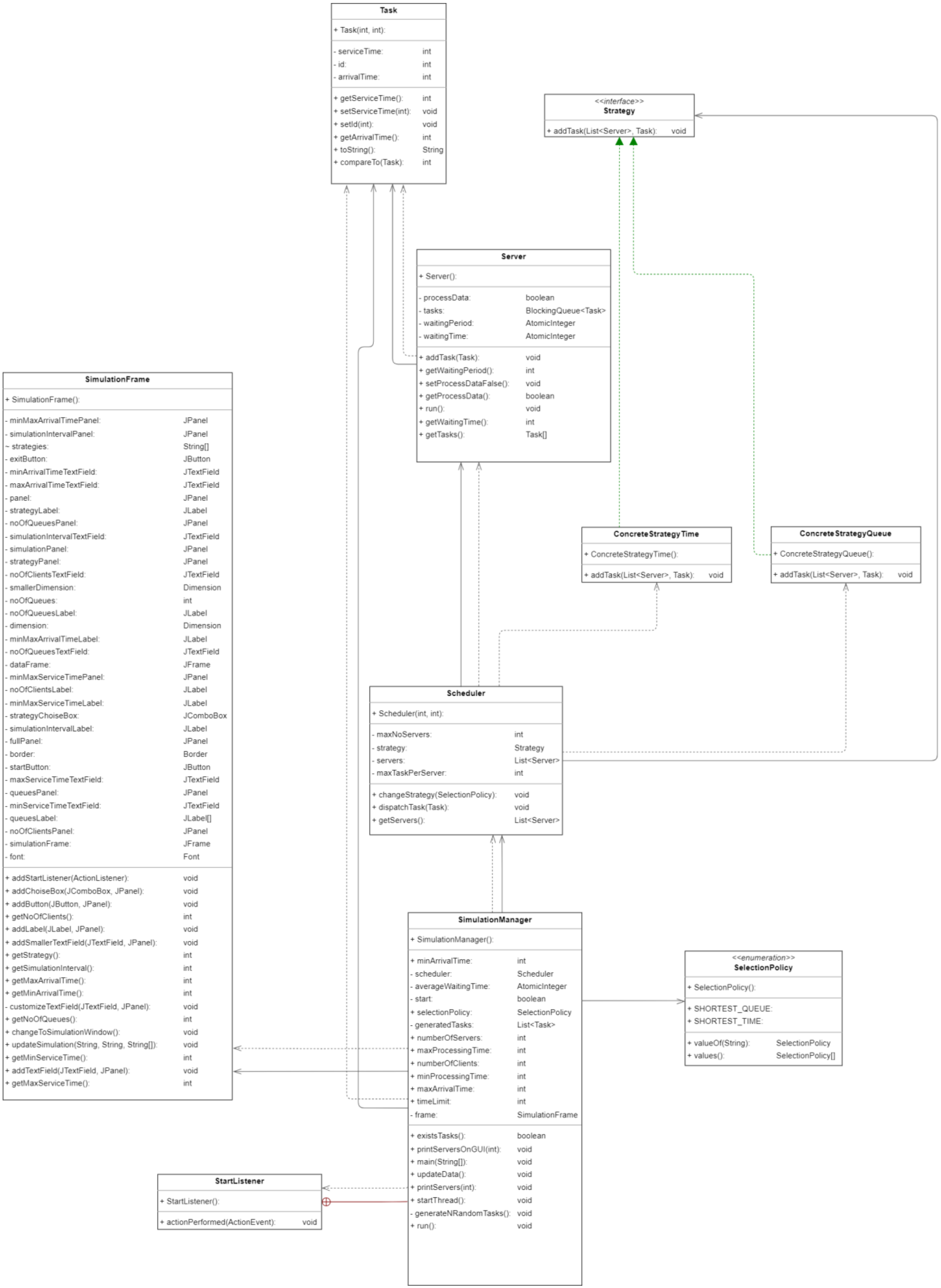


Diagrama UML


Task
+ Task(int, int):
- serviceTime: int
- id: int
- arrivalTime: int
+ getServiceTime(): int
+ setServiceTime(int): void
+ setId(int): void
+ getArrivalTime(): int
+ toString(): String
+ compareTo(Task): int
<<interface>>
Strategy
+ addTask(List<Server>, Task): void
Server
+ Server():
- processData: boolean
- tasks: BlockingQueue<Task>
- waitingPeriod: AtomicInteger
- waitingTime: AtomicInteger
+ addTask(Task): void
+ getWaitingPeriod(): int
+ setProcessDataFalse(): void
+ getProcessData(): boolean
+ run(): void
+ getWaitingTime(): int
+ getTasks(): Task[]
ConcreteStrategyTime
+ ConcreteStrategyTime():
+ addTask(List<Server>, Task): void
ConcreteStrategyQueue
+ ConcreteStrategyQueue():
+ addTask(List<Server>, Task): void
SimulationFrame
+ SimulationFrame():
- minMaxArrivalTimePanel: JPanel
- simulationIntervalPanel: JPanel
~ strategies: String[]
- exitButton: JButton
- minArrivalTimeTextField: JTextField
- maxArrivalTimeTextField: JTextField
- panel: JPanel
- strategyLabel: JLabel
- noOfQueuesPanel: JPanel
- simulationIntervalTextField: JTextField
- simulationPanel: JPanel
- strategyPanel: JPanel
- noOfClientsTextField: JTextField
- smallerDimension: Dimension
- noOfQueues: int
- noOfQueuesLabel: JLabel
- dimension: Dimension
- minMaxArrivalTimeLabel: JLabel
- noOfQueuesTextField: JTextField
- dataFrame: JFrame
- minMaxServiceTimePanel: JPanel
- noOfClientsLabel: JLabel
- minMaxServiceTimeLabel: JLabel
- strategyChoiseBox: JComboBox
- simulationIntervalLabel: JLabel
- fullPanel: JPanel
- border: Border
- startButton: JButton
- maxServiceTimeTextField: JTextField
- queuesPanel: JPanel
- minServiceTimeTextField: JTextField
- queuesLabel: JLabel[]
- noOfClientsPanel: JPanel
- simulationFrame: JFrame
- font: Font
+ addStartListener(ActionListener): void
+ addChoiseBox(JComboBox, JPanel): void
+ addButton(JButton, JPanel): void
+ getNoOfClients(): int
+ addLabel(JLabel, JPanel): void
+ addSmallerTextField(JTextField, JPanel): void
+ getStrategy(): int
+ getSimulationInterval(): int
+ getMaxArrivalTime(): int
+ getMinArrivalTime(): int
- customizeTextField(JTextField, JPanel): void
+ getNoOfQueues(): int
+ changeToSimulationWindow(): void
+ updateSimulation(String, String, String[]): void
+ getMinServiceTime(): int
+ addTextField(JTextField, JPanel): void
+ getMaxServiceTime(): int
Scheduler
+ Scheduler(int, int):
- maxNoServers: int
- strategy: Strategy
- servers: List<Server>
- maxTaskPerServer: int
+ changeStrategy(SelectionPolicy): void
+ dispatchTask(Task): void
+ getServers(): List<Server>
SimulationManager
+ SimulationManager():
+ minArrivalTime: int
- scheduler: Scheduler
- averageWaitingTime: AtomicInteger
- start: boolean
+ selectionPolicy: SelectionPolicy
- generatedTasks: List<Task>
+ numberOfServers: int
+ maxProcessingTime: int
+ numberOfClients: int
+ minProcessingTime: int
+ maxArrivalTime: int
+ timeLimit: int
- frame: SimulationFrame
+ existsTasks(): boolean
+ printServersOnGUI(int): void
+ main(String[]): void
+ updateData(): void
+ printServers(int): void
+ startThread(): void
- generateNRandomTasks(): void
+ run(): void
<<enumeration>>
SelectionPolicy
+ SelectionPolicy():
+ SHORTEST_QUEUE:
+ SHORTEST_TIME:
+ valueOf(String): SelectionPolicy
+ values(): SelectionPolicy[]
StartListener
+ StartListener():
+ actionPerformed(ActionEvent): void

4. Implementare

1. Clasa SimulationFrame

In aceasta clasa se afla codul pentru interfata grafica

- Metoda changeToSimulationFrame face trecerea de la interfata cu date de intrare catre interfata de simulare
- Metoda updateSimulation updateaza datele in timp real

2. Clasa ConcreteStrategyQueue

In aceasta clasa se adauga task-ul in cel mai scurt server ca lungime de clienti

3. Clasa ConcreteStrategyTime

In aceasta clasa se adauga task-ul in cel mai scurt server care are service time-ul cel mai mic

4. Clasa Scheduler

In aceasta clasa se creaza numarul de servere cerute si se pornesc

- Metoda ChangeStrategy selecteaza tipul de strategie pe care il folosim pentru adaugarea task-urilor in servere
- Metoda dispatch adauga task-ul in server
- Metoda getServers obtine serverele ca o lista

5. Clasa SimulationManager

In aceasta clasa se creaza o intefata care asteapta introducerea datelor, iar dupa ce se apasa butonul start se incepe simularea

- Metoda generateNRandomTask se genereaza un numar de taskuri random
- Metoda existsTask verifica daca mai exista taskuri in waiting list sau in servere
- Metoda updateData updateaza in timp real datele pe interfata de simulare
- Metoda run asteapta ca butonul start sa fie apasat, apoi verifica daca un task are arrival time-ul egal cu current time-ul si il adauga intr-un server pentru a fi procesat, asta se va intampla pana nu mai exista taskuri in waiting list sau s-a depasit timpul de simulare
- Metoda printServers afiseaza continutul fiecarui server atat in fisier txt cat si pe interfata de simulare
- Metoda startThread verifica daca s-a apasat butonul start
- Subclasa StartListener creeaza un actionListener pentru butonul start

6. <u>Interfata Strategy</u>

Este o interfata pentru clasele ConcreteStrategyQueue si ConcreteStrategyTime

7. <u>Clasa Server</u>

- Metoda addTask adauga task-uri in server
- Metoda run proceseaza task-ul odata la o secunda

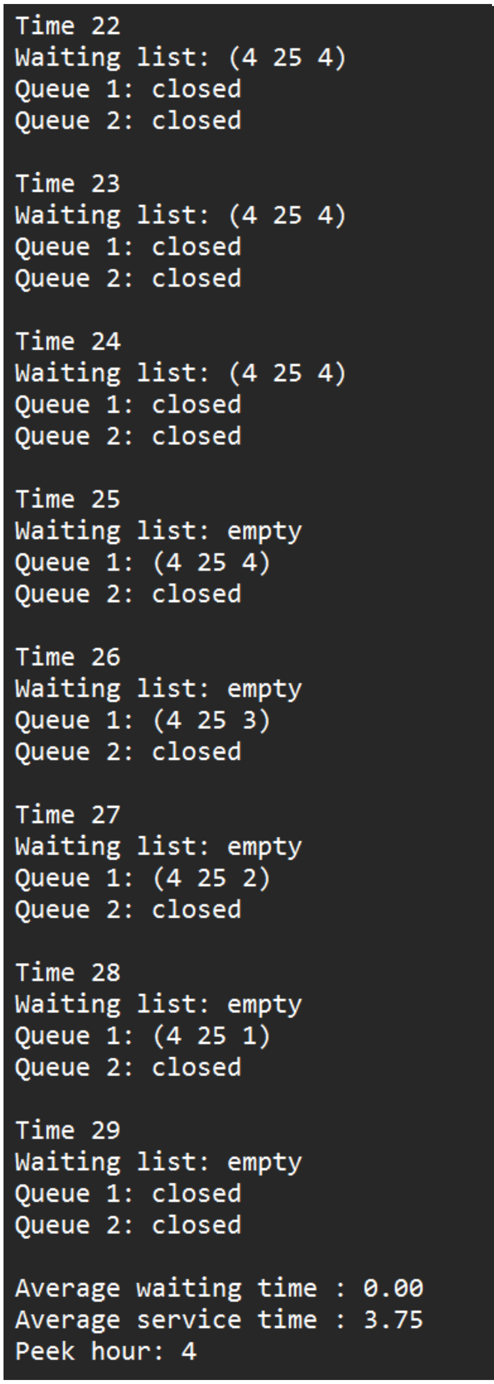
8. <u>Clasa Task</u>

In aceasta clasa se definesc atributele pe care un task trebuie sa le aiba

5. Rezultate

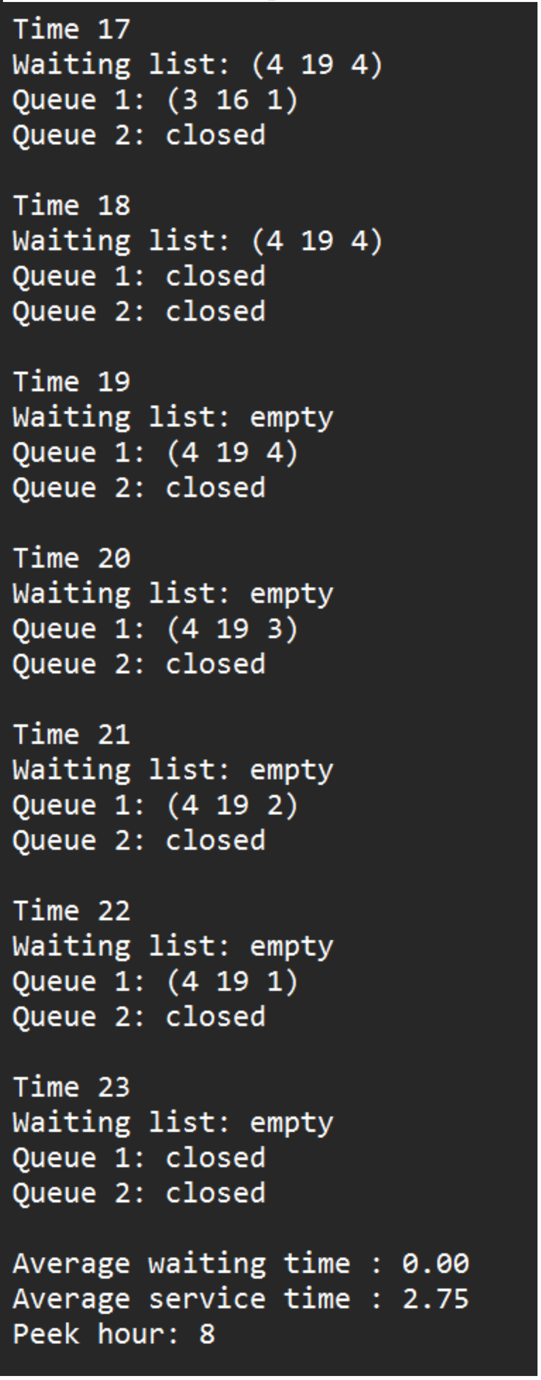
Test 1
N = 4
Q = 2
tsimulation MAX = 60 seconds
[tarrival MIN, tarrival MAX] = [2, 30]
[tservice MIN, tservice MAX] = [2, 4]

Shortest time



```
Time 22
Waiting list: (4 25 4)
Queue 1: closed
Queue 2: closed

Time 23
Waiting list: (4 25 4)
Queue 1: closed
Queue 2: closed

Time 24
Waiting list: (4 25 4)
Queue 1: closed
Queue 2: closed

Time 25
Waiting list: empty
Queue 1: (4 25 4)
Queue 2: closed

Time 26
Waiting list: empty
Queue 1: (4 25 3)
Queue 2: closed

Time 27
Waiting list: empty
Queue 1: (4 25 2)
Queue 2: closed

Time 28
Waiting list: empty
Queue 1: (4 25 1)
Queue 2: closed

Time 29
Waiting list: empty
Queue 1: closed
Queue 2: closed

Average waiting time : 0.00
Average service time : 3.75
Peek hour: 4
```

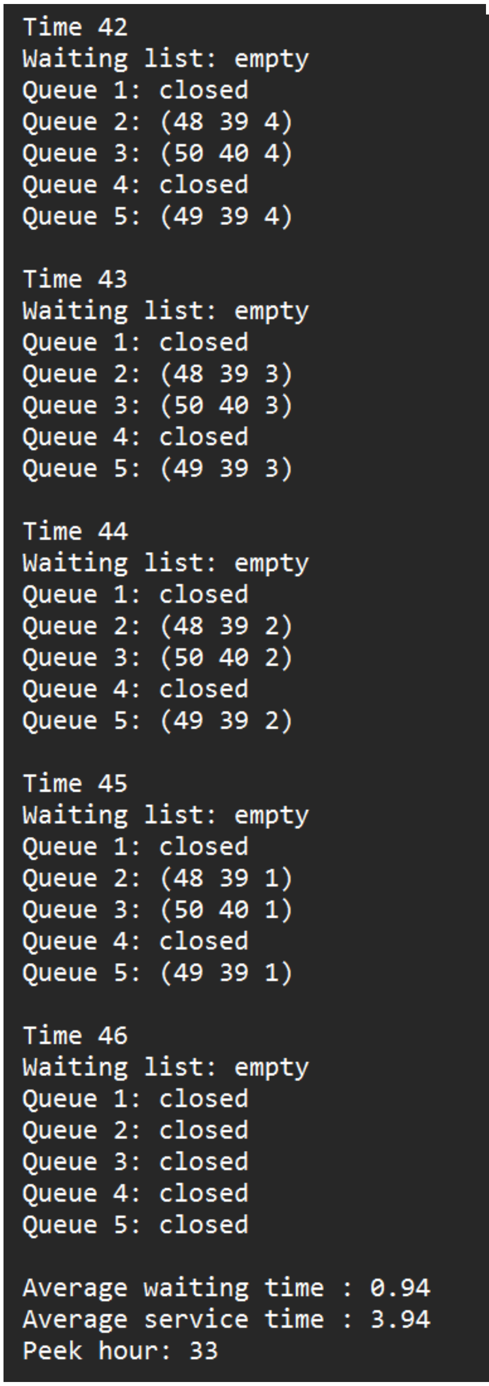
Shortest queue



```
Time 17
Waiting list: (4 19 4)
Queue 1: (3 16 1)
Queue 2: closed

Time 18
Waiting list: (4 19 4)
Queue 1: closed
Queue 2: closed

Time 19
Waiting list: empty
Queue 1: (4 19 4)
Queue 2: closed

Time 20
Waiting list: empty
Queue 1: (4 19 3)
Queue 2: closed

Time 21
Waiting list: empty
Queue 1: (4 19 2)
Queue 2: closed

Time 22
Waiting list: empty
Queue 1: (4 19 1)
Queue 2: closed

Time 23
Waiting list: empty
Queue 1: closed
Queue 2: closed

Average waiting time : 0.00
Average service time : 2.75
Peek hour: 8
```

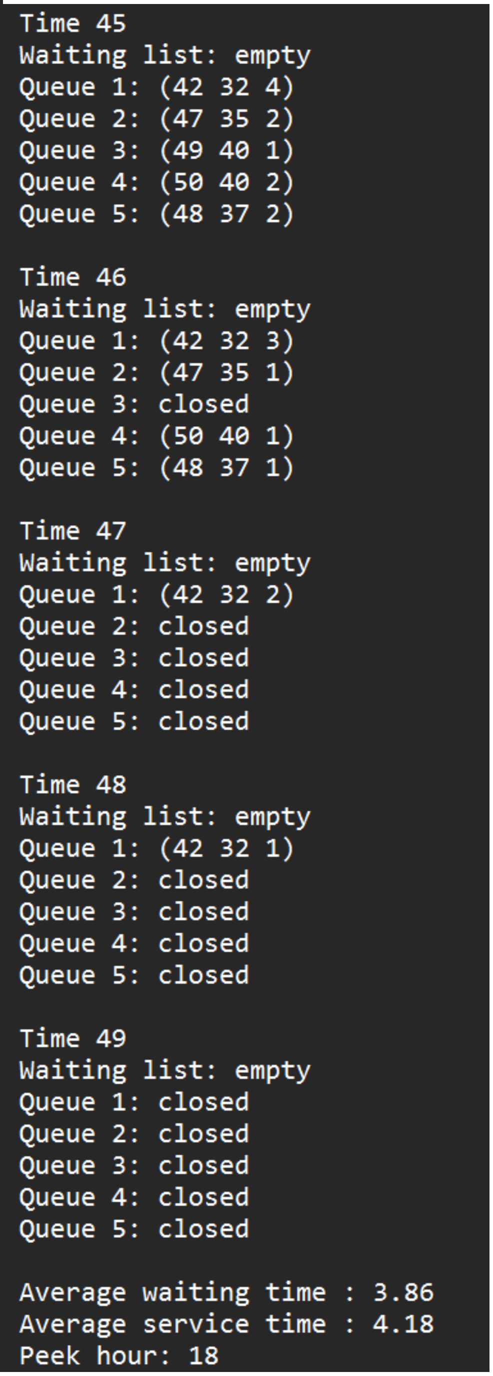
Test 2
N = 50
Q = 5
tsimulation MAX = 60 seconds
[tarrival MIN, tarrival MAX] = [2, 40]
[tservice MIN, tservice MAX]= [1, 7]

Shortest time



```
Time 42
Waiting list: empty
Queue 1: closed
Queue 2: (48 39 4)
Queue 3: (50 40 4)
Queue 4: closed
Queue 5: (49 39 4)

Time 43
Waiting list: empty
Queue 1: closed
Queue 2: (48 39 3)
Queue 3: (50 40 3)
Queue 4: closed
Queue 5: (49 39 3)

Time 44
Waiting list: empty
Queue 1: closed
Queue 2: (48 39 2)
Queue 3: (50 40 2)
Queue 4: closed
Queue 5: (49 39 2)

Time 45
Waiting list: empty
Queue 1: closed
Queue 2: (48 39 1)
Queue 3: (50 40 1)
Queue 4: closed
Queue 5: (49 39 1)

Time 46
Waiting list: empty
Queue 1: closed
Queue 2: closed
Queue 3: closed
Queue 4: closed
Queue 5: closed

Average waiting time : 0.94
Average service time : 3.94
Peek hour: 33
```

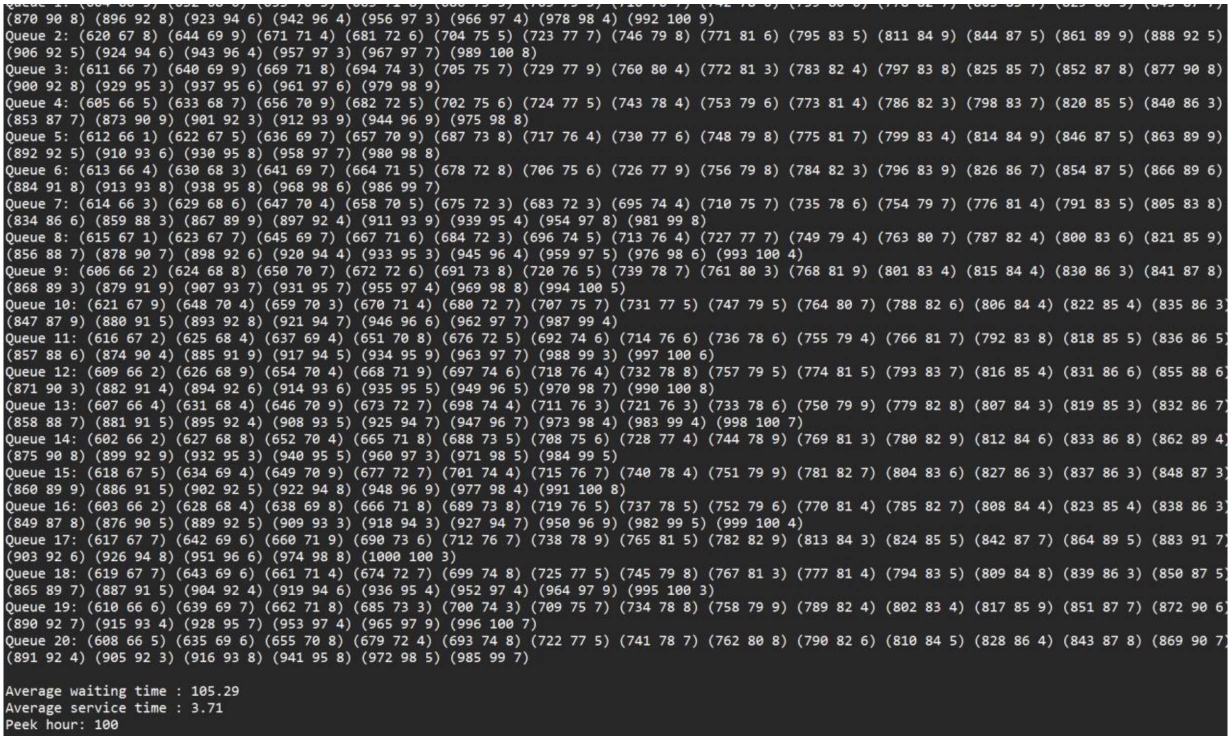
Shortest queue



```
Time 45
Waiting list: empty
Queue 1: (42 32 4)
Queue 2: (47 35 2)
Queue 3: (49 40 1)
Queue 4: (50 40 2)
Queue 5: (48 37 2)

Time 46
Waiting list: empty
Queue 1: (42 32 3)
Queue 2: (47 35 1)
Queue 3: closed
Queue 4: (50 40 1)
Queue 5: (48 37 1)

Time 47
Waiting list: empty
Queue 1: (42 32 2)
Queue 2: closed
Queue 3: closed
Queue 4: closed
Queue 5: closed

Time 48
Waiting list: empty
Queue 1: (42 32 1)
Queue 2: closed
Queue 3: closed
Queue 4: closed
Queue 5: closed

Time 49
Waiting list: empty
Queue 1: closed
Queue 2: closed
Queue 3: closed
Queue 4: closed
Queue 5: closed

Average waiting time : 3.86
Average service time : 4.18
Peek hour: 18
```

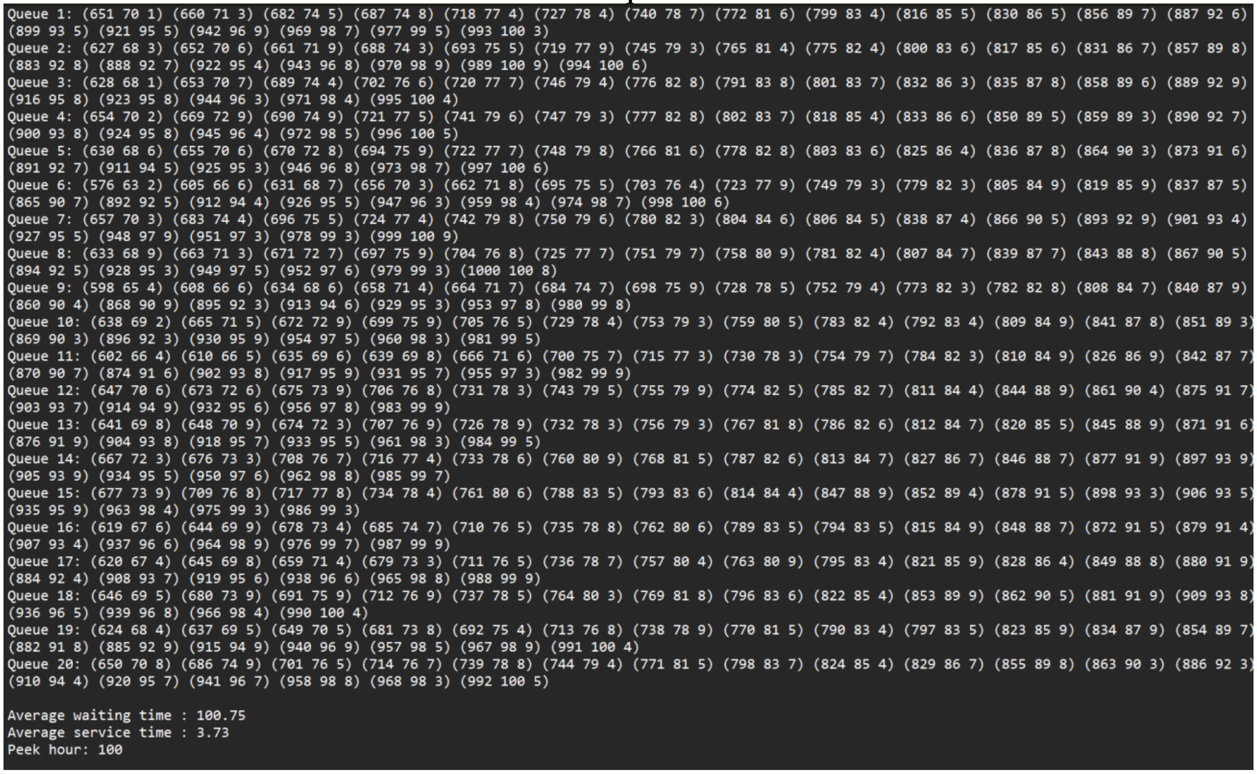
Test 3
N = 1000
Q = 20
tsimulation MAX = 200 seconds
[tarrival MIN, tarrival MAX] = [10, 100]
[tservice MIN, tservice MAX]=]= [3, 9]

Shortest time



```
(870 90 8) (896 92 8) (923 94 6) (942 96 4) (956 97 3) (966 97 4) (978 98 4) (992 100 9)
Queue 2: (620 67 8) (644 69 9) (671 71 4) (681 72 6) (704 75 5) (723 77 7) (746 79 8) (771 81 6) (795 83 5) (811 84 9) (844 87 5) (861 89 9) (888 92 5)
(906 92 5) (924 94 6) (943 96 4) (957 97 3) (967 97 7) (989 100 8)
Queue 3: (611 66 7) (640 69 9) (669 71 8) (694 74 3) (705 75 7) (729 77 9) (760 80 4) (772 81 3) (783 82 4) (797 83 8) (825 85 7) (852 87 8) (877 90 8)
(900 92 8) (929 95 3) (937 95 6) (961 97 6) (979 98 9)
Queue 4: (605 66 5) (633 68 7) (656 70 9) (682 72 5) (702 75 6) (724 77 5) (743 78 4) (753 79 6) (773 81 4) (786 82 3) (798 83 7) (820 85 5) (840 86 3)
(853 87 7) (873 90 9) (901 92 3) (912 93 9) (944 96 9) (975 98 8)
Queue 5: (612 66 1) (622 67 5) (636 69 7) (657 70 9) (687 73 8) (717 76 4) (730 77 6) (748 79 8) (775 81 7) (799 83 4) (814 84 9) (846 87 5) (863 89 9)
(892 92 5) (910 93 6) (930 95 8) (958 97 7) (980 98 8)
Queue 6: (613 66 4) (630 68 3) (641 69 7) (664 71 5) (678 72 8) (706 75 6) (726 77 9) (756 79 8) (784 82 3) (796 83 9) (826 86 7) (854 87 5) (866 89 6)
(884 91 8) (913 93 8) (938 95 8) (968 98 6) (986 99 7)
Queue 7: (614 66 3) (629 68 6) (647 70 4) (658 70 5) (675 72 3) (683 72 3) (695 74 4) (710 75 7) (735 78 6) (754 79 7) (776 81 4) (791 83 5) (805 83 8)
(834 86 6) (859 88 3) (867 89 9) (897 92 4) (911 93 9) (939 95 4) (954 97 8) (981 99 8)
Queue 8: (615 67 1) (623 67 7) (645 69 7) (667 71 6) (684 72 3) (696 74 5) (713 76 4) (727 77 7) (749 79 4) (763 80 7) (787 82 4) (800 83 6) (821 85 9)
(856 88 7) (878 90 7) (898 92 6) (920 94 4) (933 95 3) (945 96 4) (959 97 5) (976 98 6) (993 100 4)
Queue 9: (606 66 2) (624 68 8) (650 70 7) (672 72 6) (691 73 8) (720 76 5) (739 78 7) (761 80 3) (768 81 9) (801 83 4) (815 84 4) (830 86 3) (841 87 8)
(868 89 3) (879 91 9) (907 93 7) (931 95 7) (955 97 4) (969 98 8) (994 100 5)
Queue 10: (621 67 9) (648 70 4) (659 70 3) (670 71 4) (680 72 7) (707 75 7) (731 77 5) (747 79 5) (764 80 7) (788 82 6) (806 84 4) (822 85 4) (835 86 3
(847 87 9) (880 91 5) (893 92 8) (921 94 7) (946 96 6) (962 97 7) (987 99 4)
Queue 11: (616 67 2) (625 68 4) (637 69 4) (651 70 8) (676 72 5) (692 74 6) (714 76 6) (736 78 6) (755 79 4) (766 81 7) (792 83 8) (818 85 5) (836 86 5
(857 88 6) (874 90 4) (885 91 9) (917 94 5) (934 95 9) (963 97 7) (988 99 3) (997 100 6)
Queue 12: (609 66 2) (626 68 9) (654 70 4) (668 71 9) (697 74 6) (718 76 4) (732 78 8) (757 79 5) (774 81 5) (793 83 7) (816 85 4) (831 86 6) (855 88 6
(871 90 3) (882 91 4) (894 92 6) (914 93 6) (935 95 5) (949 96 5) (970 98 7) (990 100 8)
Queue 13: (607 66 4) (631 68 4) (646 70 9) (673 72 7) (698 74 4) (711 76 3) (721 76 3) (733 78 6) (750 79 9) (779 82 8) (807 84 3) (819 85 3) (832 86 7
(858 88 7) (881 91 5) (895 92 4) (908 93 5) (925 94 7) (947 96 7) (973 98 4) (983 99 4) (998 100 7)
Queue 14: (602 66 2) (627 68 8) (652 70 4) (665 71 8) (688 73 5) (708 75 6) (728 77 4) (744 78 9) (769 81 3) (780 82 9) (812 84 6) (833 86 8) (862 89 4
(875 90 8) (899 92 9) (932 95 3) (940 95 5) (960 97 3) (971 98 5) (984 99 5)
Queue 15: (618 67 5) (634 69 4) (649 70 9) (677 72 7) (701 74 4) (715 76 7) (740 78 4) (751 79 9) (781 82 7) (804 83 6) (827 86 3) (837 86 3) (848 87 3
(860 89 9) (886 91 5) (902 92 5) (922 94 8) (948 96 9) (977 98 4) (991 100 8)
Queue 16: (603 66 2) (628 68 4) (638 69 8) (666 71 8) (689 73 8) (719 76 5) (737 78 5) (752 79 6) (770 81 4) (785 82 7) (808 84 4) (823 85 4) (838 86 3
(849 87 8) (876 90 5) (889 92 5) (909 93 3) (918 94 3) (927 94 7) (950 96 9) (982 99 5) (999 100 4)
Queue 17: (617 67 7) (642 69 6) (660 71 9) (690 73 6) (712 76 7) (738 78 9) (765 81 5) (782 82 9) (813 84 3) (824 85 5) (842 87 7) (864 89 5) (883 91 7
(903 92 6) (926 94 8) (951 96 6) (974 98 8) (1000 100 3)
Queue 18: (619 67 7) (643 69 6) (661 71 4) (674 72 7) (699 74 8) (725 77 5) (745 79 8) (767 81 3) (777 81 4) (794 83 5) (809 84 8) (839 86 3) (850 87 5
(865 89 7) (887 91 5) (904 92 4) (919 94 6) (936 95 4) (952 97 4) (964 97 9) (995 100 3)
Queue 19: (610 66 6) (639 69 7) (662 71 8) (685 73 3) (700 74 3) (709 75 7) (734 78 8) (758 79 9) (789 82 4) (802 83 4) (817 85 9) (851 87 7) (872 90 6
(890 92 7) (915 93 4) (928 95 7) (953 97 4) (965 97 9) (996 100 7)
Queue 20: (608 66 5) (635 69 6) (655 70 8) (679 72 4) (693 74 8) (722 77 5) (741 78 7) (762 80 8) (790 82 6) (810 84 5) (828 86 4) (843 87 8) (869 90 7
(891 92 4) (905 92 3) (916 93 8) (941 95 8) (972 98 5) (985 99 7)

Average waiting time : 105.29
Average service time : 3.71
Peek hour: 100
```

Shortest queue



```
Queue 1: (651 70 1) (660 71 3) (682 74 5) (687 74 8) (718 77 4) (727 78 4) (740 78 7) (772 81 6) (799 83 4) (816 85 5) (830 86 5) (856 89 7) (887 92 6)
(899 93 5) (921 95 5) (942 96 9) (969 98 7) (977 99 5) (993 100 3)
Queue 2: (627 68 3) (652 70 6) (661 71 9) (688 74 3) (693 75 5) (719 77 9) (745 79 3) (765 81 4) (775 82 4) (800 83 6) (817 85 6) (831 86 7) (857 89 8)
(883 92 8) (888 92 7) (922 95 4) (943 96 8) (970 98 9) (989 100 9) (994 100 6)
Queue 3: (628 68 1) (653 70 7) (689 74 4) (702 76 6) (720 77 7) (746 79 4) (776 82 8) (791 83 8) (801 83 7) (832 86 3) (835 87 8) (858 89 6) (889 92 9)
(916 95 8) (923 95 8) (944 96 3) (971 98 4) (995 100 4)
Queue 4: (654 70 2) (669 72 9) (690 74 9) (721 77 5) (741 79 6) (747 79 3) (777 82 8) (802 83 7) (818 85 4) (833 86 6) (850 89 5) (859 89 3) (890 92 7)
(900 93 8) (924 95 8) (945 96 4) (972 98 5) (996 100 5)
Queue 5: (630 68 6) (655 70 6) (670 72 8) (694 75 9) (722 77 7) (748 79 8) (766 81 6) (778 82 8) (803 83 6) (825 86 4) (836 87 8) (864 90 3) (873 91 6)
(891 92 7) (911 94 5) (925 95 3) (946 96 8) (973 98 7) (997 100 6)
Queue 6: (576 63 2) (605 66 6) (631 68 7) (656 70 3) (662 71 8) (695 75 5) (703 76 4) (723 77 9) (749 79 3) (779 82 3) (805 84 9) (819 85 9) (837 87 5)
(865 90 7) (892 92 5) (912 94 4) (926 95 5) (947 96 3) (959 98 4) (974 98 7) (998 100 6)
Queue 7: (657 70 3) (683 74 4) (696 75 5) (724 77 4) (742 79 8) (750 79 6) (780 82 3) (804 84 6) (806 84 5) (838 87 4) (866 90 5) (893 92 9) (901 93 4)
(927 95 5) (948 97 9) (951 97 3) (978 99 3) (999 100 9)
Queue 8: (633 68 9) (663 71 3) (671 72 7) (697 75 9) (704 76 8) (725 77 7) (751 79 7) (758 80 9) (781 82 4) (807 84 7) (839 87 7) (843 88 8) (867 90 5)
(894 92 5) (928 95 3) (949 97 5) (952 97 6) (979 99 3) (1000 100 8)
Queue 9: (598 65 4) (608 66 6) (634 68 6) (658 71 4) (664 71 7) (684 74 7) (698 75 9) (728 78 5) (752 79 4) (773 82 3) (782 82 8) (808 84 7) (840 87 9)
(860 90 4) (868 90 9) (895 92 3) (913 94 6) (929 95 3) (953 97 8) (980 99 8)
Queue 10: (638 69 2) (665 71 5) (672 72 9) (699 75 9) (705 76 5) (729 78 4) (753 79 3) (759 80 5) (783 82 4) (792 83 4) (809 84 9) (841 87 8) (851 89 3)
(869 90 3) (896 92 3) (930 95 9) (954 97 5) (960 98 3) (981 99 5)
Queue 11: (602 66 4) (610 66 5) (635 69 6) (639 69 8) (666 71 6) (700 75 7) (715 77 3) (730 78 3) (754 79 7) (784 82 3) (810 84 9) (826 86 9) (842 87 7)
(870 90 7) (874 91 6) (902 93 8) (917 95 9) (931 95 7) (955 97 3) (982 99 9)
Queue 12: (647 70 6) (673 72 6) (675 73 9) (706 76 8) (731 78 3) (743 79 5) (755 79 9) (774 82 5) (785 82 7) (811 84 4) (844 88 9) (861 90 4) (875 91 7)
(903 93 7) (914 94 9) (932 95 6) (956 97 8) (983 99 9)
Queue 13: (641 69 8) (648 70 9) (674 72 3) (707 76 9) (726 78 9) (732 78 3) (756 79 3) (767 81 8) (786 82 6) (812 84 7) (820 85 5) (845 88 9) (871 91 6)
(876 91 9) (904 93 8) (918 95 7) (933 95 5) (961 98 3) (984 99 5)
Queue 14: (667 72 3) (676 73 3) (708 76 7) (716 77 4) (733 78 6) (760 80 9) (768 81 5) (787 82 6) (813 84 7) (827 86 7) (846 88 7) (877 91 9) (897 93 9)
(905 93 9) (934 95 5) (950 97 6) (962 98 8) (985 99 7)
Queue 15: (677 73 9) (709 76 8) (717 77 8) (734 78 4) (761 80 6) (788 83 5) (793 83 6) (814 84 4) (847 88 9) (852 89 4) (878 91 5) (898 93 3) (906 93 5)
(935 95 9) (963 98 4) (975 99 3) (986 99 3)
Queue 16: (619 67 6) (644 69 9) (678 73 4) (685 74 7) (710 76 5) (735 78 8) (762 80 6) (789 83 5) (794 83 5) (815 84 9) (848 88 7) (872 91 5) (879 91 4)
(907 93 4) (937 96 6) (964 98 9) (976 99 7) (987 99 9)
Queue 17: (620 67 4) (645 69 8) (659 71 4) (679 73 3) (711 76 5) (736 78 7) (757 80 4) (763 80 9) (795 83 4) (821 85 9) (828 86 4) (849 88 8) (880 91 9)
(884 92 4) (908 93 7) (919 95 6) (938 96 6) (965 98 8) (988 99 9)
Queue 18: (646 69 5) (680 73 9) (691 75 9) (712 76 9) (737 78 5) (764 80 3) (769 81 8) (796 83 6) (822 85 4) (853 89 9) (862 90 5) (881 91 9) (909 93 8)
(936 96 5) (939 96 8) (966 98 4) (990 100 4)
Queue 19: (624 68 4) (637 69 5) (649 70 5) (681 73 8) (692 75 4) (713 76 8) (738 78 9) (770 81 5) (790 83 4) (797 83 5) (823 85 9) (834 87 9) (854 89 7)
(882 91 8) (885 92 9) (915 94 9) (940 96 9) (957 98 5) (967 98 9) (991 100 4)
Queue 20: (650 70 8) (686 74 9) (701 76 5) (714 76 7) (739 78 8) (744 79 4) (771 81 5) (798 83 7) (824 85 4) (829 86 7) (855 89 8) (863 90 3) (886 92 3)
(910 94 4) (920 95 7) (941 96 7) (958 98 8) (968 98 3) (992 100 5)

Average waiting time : 100.75
Average service time : 3.73
Peek hour: 100
```

6. Concluzii

In aceasta tema am dobandit urmatoarele skilluri:
- Intelegerea threadurilor
- Intelegerea atributelor ce tin de threaduri
- Lucrarea cu threaduri

Posibile dezvoltari:
- Butonul exit sa fie inlocuit cu un buton back si sa se poata crea alta simulare

7. Bibliografie

1. https://dsrl.eu/courses/pt/materials/PT_2024_A2_S1.pdf
2. https://dsrl.eu/courses/pt/materials/PT_2024_A2_S2.pdf
3. https://www.geeksforgeeks.org/introduction-of-deadlock-in-operating-system/
4. https://docs.oracle.com/javase/tutorial/essential/concurrency/index.html
5. https://www.tutorialspoint.com/java/util/timer_schedule_period.htm
6. https://www.javacodegeeks.com/2013/01/java-thread-pool-example-using-executors-and-threadpoolexecutor.html