

## מטלה 3

### מחשבון ריצת תהליכיים

הקדם:

בגלל שניתן לבוחר בכל שפה,  
ונוחות עם שפת JAVA, העדפנו לכתוב את מטלה זאת בשפת תכנות הנ"ל. הריצה ת策ר להיות עם  
פקודה טיפה שונה (מופיע גם ב README )  
`java -jar .\TurnAroundCalculator.jar`  
<נתיב מלא לקובץ קלט>

הנחות יסוד:

- למורת שכותב "ניתן להניח שהקלטים נכונים (מספר התהליכיים =1), זמני ההגעה והעיבוד הינם=0". אנו בודקים את התקינות של התוכן.
- למען נוחות כתיבה, במקום "יחידות זמן" נשימוש במילה "שניה"\\"שניות" אבל הן חלייפות למל דבר.
- בקריאה לכל מתודה, אנו מניחים שימוש ArrayList שהוא ממון בסדר עולה לפי זמן הגעה ואינדקס (שורה בקובץ קלט).
- כדי להכיר בין תהליכיים שהגיעו באותו זמן, האינדקס הקטן ביותר הוא המנצח מבין השתיים (אינדקס= שורה בקובץ קלט).
- השורה הראשונה היא כמות התהליכיים בקובץ.
- השורה השנייה והלאה מייצגת תחילר- זמן תחיליה וזמן ריצה מבוקש מופרדים בפסיק ייחיד (לדוגמה "1,4" = מגיע ב 1 ורץ 4 שניות).
- אין צורך באמת "לחכota" ולספור כל שניה אם המעבד בIDLE, מדלגים (או עושים skip" time") לתחılır הבא עם פעולה max עם התחילר הקרוב ביותר. (מבצע את אותה פעולה רק מדמה שהEA קרה).
- כשאנו אומרים "מreira את התחילר" מדובר בלסכם את כמות השניות שהתחılır רץ ברצף עד שהוא מסתיימונעצר.
- Time quantum של 2 = RR
- בחוק Round Robin אנחנו "מראים קדימה" את הזמן אם רק תחילר אחד יכול לרוץ , בין אם הוא היחיד שפועל לבין אם התחילר הבא שפועל מאד רחוק מבחינת זמן התחילה, זה בשבייל לחסוך ריצות בלולאה על תהליכיים שלא צריכים את הזמן quantum וליעיל את זמני הריצה של התוכנית.

- בהגדרת המטלה אנו מתבקשים להכין (shrotest job first) עם PREEMPTIVE, אבל התוצאה בקובץ הבדיקה (ובשיטה של סטודנט עם איגור ובודיקה מול דוד גרמן המתרגל) כוונת המשורר היא (shortest time first REMAINING time first PREEMPTIVE עם שומות הדפסות השארנו את שם הקודם אבל ביצענו SRTF).

## מחלקות עזר:

### **ProcessPCB**

מחלקה שנועדה ליצג תהליך יחיד ואת המידע הקשור אליו

#### 1. תוכנות:

- זמן התחלה
- זמן סיום
- האם הסטיים
- כמה זמן נדרש לריצה (לא משנה את המשתנה זהה, אלא מעדים משתנה E ומחשבים כמה זמן נותר).
- כמה זמן עבר מהזמן המבוקש
- אינדקס (מספר שורה בקובץ קלט)

#### 2. מתודות:

- בנייה רגיל- זורק הודעה אם זמן מבוקש הוא שלילי
- בנייה העתקה
- דרישה למетод EQUALS- משווה לפי אינדקס
- דרישה למетод HASHCODE- מחזירה קוד האש של האינדקס (חייב להיות תואם EQUALS)
- Setters ו Getters
- מחזיר כמה זמן עבר מהגעת התהליך לשינויו getTurnAroundTime
- מחזיר כמה זמן נשאר לריצת התהליך getTimeLeft

מימושה את Comparator, משווה את זמני ההתחלה של שתי תהליכיים, אם שניהם הגיעו יחד, המנצח הוא האינדקס.

מחלקה הראשית:

### **TurnAroundCalculator**

לאויפר אחר PROCESSPCB מתודות עזר שמעתיקה (בעזרת בנייה העתקה) אויף -DEEPCOPY- אוופת את התהליכיים מקובץ הקלט ומריצה את האלגוריתמי תזמון -Main-

- בודקת קלט, אם קיימים קלט והאם הקובץ קיים בכלל
- בעזרת scanner על קובץ, שולפת את כמות התהליכיים בקובץ (בודקת תקין)
- ריצה על הקובץ ולכל שורה
  - חותכת רווחים מיותרים
  - מדלגת אם היא ריקה

- c. יוצרת PROCESSPCB עם התוכן מהשורה (ראו הנחת ייסוד הגדרת שורה) (אם שורה לא עונה על החוק זהה מעלים הودעה תואמת)
- d. מוסיף אותה למערך שלנו
4. מתריעה אם הביאו פחות שורות, מריםים בכל מקרה
- a. שורות 92-99 catch למקורה של שגיאה בקריאות תוכן הקובץ או הקובץ עצמו
- b. ממיינת את המערך של התהליכים לפי ASCOMAPRE (סדר עולה לפי זמן הגעה ואז לפי אינדקס)
5. יוצרת מערך של כל האלגוריתמי ריצה לפי סדר הנדרש
6. מריםה על כל האלגוריתמים:
- a. מעטיקה את מערך התהליכים למען לא לפגוע בריצות הבאות
- b. מריםה את המתודה process run של האלגוריתם
7. יצאת

## הסבר על המתודות:

(מציר את הנחת ייסוד- החסוך collection המתקבל הוא ARRAYLIST של PROCESSPCB מסודר בסדר עולה)

1. **FCFS** - מייצגת את "הראשון ש מגע הראשון שיקבל שירות" -
- a. בודקת תקינות של המערך המתתקבל ומוסיפה בהתאם אם הוא ריק
- b. על כל תהליך בראשימה
- i. בודקת אם יש הפרש בזמןנים, אם כן, מדלגת קידמה בזמן לתהליכי הבא
- ii. מריםה את התהליך עד סופו (סוכמת את כמות השניות עד סיום התהליך)
- iii. סוכמת את הפרש זמן ההתחלה עד זמן הסוף שלו (זמן הURNROUND של התהליך)
- iv. מעכנת את הזמן הגלובלי בזמן הסיום של התהליך
- v. עוברת לבא
- c. מחשבת את ממוצע הURNROUND ומוציאיה הודעה תואמת
- d. יצאת
2. **LCFSNP** - מייצגת את "האחרון שהגיע הוא הראשון שיקבל שירות" (לא preemptive) (non-preemptive)
- נסביר על מתודת עדר זמן putPerTime -
- a. עוברת על המערך המתתקבל
- b. בודקת על כל תהליך - האם הוא הסטטי, האם הוא קיימם כבר במחסנית והאם זמן ההתחלה שלו הגיע כבר.

### -Run process

- a. בודקת תקינות הקולט ומוסיפה הודעה תואמת
- b. יוצרת מחסנית לתוכים מוכנים
- c. עד שהיא עברה על כל התהליכים היא:
- i. משתמש בPUTPERTIME
- ii. בודקת אם המחסנית ריקה- אם כן צריך לדלג בזמן קידמה לתהליכי הקרוב ביותר
- iii. מוציאיה את ההכי חדש (זה שנמצא למעלה במחסנית)- אם הוא הסטטי כבר מדלגת
- iv. מריםה אותו עד סופו וסוכמת אותו
- d. מחשבת את ממוצע הURNROUND ומוציאיה הודעה תואמת
3. **LCFSP** - מייצגת את "האחרון שהגיע הוא הראשון שיקבל שירות" (עם preemptive) (Preemptive)
- a. בודקת תקינות הקולט ומוסיפה הודעה תואמת

- ד. מסדרת את המערך מחדש לכל מקורה
- כ. יוצרת מחסנית לתחליכים מוכנים
- ב. רצה על המערך כל עוד היא לא עברה על כל התחליכים בו (ובודקת אם המחסנית לא ריקה או אם המצביע לתהילר הנוכחי מסמן על תהילר שאולו עוד לא סיימ):
- ג. בודקת אם המעבד עצם, מדלגת בזמן לתהילר הבא
  - ה. מכניסה את כל התחליכים שמוכנים (זמן ההתחילה שלהם לפני או שווה לזמן הכללי שלו) למחסנית ומקדמת את זה - הסמן של התהילר הבא שעדיין לא הגיע (הכנסה זאת מערך מסודר מודאת שהאחרון יהיה הראשון במחסנית להוצאה.
  - ו. אם אין לנו תהילר שרצף כרגע, היא מביאה את העליון במחסנית
  - ז. בודקת את זמן ההתחילה של הבא במערך (אם הוא לא קיים יש מספר זמני של אינסוף)
  - ח. בודקת מה הזמן שנוטר לתהילר הנוכחי
  - ט. בוגע לכמות הזמן שיש בין סוף התהילר הנוכחי וזמן ההגעה של התהילר הבא יש פה שני מקרים:
    1. הנוכחי מסיים לפני הבא, אז מרים אותו עד הסוף ומקדמים את הזמן הכללי בהתאם (מחזבת את זמן הסיום שלו ומסכמת אותו)
    2. הנוכחי נעצר ע"י הבא אך הוא ירוץ עד שהטהילר מגיע, עוזרים אותו ומכניסים אותו למחסנית
- ד. מחזבת את ממוצע TURNAROUND ומוציאיה הודעה תואמת
- ה. מיצגת את "time quantum" עם Round Robin עד  $time = 2$  -
4. **RoundRobin** – השופנו תכמה לחלקה שמחזיקה את הזמן  $time$  שונה לנו בחירה של כמה שניות יש לכל תהילר, מוגדר בסיס 2
- ש. גם שתי מתודות עזר:
1. anyReady – מחזירה אמת או שקר אם קיים לפחות תהילר אחד שיכול לרוץ בזמן הכללי זהה
  2. nextArrivalTime – מחזירה את הזמן הכי הקרוב בזמן הכללי הזה שיש תהילר שמתחליל בו
  3. onlyThisReady – מחזירה אמת או שקר אם התהילר הנוכחי הוא היחיד שיכול לרוץ
- RunProcess**
- א. בודקת תקינות הקלט ומוציאיה הודעה תואמת
  - ב. מתחילה את הזמן הכללי לתהילר הראשון במערך המומן
  - ג. כל עוד לא כל התחליכים סיימו:
  - ה. היא תבודוק אם התהילר הנוכחי לא יכול לרוץ –
1. אם לא היא תבודוק אם בכלל יש תהילר שיכול לרוץ – אם אין צזה היא תציג בזמן הקרוב ביותר ביתר
  2. תגדיל באחד את המעבר על המערך (ישפה טרייך שחזור על עצמו, אם נעשה פעולה מודולו על האינדקס של המערך, גם אם הוא ינסה לצאת מגודל המערך, הוא יחזיר לו 0)
- ה. תבודוק כמה זמן נותר לריצת התהילר הנוכחי
  - ו. אם התהילר הזה היחיד שיכול לרוץ (כל שאר התחליכים סיימו או זמן ההתחילה שלהם רחוק)
1. תחשב כמה זמן יש עד זמן ההתחילה של התהילר הבא
  2. אם הוא יותר קטן מהזמן  $time$ , ניקח את הזמן  $time$ , התהילר החדש יכנס בתור הבא.
  3. אם הוא גדול מהזמן  $time$  וזמן הריצה שנשאר לתהילר הנוכחי, היא תיקח את הזמן מזמן הריצה שנשאר לתהילר הנוכחי.
- ו. אם מספיק לו פחות זמן מהזמן  $time$  לשיום, היא תיקח אותו.

- ו. תקדם את הזמן הכללי בכמות הזמן הزاد ותעדכן את הזמן ששומש בתהיליך
- ו.i. אם התהיליך סיים היא תסימן אותו ותסכים את זמן הסיום שלו
- ו.ii. אם אין מישרו שיכל לרוץ אחרי הנוכחי, היא תקפוץ קדימה בזמן לבא בתור
- ו.p. מחשבת את ממוצע הROUND TURNAROUND ומוציאה הودעה תואמת
- SJF|SRTF 5.** מייצגת את "כמות הזמן כי נמדד שנותר ראשון" -
- בודקת תקינות הקלט ומוציאה הודעה תואמת
  - amatehlat מערך ומינית אותו עם ASCOMPARE
  - מכינה אובייקט comparator שהולך להשוות בין כמה זמן נשאר לתהיליך, זמן התחלתה ואז לפיה אינדקס
  - הובייקט comparator משומש בתור קביעת priority בטור עדיפות
  - amatehlat את הזמן הכללי Co
  - כל עוד לא עברנו על כל התהיליכים שיכולים להגיע או שיש תהיליך שמחכה לקבל זמן מעבד:
  - i. אם אין מישרו שמחכה למעבד כרגע, דרג קדימה בהזמן לזמן ההתחלה של התהיליך הבא בתור
  - ii. אם כלום לא רץ כרגע, הבא מהTOR את התהיליך עם הזמן כי קצר(שנותר) שיש לנו
  - iii. מעדכן את הזמן הגעה של התהיליך הבא, את הזמן שנותר לתהיליך הנוכחי ומתי הוא אמרו לס"ם
  - iv. אם התהיליך יספק לס"ם לפני התהיליך הבא:
    1. יעדכן את הזמן הכללי בהתאם.
    2. יעדכן את הזמן ששומש .
    3. יסמן את התהיליך כס"ם.
    4. יסכום את הזמן הסיום שלו.
  - .v. אם לא יספק:
    1. יירץ עד זמן ההגעה של הבא בתור
    2. יעדכן את הזמן הכללי
    3. יעדכן את הזמן ששומש
    4. יזכיר את הנוכחי למחסנית
- ו.g. מחשבת את ממוצע הROUND TURNAROUND ומוציאה הודעה תואמת.

## צילומי מסך של ריצות:

הרצת קובץ 1. הריצתLFCS , 19.25 ס"מPREEMPTIVE ב-LFCFS , 17.25 FCFS ס"מ לא .  
16.5 ס"מ PREEMPTIVE בע-SJF , 24.5 ס"מRR , 20.0 ס"מPREEMPTIVE בע-RR .

```
① README.md  J FCFS.java X
MainPackage > Runners > J FCFS.java ...
1 // Names: Oz Ben Moshe, David Norman, IDs: 208639906, 206395592
2 package MainPackage.Runners;
3
4 import MainPackage.ProcessPackage.ProcessPCB;
5 import java.util.Collection;
6
7 public class FCFS implements ProcessRunner{
8
9 /**
10 * First-Come First-Served (non-preemptive);
PROBLEMS ⑧ OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\Ozbm1\projects\OS\C-project-05\ex03> java -jar .\TurnAroundCalculator.jar "C:\Users\Ozbm1\projects\OS\C-project-05\ex03\example_files\input1.txt"
FCFS: mean turnaround = 17.25
LCFS (NP): mean turnaround = 19.25
LCFS (P): mean turnaround = 20.0
RR: mean turnaround = 24.5
SJF (P): mean turnaround = 16.5
PS C:\Users\Ozbm1\projects\OS\C-project-05\ex03> 
```

The screenshot shows a terminal window with the following content:

```
terminal Help < > Q ex03
...
① README.md J FCFS.java x
MainPackage > Runners > J FCFS.java > ...
1 // Names: Oz Ben Moshe, David Norman, IDs: 208639906, 206395592
2 package MainPackage.Runners;
3
4 import MainPackage.ProcessPackage.ProcessPCB;
5 import java.util.Collection;
6
7 public class FCFS implements ProcessRunner{
8
9     /**
10      * First-Come First-Served (non-preemptive):
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\Ozbm1\projects\OS\C-project-05\ex03> java -jar .\TurnAroundCalculator.jar "C:\Users\Ozbm1\projects\OS\C-project-05\ex03\example_files\input2.txt"
FCFS: mean turnaround = 28.125
LCFS (NP): mean turnaround = 22.75
LCFS (P): mean turnaround = 24.625
RR: mean turnaround = 36.625
SJF (P): mean turnaround = 21.625
PS C:\Users\Ozbm1\projects\OS\C-project-05\ex03>
```

3. הרצת קובץ PREEMPTIVE ב-11.5 ס"מ עם FCFS, LFCFS ו-PREEMPTIVE (3 ס"מ) היא הגדולה ביותר.

The screenshot shows a Java IDE interface with the following details:

- File**: README.md
- Editor Tab**: FCF.java
- Code Content (FCF.java):**

```
1 // Names: Oz Ben Moshe, David Norman, IDs: 208639906, 206395592
2 package MainPackage.runners;
3
4 import MainPackage.ProcessPackage.ProcessPCB;
5 import java.util.Collection;
6
7 public class FCFS implements ProcessRunner{
8
9     /**
10      * First-Come First-Served (non-preemptive):
```

- Terminal Output:**

```
PS C:\Users\Ozben\projects\OS\c-project-OS\ex03> java -jar .\TurnAroundCalculator.jar "C:\Users\Ozben\projects\OS\C-project-OS\ex03\example_files\input3.txt"
FCFS: mean turnaround = 11.5
LCFS (NP): mean turnaround = 11.5
LCFS (P): mean turnaround = 11.5
RR: mean turnaround = 11.5
SJF (P): mean turnaround = 11.5
PS C:\Users\Ozben\projects\OS\c-project-OS\ex03>
```

.4 הרצת קובץ 4 סעיפים ב-15.0 PREEMPTIVE ,14.0 סעיף ב-LFCFS ,14.0 סעיף ב-PREEMPTIVE SJF ,20.5 סעיף ב-RR ,17.5 סעיף ב-PREEMPTIVE

```

1 // Names: Oz Ben Moshe, David Norman, IDs: 208639906, 206395592
2 package MainPackage.Runners;
3
4 import MainPackage.ProcessPackage.ProcessPCB;
5 import java.util.Collection;
6
7 public class FCFS implements ProcessRunner{
8
9     /**
10      * First-Come First-Served (non-preemptive);

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\Ozben\projects\OS\C-project-05\ex03> java -jar .\TurnAroundCalculator.jar "C:\Users\Ozben\projects\OS\C-project-05\ex03\example\_files\input4.txt"

FCFS: mean turnaround = 14.0  
LCFS (NP): mean turnaround = 15.0  
LCFS (P): mean turnaround = 17.5  
RR: mean turnaround = 20.5  
SJF (P): mean turnaround = 14.0

PS C:\Users\Ozben\projects\OS\C-project-05\ex03>

.5 הרצת קובץ 5 סעיפים ב-6.0 PREEMPTIVE ,6.0 סעיף ב-LFCFS ,6.0 סעיף ב-PREEMPTIVE SJF ,7.5 סעיף ב-RR ,6.5 סעיף ב-PREEMPTIVE

```

1 // Names: Oz Ben Moshe, David Norman, IDs: 208639906, 206395592
2 package MainPackage.Runners;
3
4 import MainPackage.ProcessPackage.ProcessPCB;
5 import java.util.Collection;
6
7 public class FCFS implements ProcessRunner{
8
9     /**
10      * First-Come First-Served (non-preemptive);

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\Ozben\projects\OS\C-project-05\ex03> java -jar .\TurnAroundCalculator.jar "C:\Users\Ozben\projects\OS\C-project-05\ex03\example\_files\input5.txt"

FCFS: mean turnaround = 6.0  
LCFS (NP): mean turnaround = 6.0  
LCFS (P): mean turnaround = 6.5  
RR: mean turnaround = 7.5  
SJF (P): mean turnaround = 6.0

PS C:\Users\Ozben\projects\OS\C-project-05\ex03>

.6

## README.md

Turn around time Calculator

how to run?

```
java -jar TurnAroundCalculator.jar <your file path (full) here>
```

how to build jar(in case needed):

```
& "C:\Program Files\Java\jdk-25.0.2\bin\jar.exe" cfe
.\TurnAroundCalculator.jar MainPackage.ProcessPackage.TurnAroundCalculator
-C .\build-jar .
```