SPL – עבודה 3

תוכן :

1 – תיאור כללי:

בעבודה זו תידרשו לבנות סימולציה של קרן השקעות נדלן (Real Estate Investment Trust – REIT) בשם BGU-REIT, אשר בבעלותה ובתפעולה נדל״ן רווחי.
החברה מעסיקה פקידים רבים ועובדי תחזוקה.
BGU-REIT מקבלת רשימה של בקשות רכישה, ומשימתה היא לבצע סימולציה של תהליך ההשכרה של הנכס המבוקש.
כל השכרה תניב לחברה הכנסה. הכנת נכס ללקוח מחייבת כלים שונים וכלי תחזוקה שונים.
משימתכם היא לוודא שלעובדי התחזוקה יש את הכלים הדרושים וכלי התחזוקה.
תנהלו את תהליך הסימולציה, החל מיצירת לקוחות שמעבירים דרישות לפקידים. הפקידים ינסו למלא דרישות אלה, ע״י חיפוש אחר נכס פנוי המתאים לדרישותיהם. אם דרישה כזו לא יכולה להיענות, הלקוח יכנס לרשימת המתנה, עד שהנכס הרצוי יתפנה. בסוף השכירות, הלקוח ימלא דו״ח נזק ויעביר אותו להנהלה, אשר תשלח את עובדי התחזוקה לתקן את הנזקים לפי הצורך.
שכירות מוצלחת תניב לBGU-REIT הכנסה. בסוף התהליך, עליכם להדפיס סטטיסטיקה אשר תייצג את מידת ההצלחה של BGU-REIT בעסקה – הכסף שהורווח, הכסף שהושקע וזמן ביצוע כל דרישה.
שימו לב שנסביר בפירוט את רוב חלקי המערכת, אולם שמרנו מספר חלקים בערפל ע״מ לאפשר לכם את החופש באופן יישומם. אם אין הסבר מפורש של סעיף מסוים, הכוונה היא שיש לכם את החופש ליישם אותו כראות עיניכם, כל עוד הוא מספק את התוצאה הנדרשת.
הערה: נמצא ב- (Found In), מתאר משהו שנמצא במחלקה כלשהי, אבל לא מוגבל לכך. באפשרותכם להוסיף את אותו פריט בכל מקום שתמצאו לנכון.
הערה: שדות ושיטות המתוארות למטה הן **בגדר חובה**, וחייבות להיות כלולות ביישום שלכם. אולם, ניתן לכם החופש להוסיף שדות ומחלקות נוספות כרצונכם.

2 – מבנים:

בחלק זה, נפרט את החלקים השונים שנמצאים במערכת, הפירוט של כל חלק ותפקידו. שדות המצוינים עבור אובייקט כלשהו הם שדות חובה, אולם מותר להוסיף שדות נוספים.

2.1 – נכסים:

נמצא ב: ניהול (Management).
אובייקט זה יחזיק את האינפורמציה של נכס ספציפי (יחיד).
שדות חובה:

1. Name
2. Type
3. Location
4. Collection of AssetContent
5. Status
6. Cost per night
7. Size

Type:
סוג הנכס יהיה אחד מרשימה של סוגי נכסים שתתקבל בקובץ קלט (Input File).
Location:
קואורדינטות יישמרו כ (x,y), קואורדינטות במרחב דו-מימדי. המרחק יחושב כמרחק האוקלידי בין דתי קואורדינטות שונות.

$$dist((x, y), (a, b)) = \sqrt{(x - a)^2 + (y - b)^2}$$

Status:
אחת משלוש האפשרויות הבאות:

1. AVAILABLE
2. BOOKED
3. OCCUPIED
4. UNAVAILABLE

כאשר נכס פנוי אם הוא לא בשימוש, מוזמן (Booked) כאשר הנכס מוזמן ע"י קבוצת לקוחות אך עוד לא תפוס, תפוס כאשר הנכס נמצא כרגע בשימוש ע"י לקוח, ולא זמין אם בלתי אפשרי להשכיר את הנכס, כל עוד הוא לא תוקן.
תיקון נכס מחייב כלי עבודה, חומרי גלם מתכלים ותהליך זה יפורט בהמשך.
Size:
מספר הנפשות שיכולות לגור בנכס.

2.2 – AssetContent:

נמצא ב: Asset.
Asset content הוא אובייקט שמחזיק פרטים של פריט אחד שנמצא בנכס. נכס בדרך כלל יכיל אוסף (Collection) של פריטים אלה.
האובייקט יכיל את השדות הבאים:

1. Name
2. Health
3. Repair Cost multiplier

Health:
מצב הנכס יתדרדר תוך כדי שימוש בו. המשתנה מייצג את המצב של פריט בודד בנכס.

Repair Cost multiplier:
חישוב הזמן שיידרש מאנשי התחזוקה לתקן את הפריט בנכס, מחושב ע"י:
(100-Health)*Repair cost multiplier.
ככל שרכיב עלות התיקון (Multiplier) גבוה יותר, כך יגדל זמן התיקון.
Health יהיה מספר, לא בהכרח שלם, המתחיל מ-100 ונע בין 0 ל-100 (כולל הקצוות).

2.3 – Assets:

נמצא ב: Management, RunnableClerk
אובייקט זה יחזיק אוסף (Collection) של נכסים (Assets), ויכיל שיטות הקשורות לנכסים.
יש שני אובייקטים במחלקה זו:

1. קבלת רשימה של הנכסים הפגומים (Retrieve the list of damaged assets).
2. שיטת חיפוש שתחזיר לפקיד (Clerk) נכס אשר מתאים לסוג ולדרישות.

2.4 – RepairToolInformation:

נמצא ב: Management.
אובייקט זה יחזיק את האינפורמציה הנוגעת לכלי עבודה יחיד. לכל כלי עבודה (Tool) יש:

1. Name
2. Quantity

מטרתו של פריט זה היא לאכסן את המידע על כלי מסוים הנדרש לתיקון רכיב תוכן (Content Item) של נכס (Asset) מסוים.
<u>דוגמא</u>: ייתכן שתיקון תנור (Stove) ידרוש:
Hammer 1
Drill 1
Screw Drivers 2
ואז אובייקט זה מאכסן:
<Hammer, 1><Drill, 1><Screw Driver, 2> באוסף (Collection) ומקושר לתנור.
הערה: אין לשמור את זה במחסן (Warehouse).

2.5 – RepairMaterialInformation:

נמצא ב: ניהול.
אובייקט זה יחזיק את האינפורמציה של חומר גלם (Repair Material) יחיד. כל חומר גלם יכיל את השדות הבאים:

1. Name
2. Quantity

מטרתו של פריט זה היא לאכסן את המידע על חומר גלם מסויים הנדרש לתיקון רכיב תוכן (Content Item) של נכס (Asset) מסוים.
<u>דוגמא</u>: ייתכן שתיקון תנור (Stove) ידרוש:
Nails 10
Screws 5
Sand Cloth 1
ואז אובייקט זה מאכסן:
<Nail, 10><Screw, 5><Sand Cloth, 1> באוסף (Collection) ומקושר לתנור.
הערה: אין לשמור את זה במחסן (Warehouse).

Warehouse – 2.6:

נמצא ב: Management, RunnableMaintainenceRequest.
אובייקט זה מכיל:
1. Collection of repair tools
2. Collection of repair materials

המחסן הוא מקום האכסון המשותף (Shared Storage Component), בו אנשי תחזוקה שונים משיגים את הכלים והחומרים הדרושים להם.
יהיו לכם שיטות שיאפשרו לאנשי התחזוקה להשיג (Acquire) ולשחרר (Release) כלים, וכן לצרוך חומרי גלם.
הערה: המחסן מכיל מספיק חומרי גלם וכלים ע״מ לסיים את תהליך הסימולציה בהצלחה. אין **שום** מקרה בו הסימולציה תיתקע בגלל חוסר כלים או חומרי גלם בו.

RepairTool – 2.7:

נמצא ב: Warehouse.
אובייקט זה מיועד להיות במחסן. הוא יכיל את שם הכלי ואת הכמות הנוכחית שלו במחסן.
וודאו בטיחות תהליכית (Thread Safety) באובייקט זה. למה? כיצד?

RepairMaterial – 2.8:

נמצא ב: Warehouse.
אובייקט זה מיועד להיות במחסן.
הוא יחזיק את שם חומר הגלם, ואת הכמות שנמצאת במחסן.
וודאו בטיחות תהליכית (Thread Safety) באובייקט זה.
במה הוא שונה מהיישום עבור כלי עבודה?

Location – 2.9:

נמצא ב: ClerkDetails, Assets.
אובייקט פשוט המכיל קואורדינטה (x,y) במרחב דו מימדי ויהיה בו שימוש במקומות מתאימים בתוכנית.
יישמו שיטה בשם: **calculateDistance(Location other)** המקבלת מיקום אחר ומחשבת את המרחק בין שני המיקומים.
הערה: כיוון שתקבלו כפרמטר אובייקט מאותה מחלקה, תוכלו לגשת לשדות שלו באופן ישיר, אפילו אם הם פרטיים.

CustomerGroupDetails – 2.10:

נמצא ב: RunnableCustomerGroupManager.
אובייקט זה יחזיק את הפרטים של קבוצות הלקוחות כפי שנקרא (Parsed) מקבצי הקלט (Input Files).
שדות:
1. Collection of Renal Requests
2. Collection of Customers
3. Group manager name

שיטות:
1. addCustomer
2. addRentalRequest.

2.11 – ClerkDetails:

נמצא ב: RunnableClerk.
אובייקט זה יחזיק את השדות הבאים:
1. Name
2. Location

המידע יקרא (Parsed) מקבצי הקלט (Input Files) וישמר באובייקט זה.
האובייקט ייעטף (Wrapped) ע״י RunnableClerk וכן RunnableClerk יעשה בו שימוש.

2.12 – DamageReport:

נמצא ב:
דו״ח נזק ייכתב ברגע שCustomerGroup יסיימו לסמלץ את הCallableSimulateStayInAsset שלהם, ואחוז הנזק יוחזר (Returned) ואז הRunnableCustomerGroupManager יצור את אובייקט הDamageReport שיחזיק את השדות הבאים:
1. Asset
2. Damage percentage.

אובייקט זה יישלח להנהלה (Management) להמשך טיפול.
אחוז הנזק (Damage Percentage) בדו״ח הנזק הוא סכום של כל אחוזי הנזק שהוחזרו ע״י כל הלקוחות בקבוצה.

2.13 – RentalRequest:

נמצא ב: Management.
אובייקט זה יחזיק את האינפורמציה הנוגעת לבקשת שכירות (Rental Request).
שדות:
1. Id
2. Asset Type
3. Asset Size
4. Duration of stay
5. Asset
6. Request status

Asset type:
סוג הנכס המבוקש ע״י מנהל קבוצת הרכישה (Customer group manager).
Asset size:
גודל הנכס המבוקש ע״י מנהל קבוצת הרכישה.
הערה: הנכס שיוזמן (Booked) יכול להיות גדול מהגודל המבוקש, אולם וודאו שאתם בוחרים בקטן ביותר מבין האפשריים.
Duration of stay:
הזמן, בימים, כאשר בכל יום יש 24 שניות בתהליך הסימולציה שלנו.
Asset:
הנכס עצמו, שנמצא ע״י RunnableClerk ויעשה בו שימוש ע״י RunnableCustomerGroupManager.
Request status:
אחת מארבע האפשרויות:
א. INCOMPLETE:
אם הבקשה טרם טופלה.
ב. FULFILLED:
אם הבקשה בוצעה, אבל הלקוח עדיין לא השתמש בה.
ג. INPROGRESS:

הלקוח משתמש בנכס ברגע זה.

ד. COMPLETE:

הלקוח עזב את הנכס.

הערה: האובייקט יישלח לCallableSimulateStayInAsset אשר יבצע סימולציה של תהליך ההשכרה של הבקשה.

3 – אובייקטים פסיביים (Passive Objects)

בחלק זה, נפרט על הפסיביים השונים במערכת, תוכן כל אחד מהם ומשימותיהם.
שדות המפורטים עבור אובייקט מסויים הם חובה, אולם מותר להוסיף שדות נוספים אם תרצו.

3.1 – Management:

אובייקט זה יכיל את השדות הבאים:

1. Collection of clerk details
2. Collection of customer group details
3. Assets
4. Warehouse
5. Collection of RepairToolInformation
6. Collection of RepairMaterialInformation

שיטות:

addClerk()
addCustomerGroup()
addItemRepairTool()
addItemRepairMaterial()

הערה: מהי הדרך הטובה ביותר לשמור כל אוסף (Collection)? Hashmap? ArrayList? Vector? Queue?

ה-Management מבצעת את הסימולציה של המערכת באופן הבא:

1. מריצה (Runs) כל פקיד (Clerk) כRunnableClerk כתהליכון (Thread).
   האם צריך Executor? אני יודע שאתם רוצים! אבל האם זה נחוץ?
2. מריצה כל קבוצת לקוחות (CustomerGroup) כRunnableCustomerGroupManager, כתהליכון (Thread). האם צריך Executor?
3. כל עוד יש RentalRequests שמחכות לטיפול:
   א. מחכה שכל RunnableClerk יסיים את היום שלו. (איך?)
   ב. עבור כל Damage report – מקבלת את הנכס ומבצעת new RunnableMaintainenceRequest אשר מבצעת סימולציה של תיקון הנכס. (האם צריך Executor?)
   ג. ברגע שהטיפול הסתיים, הפקידים מקבלים הודעה על התחלת משמרת חדשה (איך?).

3.2 – Customer:

האובייקט מכיל את השדות הבאים:

1. Name
2. Vandalism Type
3. Minimum Damage
4. Maximum Damage

כאשר סוג הוונדליזם הוא אחד מבין:

1. ARBITRARY
2. FIXED
3. NONE

כמפורט בסעיף 2.12.
הלקוח מחשב את אחוז הנזק לנכס. אם הנזק הוא מסוג קבוע (Fixed), אז הממוצע בין MinimumDamage ו-MaximumDamage יוחזר.
הלקוח לא משנה את מצב (Health) הנכס, רק ה-RunnableCustomerGroupManager.

3.3 – Statistics:

נמצא ב: Management.
אובייקט זה יכיל את השדות הבאים:
1. Money Gained
2. Rentals
3. Collection of Repair tools
4. Collection of repair materials.

Money Gained:
ההכנסה מההשכרות.
Rental Requests:
אוסף של בקשות שכירות, כולל האינפורמציה שלהם.
Repair Tools:
הכלים שידרשו לתהליך.
Repair Materials:
חומרי הגלם שיצרכו וכמותם.

ניתן לבצע עידכון לאובייקט זה מכל מקום בתוכנה, אולם מומלץ לבצע זאת במספר מינימלי של מקומות.

4 – אובייקטים אקטיביים:

4.1 – RunnableClerk:

אובייקט זה יכיל את השדות הבאים:
1. Clerk Details
2. Collection of RentalRequests
3. Number of rental requests

Clerk Details:
פרטי הפקיד.
Collection of RentalRequests:
אוסף משותף (Shared) בין כל הפקידים שנמצא בתוכנית (Application), וכל תא (Slot) מכיל Rental Request.
Number of Rental Requests:
מחזיק את מספר בקשות השכירות (Rental Requests) שטרם טופלו (that are yet handled).
מאיזה סוג זה יהיה? למה זה חשוב? איפה אתם הולכים להשתמש בזה?

הפקיד יבצע סימולציה של מהלך יום (day cycle) בחייו:
1. בתחילת היום, הפקיד ייקח Rental Requests חדשים מהאוסף המשותף.
2. לאחר מכן, הפקיד יחפש ויאתר נכס המתאים לבקשה.
3. ברגע שנכס כזה נמצא, הפקיד יבצע סימולציה של הגעה לנכס ע״י חישוב המרחק האוקלידי בין המיקום שלו למיקום הנכס, וביצוע שינה מספר שניות התואם למרחק.

לדוגמא:
אם הפקיד נמצא ב(1.1) והנכס נמצא ב(3.1) המרחק הוא 2. הפקיד ישן 4 שניות (החישוב הוא הלוך-חזור).
4. רק אז, הפקיד יודיע ללקוח שהנכס נמצא.

כל עוד זמן השינה הכולל קטן מ-8 שניות, הפקיד ממשיך לטפל בבקשה. ברגע שמספר השניות עובר את 8, הפקיד יכנס למצב המתנה (Wait mode), עד שיקבל (Notified) משמרת חדשה ע״י ה-Management.

הפקיד מפסיק לעבוד כאשר הבקשות לטיפול (Rental Requests) יורדות ל-0.
הערה: על הנכס שנמצא ע"י הפקיד להיות מאוכסן באובייקט הRentalRequest. באופן זה, הRunnableCustomerGroupManager יכול לקבל את הנכס המדובר בקלות, ולבצע סימולציה של השהייה בו.

4.2 – RunnableCustomerGroupManager:

אובייקט זה מכיל את השדות הבאים:
1. CustomerGroupDetails

המנהל מקבל את הRentalRequest מהקבוצה ע"י אובייקט ה-CustomerGroupDetails.
פעולת המנהל:
כל עוד יש RentalRequests:
1. מזין Rental Request לManagement, באופן זה הבקשה יכולה להיגע לRunnableClerks.
2. מחכה עד שהRentalRequest בוצעה ע"י אחד מהRunnableClerks שרצים כרגע (איך?)
3. ברגע שהבקשה בוצעה, הוא יוצר CallableSimulateStayInAsset עבור כל Customer בקוצה.
4. מחכה עד שכל הCallableSimulateStayInAsset בוצעו.
   הערה: אתם יכולים להשתמש בExecutorCompletionService, דוגמא לשימוש נמצאת כאן.
5. משלב את הנזק שהוחזר ע"י כל CallableSimulateStayInAsset לערך אחד, **ומעדכן** את הנכס בנזק שנוצר ע"י הלקוחות.
6. מייצר (Generates) DamageReport, אשר נשלח להנהלה.

התהליך ממשיך עד שכל הRental Requests בקבוצה בוצעו, והסימולטור סיים.

4.3 – RunnableMaintenanceRequest:

נמצא ב: Management
הערה: חייב להיות Runnable.
אובייקט זה יחזיק את השדות הבאים (Current fields):
1. Collection of RepairToolInformation.
2. Collection of RepairMaterialInformation.
3. Asset.
4. Warehouse.

מהנכס (Asset) אנו נשיג את הפריטים הפגומים בו (Damaged contents).
פריט נחשב פגום אם המצב (Health) שלו מתחת ל65%.
לאחר מכן, "עלות" החישוב הכוללת, **ביחידות זמן**, מחושב, כאשר עבור כל פריט אחוז הנזק מוכפל ע"י כופל עלות התיקון (RepairCost) שנמצא בAssetContent.
לאחר מכן:
1. השג את הכלים הדרושים לתיקון ואת הכמות שלהם (במחסן תמיד יהיו מספיק כלים).
2. השג את החומרים הדרושים ואת הכמות שלהם (במחסן תמיד יהיו מספיק חומרי גלם).
3. תישן (Sleep) את העלות בMilliseconds אחרי עיגול המספר לLong הקרוב ביותר.
4. שחרר את הכלי עבודה שהיו בשימוש.
5. סמן את הנכס כמתוקן, ע"י החזרת המצב (Health) של כל הפריטים שלו ל100.

אתם תיצרו RunnableMaintenanceRequest אחת לכל נכס פגום.

4.4 – CallableSimulateStayInAsset:

נמצא ב: RunnableCustomerGroupManage.

ה-Callable הזה יבצע סימולציה של שהות של לקוח **אחד** בנכס. לאחר סיום הסימולציה, אחוז הנזק לנכס מחושב ומוחזר.

חישוב אחוז הנזק:

חישוב אחוז הנזק מבוצע פעם אחת לכל נכס, לכל לקוח. חישוב הנזק מבוצע באופן הבא, בהתאם לסוג הוונדליזם:

ARBITRARY:

במקרה זה האחוז יחושב ע"י בתחום של minimumDamage ו-maximumDamage, באופן רנדומלי.

FIXED:

במקרה זה יוחזר הממוצע בין הנזק המינימלי למקסימלי.

NONE:

במקרה זה, יוחזר נזק של שחיקה בלבד, בסך 0.5%.

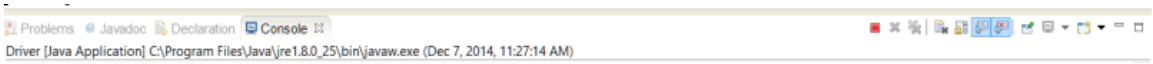
- שינה מבוצעת ע"י המרת כל 24 שעות ל24 שניות.

5 – מהלך התוכנית – Program Cycle

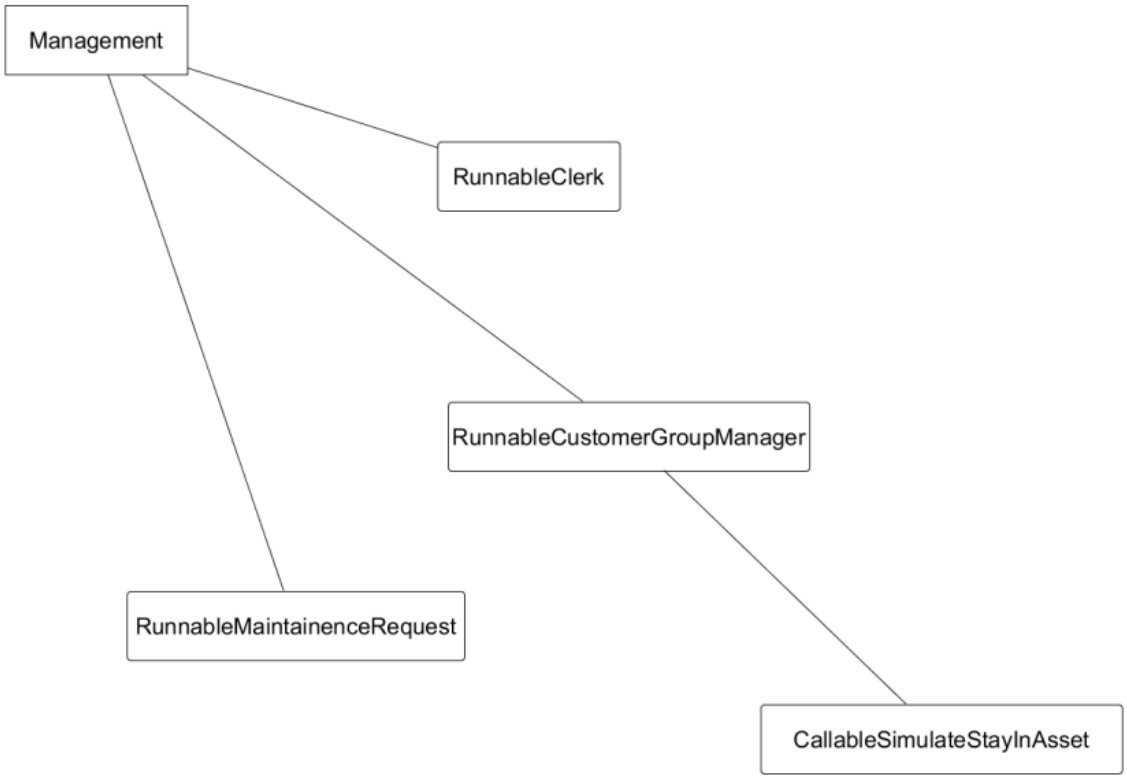
ראשית, עליכם לקרוא (Parse) 4 קבצים וליצור אובייקטים לשמירת האינפורמציה שנקראה מקבצים אלה. הקבצים יכילו אינפורמציה הנוגעת לחלקים שונים במערכת, כמפורט באופן מלא בחלק 8.
ברגע שהקריאה הסתיימה, תתחילו את מהלך הסימולציה, ותקראו (Call) לManagement.
הManagement יפעיל (Launch) את הCustomerGroupManagers, Clerks, Maintenance Persons, כאשר כל Customer Group מכיל אוסף של בקשות השכירות (Rental Requests) שלו.
הפקידים ייקחו בקשות שכירות מהתור, יימלאו אותן ויספקו אותן ללקוח.
הלקוח ישהה בנכס ובסוף השהות יצור דו"ח נזק (Damage Report), אשר יישלח לManagement.
הManagement תעביר את דו"ח הנזק לאחד מאנשי התחזוקה לביצוע התיקון.
אנשי התחזוקה יבצעו תיקון של הנכסים.
והמעגל ימשיך.
התוכנה (Application) תחכה לתהליך להסתיים, ורק אז עליה להיסגר באופן מלא.
וודאו שאתם מכבים את כל הExecutors ואת התהליכונים (Threads) שייתכן שעדיין עובדים בתוכנה.

6 – תהליך הכיבוי – Shutdown Process

זיכרו: יציאה מפונקציית הMain לא גורם לתהליכונים (threads) אחרים להפסיק לרוץ.
תצטרכו לאפשר כיבוי בחינניות (Gracefully) ברגע שהתוכנה מסיימת את מעגל החיים שלה. כאשר המנהל יודע שבקשת השכירות האחרונה בוצעה, עליכם להדפיס את תכולת הStatistics object, ולכבות את האפליקציה, כלומר עליכם להתמודד עם כיבויי כל התהליכונים שנמצאים בתוכנית, וכיבויי כל הExecutors בהם אתם משתמשים. הדבר צריך להיות מבוצע בחינניות. כלומר בלי יציאות פתע מתהליכונים. ריצת תהליכות (Thread flow) צריכה להגיע לסיומה.
כאשר סיימתם, אם אתם עדיין רואים את הריבוע האדום באקליפס, לא סגרתם את התוכנה כמו שצריך...



7 – דיאגרמת יחסים – Relations Diagram



8 – קבצי קלט וקריאתם – Input Files and Parsing

8.1 – קבצי קלט – Input files:

אתם יכולים להניח כי הקלט תקין. אתם יכולים להניח שהסדר של המידע יהיה כפי שתראו בדוגמא זו.

ישנם 4 קבצי קלט שונים:

1. **InitialData:**

א. מחסן המכיל רשימה של כלים ורשימה של חומרי גלם.

ב. צוות שמכיל פקידים ומספר אנשי התחזוקה.

```
<REIT>
        <Warehouse>
                <Tools>
                        <Tool>
                                <Name></Name>
                                <Quantity></Quantity>
                        </Tool>
                            ⋮
                        <Tool>
                                <Name></Name>
                                <Quantity></Quantity>
                        </Tool>
                </Tools>
                <Materials>
                        <Material>
                                <Name></Name>
                                <Quantity></Quantity>
                        </Material>
                            ⋮
                        <Material>
                                <Name></Name>
                                <Quantity></Quantity>
                        </Material>
                </Materials>
        </Warehouse>
        <Staff>
                <Clerks>
                        <Clerk>
                                <Name></Name>
                                <Location x="" y=""/>
                        </Clerk>
                            ⋮
                        <Clerk>
                                <Name></Name>
                                <Location x="" y=""/>
                        </Clerk>
                </Clerks>
                <NumberOfMaintenancePersons></NumberOfMaintenancePersons>
                <TotalNumberOfRentalRequests></TotalNumberOfRentalRequests>
        </Staff>
</REIT>
```

2. AssetContentsRepairDetails:

קובץ זה מכיל את האינפורמציה הנוגעת לפריטים השונים שיכולים להימצא בנכסים, והכלים וחומרי הגלם הדרושים לתיקון הנכס במקרה של נזק.

```
<AssetContentsRepairDetails>
        <AssetContent>
                <Name></Name>
                <Tools>
                        <Tool>
                                <Name></Name>
                                <Quantity></Quantity>
                        </Tool>
                            ⋮
                        <Tool>
                                <Name></Name>
                                <Quantity></Quantity>
                        </Tool>
                </Tools>
                <Materials>
                        <Material>
                                <Name></Name>
                                <Quantity></Quantity>
                        </Material>
                            ⋮
                        <Material>
                                <Name></Name>
                                <Quantity></Quantity>
                        </Material>
                </Materials>
        </AssetContent>
                ⋮
        <AssetContent>
                <Name></Name>
                <Tools>
                        <Tool>
                                <Name></Name>
                                <Quantity></Quantity>
                        </Tool>
                            ⋮
                        <Tool>
                                <Name></Name>
                                <Quantity></Quantity>
                        </Tool>
                </Tools>
                <Materials>
                        <Material>
                                <Name></Name>
                                <Quantity></Quantity>
                        </Material>
                            ⋮
                        <Material>
                                <Name></Name>
                                <Quantity></Quantity>
                        </Material>
                </Materials>
        </AssetContent>
</AssetContents>
```

3. Assets:

קובץ זה יכיל רשימה של נכסים, ואת האינפורמציה הנוגעת להם.

```
<Assets>
        <Asset>
                <Type></Type>
                <Size></Size>
                <Location x=" " y=" "/>
                <CostPerNight></CostPerNight>
                <AssetContents>
                        <AssetContent>
                                <Name></Name>
                                <RepairMultiplier></RepairMultiplier>
                        </AssetContent>
                                ⋮
                        <AssetContent>
                                <Name></Name>
                                <RepairMultiplier></RepairMultiplier>
                        </AssetContent>
                </AssetContents>
        </Asset>
            ⋮
        <Asset>
                <Type></Type>
                <Size></Size>
                <Location x=" " y= " "/>
                <CostPerNight></CostPerNight>
                <AssetContents>
                        <AssetContent>
                                <Name></Name>
                                <RepairMultiplier></RepairMultiplier>
                        </AssetContent>
                                ⋮
                        <AssetContent>
                                <Name></Name>
                                <RepairMultiplier></RepairMultiplier>
                        </AssetContent>
                </AssetContents>
        </Asset>
</Assets>
```

4. CustomerGroups:

קובץ זה יכיל רשימה של Customer Groups, ה-Rental Requests שלהם והאינפורמציה עליהם.

```
<Customers>
    <CustomerGroups>
        <CustomerGroupDetails>
            <GroupManagerName></GroupManagerName>
            <Customers>
                <Customer>
                    <Name></Name>
                    <Vandalism></Vandalism>
                    <MinimumDamage></MinimumDamage>
                    <MaximumDamage></MaximumDamage>
                </Customer>
                    ⋮
                <Customer>
                    <Name></Name>
                    <Vandalism></Vandalism>
                    <MinimumDamage></MinimumDamage>
                    <MaximumDamage></MaximumDamage>
                </Customer>
            </Customers>
            <RentalRequests>
                <Request id=" ">
                    <Type></Type>
                    <Size></Size>
                    <Duration></Duration>
                </Request>
                    ⋮
                <Request id=" ">
                    <Type></Type>
                    <Size></Size>
                    <Duration></Duration>
                </Request>
            </RentalRequests>
        </CustomerGroupDetails>
            ⋮
```

```
                    ⋮
            <CustomerGroupDetails>
                    <GroupManagerName></GroupManagerName>
                    <Customers>
                            <Customer>
                                    <Name></Name>
                                    <Vandalism></Vandalism>
                                    <MinimumDamage></MinimumDamage>
                                    <MaximumDamage></MaximumDamage>
                            </Customer>
                                    ⋮
                            <Customer>
                                    <Name></Name>
                                    <Vandalism></Vandalism>
                                    <MinimumDamage></MinimumDamage>
                                    <MaximumDamage></MaximumDamage>
                            </Customer>
                    </Customers>
                    <RentalRequests>
                            <Request id=" ">
                                    <Type></Type>
                                    <Size></Size>
                                    <Duration></Duration>
                            </Request>
                                ⋮
                            <Request id=" ">
                                    <Type></Type>
                                    <Size></Size>
                                    <Duration></Duration>
                            </Request>
                    </RentalRequests>
            </CustomerGroupDetails>
        </CustomerGroups>
</Customers>
```

8.2 – קריאה מקבצים – Parsing:

Parsing זהו האקט של קריאת טקסט והמרתו לפורמט תוך-זיכרון יעיל יותר, "הבנה" של המידע ברמה מסוימת.
XML Parser לדוגמה, מפרסר קבצי XML. HTML parser מפרסר קבצי Html.
במקרה שלנו, אנו נפרסר קבצי XML בפורמט המתואר למעלה.
הקבצים יהיו בתבנית XML מדויקט, ודיוק הקלט ניתן להנחה, דבר המקל על תהליך הקריאה.
אנא ודאו שאתם קוראים את המדריך לJava xml parser API לפני נסיון לפרסר קבצים כלשהם.
עליכם ליישם מחלקה בשם "Driver" אשר מיישמת את פונקציית הMain שלכם.
היא מקבלת את שמות הקבצים כפרמטרים, קוראת לפונקציות הסטטיות המתאימות ע"מ לפרסר את הקבצים השונים, ויוצרת את האובייקטים הדרושים. ברגע שכל האובייקטים הדרושים נוצרו, פונקציית הmain מבצעת את תהליך הסימולציה [executes the simulation process (Management)].
על המנהל לא לקרוא שום קובץ (או לפרסר). הקריאה חייבת להיעשות במחלקה **אחת** בלבד. פרסור יכול להתבצע ע"י שימוש במספר פונקציות סטטיות במחלקה סטטית חדשה (in a new static class),

אשר מכילה את פונקציות הפירסור הסטטיות השונות, ועליה להיקרא מהMain, לפני התחלת תהליך הסימולציה.
אסור על הMain לקרוא או לפרסר את הקבצים, הוא מקבל את האובייקטים ומפעיל את תהליך הסימולציה (למה?)

9 – toString()

כל האובייקטים שנמצאים באובייקט Statistics צריכים ליישם שיטת toString(), אשר מחזירה מחרוזת מרוכזת של שדות האובייקט. כאשר מריצים את toString() של האובייקט Statistics, מוחזר הדפס של כל תכולתו בפורמט קריא לבני אדם.
StringBuilder זמין לרשותכם ע"מ לרכז את האלמנטים בלולאה.
שימוש ב"+" הוא גרוע (למה?).
כאשר תרצו להדפיס את התוכן של האובייקט Statistics שאתם מחזיקים, כל שתצטרכו לעשות זה להדפיס את toString() שלו, ובתוכה תריצו את toString() של כל האובייקטים שמוכלים בו, תרכזו את כולם ותחזירו את התוצאה.

10 – דרישות חובה

עליכם להשתמש בכל אלה בתוכנה שלכם:

1. Blocking Queue
2. Semaphore
3. Guarded Blocks
   מה קורה כאשר מבצעים notify() והתהליכון לא רוצים לבצע notify() **אינו** במצב המתנה (Wait mode)? האם זאת בעיה? מה לגבי notify() לעומת notifyAll()? מתי משתמשים בכל אחת מהן?
4. CountDownLatch
5. Executor Service:
   Executors מטרתם לנהל תהליכונים. בכל פעם שתרצו להשתמש בexecutor, שאלו את עצמכם: האם אני **צריך** executor שינהל את התהליכונים שלי? אם התשובה היא **לא**, אל תשתמשו בExecutor!
6. Treads
7. Atomic:
   Java מספקת אובייקטים אטומיים אשר מסייעים לכם לשמור על בטיחות תהליכונים.
   השתמשו לפחות באחד מאלו.
   היו מוכנים להסביר איפה השתמשתם בכל אחד מהם, וכן לפרט את הסיבות להחלטה זו.

בנוסף, שימו לב להבדלים בין: ArrayList ו-Vector. וודאו שאתם מבינים איזה מהם מתאים יותר לצרכים שלכם בכל מצב.

11 – Application Progress: Logger

רישום (Logging) הוא תהליך של כתיבת הודעות לוג (Log messages) במהלך ריצת תוכנית למקום מרכזי.
רישום זה מאפשר לכם לדווח על שגיאות והערות (error and warning messages), וכן על הודעות המכילות מידע (כמו סטטיסטיקות ריצה), כך שההודעות יוכלו להיאסף לאחר מכן ולעבור ניתוח.
לוגרים (Loggers) הם נכס חשוב, במיוחד עבור תוכנית מבוססת תהליכונים. ראו מדריך כאן, על שימוש נכון בלוגר.

התוכנה שלכם צריכה ליישם לוגר בעיקר לצרכים אינפורמטיביים, ולעקוב אחר תהליך האפליקציה והתהליכונים השונים.
אנא שמרו את המידע מהלוגים לקובץ ע"מ להסביר את תהליך הריצה של האפליקציה לבודק.
על הלוגר שלכם להדפיס טקסט קריא ואינפורמטיבי שיקל על המעקב וההבנה.
ככל שהלוגר שלכם יהיה טוב יותר, כך יוכל הבודק להבין בקלות יותר את התוכנה שלכם.
וודאו שאתם משקיעים זמן בלהחליט איפה ומה להדפיס בתהליך הרישום שלכם.

12 – קומפילציה – Compiling the application: Another Neat Tool or ANT

האם אהבתם את makefile? מצוין! אנו נעשה את אותו הדבר עבור Java.
עליכם ליצור קובץ ANT לתוכנה שלכם ע"מ להשלים (אולי הכוונה לקמפל) אותה.
קובץ הANT שקול לmakefile. עליכם לקרוא את שני המדריכים הבאים:

- What is Ant
- Ant Tutorial

למידע נוסף ומקיף בנוגע לAnt, תוכלו לפנות לAnt Manual.
כשתסיימו לקרוא ולהבין מהו Ant, עליכם לבנות קובץ Ant לתוכנה שלכם.
המטרה (Target) ההכרחית היחידה לקובץ הבנייה – Ant שלכם היא:
"run": ant run –Darg()=InitialData.xml-Darg1=CustomerGroups.xml -Darg2=RentalRequests.xml -Darg3=Assets.xml –Darg4=AssetContents.xml"
בהרצת ANT עליכם לקמפל את התוכנה ולהריץ אותה בעזרת קבצי הקלט המתוארים למעלה.
סדר הקבצים **חשוב**.
זוהי דרישת חובה, התוכנה צריכה להתקמפל ללא שגיאות בעזרת Ant.
קימפול ע"י Eclipse בלבד אינו מתקבל.

13 – Deadlocks, Waiting, Liveness and Completion

Deadlocks:

עליכם לזהות מצבי Deadlock אפשריים ולמנוע מהם לקרות. היו מוכנים להסביר לבוחן אילו מצבי Deadlock זיהיתם, ואיך התמודדתם איתם.

Waiting:

עליכם להבין איפה מקרי Wait יכולים לקרות בתוכנה, ואיך פתרתם נושא זה.

Liveness:

נעילה וסנכרון (Locking and Syncronization) נדרשים בכדי שהתוכנה שלכם תעבוד כמו שצריך.
וודאו לא לנעול יותר מדי ולא לפגוע בחיות (Liveness) של התוכנה. זכרו, ככל שהתוכנה תבצע את המשימות מהר יותר – יותר טוב.

Completion:

כמו בכל תכנון רב-תהליכי, תשומת לב מיוחדת צריכה להינתן לכיבוי המערכת. כאשר כל הפקודות בוצעו ואין דבר ברשימת ההמתנה, עלינו לסגור את התוכנה.
תהליך זה צריך להתבצע בחינניות, לא בפתאומיות.

14 – Unit Tests

עליכם לכתוב בדיקות וממשק עיצוב לפי חוזה (Design by contract) למחלקת Warehouse, וכן בדיקות Unit Testing, ע״פ חוקי הTDD – Test Driven Development.
אנא שימו לב שההגשה לא צריכה לכלול יישום מלא של המחלקה warehouse, אלא אחד ריק.
הגשת חלק זה היא עד ה12.12.2014, 13:30.
הקבצים הבאים נדרשים:

1. Warehouse Interface.
2. מימוש ריק של הממשק warehouse.
3. Unit test עבור מחלקת warehouse.

אם ברצונכם להוסיף פונקציות עזר חדשות ע״מ לבצע את הבדיקות, עליכם **לא** לשנות את הממשק **ולא** לשנות את היישום. מה שתצטרכו לעשות הוא לרשת את מחלקת Warehouse (WarehouseTesting extends Warehouse למשל) ובמחלקה WarehouseTesting תוכלו להוסיף שיטות.
לאחר מכן תיצרו מחלקת Unit Test שתבדוק את WarehouseTesting ולא את Warehouse.
מאחר ו-WarehouseTesting מכילה את השיטות הנוספות, הכל יכול להיבדק באופן נכון. מאחר וWarehouseTesting מכילה את מה שWarehouse מכילה, בסופו של דבר, ע״י בדיקת WarehouseTesting אתם גם בודקים את Warehouse.
אינכם צריכים לבדוק סנכרון (Synchronization) במחלקה שלכם.

15 – תיעוד וסטייל – Documentation and coding style

ישנם שיטות עבודה רבות אשר **חובה** להשתמש בהן כאשר כותבים כל קטע קוד.

- עקבו אחר Java Programming Style Guidelines. חלק מהציון יהיה בדיקה האם עקבתם אחר קווים מנחים אלה או לא. חשוב להבין שתכנות היא עבודה מוכוונת-צוות. אם הקוד שלכם קשה לקריאה, אזי אתם נכשלים כחלק יעיל מכל צוות שהוא, באקדמיה או בעבודה. לכן, עליכם לעקוב אחר כללים אלה, שיהפכו את הקוד שלכם לקל לקריאה והבנה ע״י העמיתים שלכם והבודקים.
- אל תפחדו להשתמש בשמות ארוכים למשתנים ושיטות המשפרים את ההבנה של ייעודם.
- הימנעו מחזרות בקוד. במקרה שאתם שמים לב שאתם כותבים את אותו בלוק של קוד בשני מקומות שונים, משמעות הדבר היא שעליכם לשים את קטע הקוד בפונקציה פרטית, כך ששני המקומות יוכלו להשתמש בו.
- תיעוד מלא של המחלקות השונות והשיטות הפרטיות והמוגנות שלהם, ע״פ פורמט Javadoc. אם תרצו, תוכלו גם לתעד את השיטות המקומיות (Local) שלכם, אולם אין זה חובה.
- הוסיפו הערות לבלוקים של קוד, אשר בלעדיהם הבנתם תהיה קשה.
- אסור שהקבצים שלכם יכילו קוד שהוצא כהערה. זה זבל. כאשר אתם מסיימים לכתוב, ודאו כי הקוד נקי.
- פונקציות נחשבות ארוכות החל מ30+ שורות. זה אומר שהפונקציה שלכם מבצעת יותר מדיי משימות, כלומר אתם יכולים להזיז משימות אלה למקום משלהם בפונקציה פרטית, ולהשתמש בהם לפי הצורך. זהו צעד חשוב לקראת קוד קריא. עליכם לבצע זאת **גם אם אתם משתמשים בקוד פעם אחת בלבד**.
- מספרי מזל הם גרועים. למה? אסור שבתוכנה שלכם יהיו מספרים בקוד (בעלי משמעות נסתרת).

- Getters, Setters הם גרועים. הם **שוברים** את הencapsulation. אם האובייקט שלכם אינו אובייקט Data Structure, עליכם להימנע משימוש בגטרים וסטרים בכל מחיר! ניתן לקרוא למה כאן, כאן, כאן, כאן, וכאן.
- אל תשלחו אוסף (Collection) של פריטים לבנאי. הם יכולים לחשוף יישומים פנימיים.
  במקום זאת, צרו מחלקה עבור האובייקט המוסיפה פריט אחד אליו. לדוגמא:
  צרו שיטות addClerk או addCustomerGroup במחלקת Manager, וכך תוכלו להוסיף פקידים או קבוצת לקוחות לפי דרישה.
- אל תחשפו את היישום הפנימי של האובייקטים שלכם.
  לדוגמא:
  ברצונכם להוסיף אובייקט RentalRequest לאובייקט-אוסף בתוך אובייקט Management.
  אל תחשפו (למחלקות אחרות) את האוסף הפנימי (ע"י שימוש בgetter) ואז הוספת RentalRequest אליו, אלא במקום זאת, שלחו את RentalRequest לאובייקט Management, אשר מיישם שיטה כמו: addRentalRequest(RentalRequest rentalRequest) המוסיפה את בקשת השכירות לאוסף. זה צריך להיעשות לכל הבקשות גישה לאובייקטים פנימיים.
  אל תחזירו data פנימי מאובייקט, אלא יישמו שיטה עבור אותו אובייקט אשר מממשת את רצונכם.
  דוגמא נוספת:
  ברצונכם לחשב את המרחק בין שני מקומות? אל תקבלו את הקואורדינטות מכל אובייקט Location ועשו את החישוב בעצמכם, אלא במקום זאת יישמו שיטה המחשבת את המרחק בתוך האובייקט Location, אשר מקבלת Location אחר ומחשבת את המרחק באופן פנימי, ומחזירה את הערך הרצוי.
  הערה: מאחר ואתם ניגשים לאותה מחלקה, אתם יכולים לראות שדות פרטיים באופן ישיר.

על שיטת היישום שלכם לעקוב אחר הוראות אלו כדי לשמור על הקוד מסודר, נקי וקל לקריאה.

16 – הוראות הגשה – Submission Instructions

- הגשה תבוצע בזוגות בלבד. אם אין לכם בן זוג, מצאו אחד. עליכם לקבל אישור מפורש מצוות הקורס להגיש את העבודה לבד. אסור להגיש בקבוצות גדולות מזוג.
- עליכם להגיש קובץ tar.gz אחד המכיל את כל הקוד שלכם. על הקובץ להיקרא "assignment3.tar.gz".
  הערה: אנו דורשים שימוש בtar.gz. כל סיומת אחרת תפגע בציון העבודה.
- הקובץ חייב לכלול קובץ ANT עם קונפיגורציה מתאימה לתוכנית שלכם. הבודק יבצע חילוץ של הקובץ, יבנה את האפליקציה בעזרת קובץ הANT ויפעיל אותה.
- בקשות להארכת זמן....

17 – ציון

- למרות שאתם חופשיים לעבוד כרצונכם, העבודות יבדקו על מחשבי המעבדה של המחלקה, אז ודאו לבדוק את התוכנית שלכם לעומק עליהם לפני ההגשה הסופית.
  כל תירוץ שהוא לא יזכה בנקודות.

הציון יורכב מסעיפים רבים, הכוללים בינהם:

- עיצוב ויישום ה TDD שלכם.

- עיצוב ויישום התוכנה.
- עבור כל סעיף ב- 10 אשר יישמתם, והסיבות להחלטתכם.
- בדיקות אוטומטיות יבוצעו על התוכנית שלכם. עליה לבצע זאת בהצלחה ובזמן הגיוני.
- Liveness ו-Deadlock, סיבות לDeadlock והיכן בתוכנה שלכם Deadlock יכול היה לקרות, ללא הפתרונות שלכם לבעיה.
- Synchronization – מהו, איפה השתמשתם בו וסיבה מניחה את הדעת להחלטתכם.
- בדיקה האם היישום שלכם עומד בדרישות סעיף 15.

18 – שאלות ועזרה:

- פורום העבודה.
- אנו לא נענה למיילים הנוגעים לעבודה. אנא הימנעו משליחתם.
- שאלות נפוצות יעודכנו באתר עבודה 3. חובה לקרוא אותן באופן יום-יומי לשינויים. כל השינויים שנכתבים שם הם מחייבים.
- אנו נבקר במעבדות ונעזור לסטודנטים בזמנים שיפורסמו באתר.

בהצלחה!