תכנות מונחה עצמים מתקדם
עבודת הגשה מס 1

מבוא

זהו התרגיל הראשון בקורס ומהווה בסיס לסדרת תרגילים שיינתנו בהמשך הקורס, בנושא ניהול משלוחים. מטרת התרגיל היא כתיבת מערכת המדמה פעולה של חברת משלוחים, המיוצגת ע"י חבילות, סניפים, מרכז מיון ורכבי הפצה. המערכת מנהלת ומתעדת כל שלב בתנועת חבילות משלב היצירה עד שלב המסירה ללקוח.

נא לקרוא את כל המסמך לפני תחילת העבודה!

דגשים להגשה

- ניתן להגיש עבודה זו בזוגות – רק אחד מהסטודנטים יגיש את העבודה במודל. בתיעוד בכל קובץ יש לציין שם ות.ז. של מגיש\ים, בתוך תיעוד ה javadoc.
- חלק מניקוד העבודה מתבצע ע"י בדיקות אוטומטיות ולכן חשוב מאוד להגדיר את כל המחלקות, המטודות והשדות בדיוק כפי שצוינו במסמך.
- על העבודה לעבוד בצורה מדויקת עם קובץ ה program שתקבלו בהמשך.
- לכל שאלה לפנות למתרגל האחראי על עבודה ראשונה – מיכאל פינקלשטיין במייל finkm@ac.sce.ac.il. נושא ההודעה חייב להיות "aoop21_hw1".
- חובה לתעד כל קובץ, מחלקה ופונקציה ע"י javaDoc - ניתן להיעזר בתיעוד באתר oracle או בקבצים הרלוונטיים במודל.

דגשים לעבודה זו

- על כל העבודה להיות פרויקט יחיד המחולק לpackages לפי המטלות.
- על כל השדות בכל המחלקות להיות פרטיים בלבד. גישה אליהם תתבצע בעזרת get/set .
- לכתוב בכל מחלקה את הבנאים הדרושים כדי שתעבוד המחלקה הראשית Program .
- על כל הקבועים להיות תחת שדות final. שימו לב, כלל ההתייחסויות במהלך הקוד לערכים ספציפיים ייעשו באמצעות גישה לקבועים הנ"ל.
- העבודה מכילה נושאים המיועדים ללמידה עצמאית.
- **בכל מחלקה חובה לממש בנוסף למתודות המתוארות:**
  toString(), equals(), getters&setters.
- **חובה להקפיד על המבנה הנתון, כל הנחיה להורשה/ממשק/ enum היא מחייבת**
- **ניתן להוסיף מתודות ומשתנים לבחירתכם לפי הצורך.**

<u>תיאור המערכת</u>

המערכת מדמה עבודה של חברת שליחויות. לחברה מספר סניפים, מרכז מיון ומשרד ראשי. לכל סניף משוייכים רכבי הפצה – לסניפים רכבים מסוג Van, ולמרכז מיון רכבים מסוג "משאית סטנדרטית" ורכב אחד נוסף מסוג "משאית לא סטנדרטית" לטובת ביצוע הובלות מטען חריג. כאשר קיים משלוח שיש לבצע, מערכת יוצרת "חבילה" ומשייכת אותה לסניף המתאים. הטיפול בחבילות מתבצע באופן הבא:

<u>עבור חבילות קטנות וסטנדרטיות:</u>

המערכת (משרד ראשי) משייכת את החבילה לסניף מקומי מתאים לפי כתובתו של השולח. הסניף המקומי שולח את אחד הרכבים שלו לאסוף את החבילה מכתובתו של השולח ומביא אותה לסניף. משאית סטנדרטית ממרכז מיון מגיעה לסניף המקומי ואוספת את החבילה למרכז מיון. החבילה מאוחסנת במרכז מיון עד שמשאית נוספת של מרכז מיון לוקחת את החבילה לסניף היעד (לפי כתובת המקבל). סניף היעד שולח רכב הפצה לכתובתו של הלקוח המקבל ומוסר לו את החבילה.

<u>עבור חבילות לא סטנדרטיות:</u>

המערכת (משרד ראשי) משייכת את החבילה למרכז מיון. אם גודל החבילה מתאים למשאית הלא סטנדרטית של מרכז מיון, המשאית מגיעה לשולח, אוספת את החבילה ומעבירה אותה ישירות למקבל.

<u>עבור כל סוגי חבילות</u> בכל העברה של חבילה בין לקוח, רכבים, סניפים, ומרכז מיון משתנה סטטוס המשלוח, ובנוסף נוספת רשומה להיסטוריית העברות של חבילה (tracking).

<u>פעולה:</u>

בשלבים הבאים המערכת תפעל בסביבה מרובת תהליכונים ועם ממשק גרפי. בשלב זה המערכת מבצעת סימולציה של עבודה לפי שעון, לצורך כך המערכת משתמשת במשתנה "שעון", אותו היא מקדמת בכל איטרציה, כאשר כל איטרציה מסמלצת פעימת זמן ("טיק" של שעון, יחידת זמן) אחת. בכל פעימה הרכבים, הסניפים ומרכז מיון יבצעו יחידת עבודה אחת:

- <u>רכבים:</u> רכב שמבצע הובלה כלשהי, יצמצם את הזמן הנותר להגעה ליעד. כאשר הזמן הזה יתאפס, יבצעו את הפעולה שלשמה הגיעו – איסוף או מסירה של חבילה.

- <u>סניפים מקומיים:</u> יבדקו אם יש חבילות שממתינות לאיסוף או לחלוקה ואם ישנו רכב פנוי, ישלחו אותו לבצע את המשלוח (כל נסיעה של רכב מסוג Van) היא הובלה של חבילה אחת בלבד.

- <u>מרכז מיון:</u> אם ישנן משאיות סטנדרטיות פנויות, מרכז מיון שולח אותן לסניפים המקומיים (לפי הסדר של מספרי סניפים), לפני שהמשאית יוצאת לסניף כלשהו, מרכז מיון מעביר אליה את כל החבילות שממתינות להעברה לסניף זה. המשאית תעביר את החבילות לסניף המקומי ותאסוף מאותו סניף את החבילות שצריכות להגיע למרכז מיון. בנוסף, מרכז מיון יבדוק אם המשאית הלא סטנדרטית פנויה, במידה וכן, יבדוק אם יש חבילות לא סטנדרטיות שממתינות לאיסוף, וישלח את המשאית לאסוף אותן ולהעביר אותן ישירות ללקוח המקבל. <u>שימו לב:</u> בכל יחידת עבודה שליחת משאיות ממשיכה מאותו מקום (מספר סניף) בו הסתיימה ביחידת עבודה קודמת, לדוגמא אם בפעימה קודמת משאית אחרונה נשלחה לסניף 3, בפעימה הנוכחית המשאית הראשונה תשלח לסניף מספר 4. לאחר שנשלחה משאית לסניף האחרון, המשאית הבאה תשלח לסניף הראשון.

- <u>משרד ראשי:</u> מפעיל את כל המערכת. יוצר אובייקטים של חבילות (בכך מדמה הזמנה של משלוח ע"י לקוח) מסוגים שונים וכתובות שונות ומשייך אותן לסניפים המתאימים, מנהל את שעון המערכת ומפעיל את כל שאר המחלקות שאמורות לבצע "יחידות עבודה". שומר את כל החבילות שנוצרו במערכת ומדפיס דו"ח בסיום העבודה.

מבנה המחלקות של המערכת מפורט בהמשך. ***שימו לב, יש להקפיד על כללי OOP ולדאוג לבנות עץ הורשה וממשקים מתאימים, כך שהתכנות יהיה יעיל וללא חזרות על קוד.***

1. מחלקות עזר, ממשקים, package: components : enum

1.1. Tracking– המחלקה מייצגת רשומה בהיסטוריית העברות של חבילה. כל חבילה מכילה אוסף של רשומות מסוג זה, בכל שינוי סטטוס (ומיקום) של חבילה מתווספת רשומה חדשה לאוסף. כל רשומה כוללת זמן יצירת הרשומה (לפי השעון של התוכנית, הסבר על השעון בהמשך), נקודה בה החבילה נמצאת וסטטוס החבילה.

1.1.1. **Fields**:

- **time**:int

מספר שלם, ערך של שעון המערכת ברגע יצירת הרשומה.

- **node**: Node

מיקום החבילה – לקוח/סניף/מרכז מיון/רכב הובלה. כאשר החבילה נמצאת אצל הלקוח (השולח או המקבל), ערך השדה הזה הוא null.

- **status**: Status

סטטוס החבילה ברגע יצירת הרשומה (פרטים בהמשך).

1.1.2. **Constructors**:

- public **Tracking**(int time, Node node, Status status)

1.1.3. **Methods**

- לפי הצורך

1.2. Node – מייצג מיקום של חבילה, יכול להתייחס לסניפים ולמשאיות (כל הנקודות בהן החבילה יכולה להימצא בשלבים השונים של העברתה). יש להחליט לבד מה הוא הטיפוס המתאים לכך.

1.2.1. **Methods:**

- **collectPackage**(Package p):void

מתודה אשר מטפלת באיסוף / קבלת חבילה ע"י המחלקה המממשת.

- **deliverPackage**(Package p):void

מתודה אשר מטפלת במסירת החבילה לגורם הבא בשרשרת העברות.

- void **work**(): void

מתודה המבצעת יחידת עבודה.

1.3. Status – enum שכולל רשימת סטטוסים התואמים לשלבי המשלוח. השלבים הם:

יצרית חבילה (**CREATION**) – סטטוס התחלתי של כל חבילה שנוצרת.

איסוף (**COLLECTION**) – סטטוס זה חבילה מקבלת כאשר רכב הובלה נשלח לאסוף אותה מכתובת השולח.

אחסון בסניף (**BRANCH_STORAGE**) – החבילה שנאספה מלקוח הגיעה לסניף המקומי של השולח.

העברה למרכז מיון (**HUB_TRANSPORT**) – החבילה בדרך מהסניף המקומי למרכז מיון.

אחסון במרכז מיון (**HUB_STORAGE**) – החבילה הגיעה למרכז מיון וממתינה להעברה לסניף היעד.

העברה לסניף היעד (**BRANCH_TRANSPORT**) – החבילה בדרך ממרכז מיון לסניף היעד.

נכונות למסירה (**DELIVERY**) – החבילה הגיעה לסניף היעד ומוכנה למסירה ללקוח הסופי.

חלוקה (**DISTRIBUTION**) – החבילה בדרך מסניף היעד ללקוח הסופי.

נמסר (**DELIVERED**) – החבילה נמסרה ללקוח הסופי.

כלומר, הרשימה אמורה להכיל את הערכים הבאים: **CREATION, COLLECTION, BRANCH_STORAGE, HUB_TRANSPORT, HUB_STORAGE, BRANCH_TRANSPORT, DELIVERY, DISTRIBUTION, DELIVERED**

1.4. Priority – enum שכולל רשימה עדיפויות (דחיפות החבילה):

LOW, STANDARD, HIGHT

1.5. Address – כתובת (של השולח או של המקבל). הכתובת מורכבת משני מספרים שלמים, מופרדים ע"י מקו. המספר הראשון (zip) קובע את הסניף אליו שייכת הכתובת, המספר השני (street) את מספר הרחוב.

1.5.1. **Fields:**

- **zip** : int

מספר סניף אליו שייכת הכתובת. לדוגמא, הכתובת 3-124459 שייכת לסניף מספר 3.

- **street** : int

מספר רחוב, יכול לקבל ערכים בני 6 ספרות בלבד.

1.5.2. **Constructors:**
public **Address** (int zip, int street)

<u>הערה:</u> כתובת מספרית משמשת בשלבים הבאים לחישובים, כגון חישוב זמן נסיעה.

2. רכיבים: package: components

2.1. Package – טיפוס כללי מייצג חבילות.

2.1.1 **Fields**:

- **packageID**: int מספר רץ
מספר מזהה של חבילה. מספור החבילות מתחיל מ-1000.
- **priority**: enum < Priority >
עדיפות (פירוט בסעיף 1.4)
- **status**: enum < Status>
סטטוס נוכחי (מפורט בסעיף 1.3)
- **senderAddress**: Address
כתובת השולח (מפורט בסעיף 1.5)
- **destinationAddress**: Address
כתובת המקבל
- **tracking**: ArrayList <Tracking>

אוסף רשומות עם היסטוריית העברות (אובייקטים של מחלקת Tracking). ביצירת החבילה מאותחל כאוסף ריק ומיד מתווסף אליו האובייקט הראשון המתייחס ליצירת החבילה. בכל פעולת העברה שלה חבילה מתווסף לאוסף זה ואובייקט נוסף. בסוף הרצת התוכנית עבור כל חבילה מודפסת היסטוריית העברות שלה על סמך אוסף זה (ראו דוגמת הדפסה).

2.1.2 **Constructors**:

- **Package** (Priority priority, Address senderAddress, Address destinationAdress)

בנאי שמקבל כארגומנטים עדיפות, כתובות השולח ומקבל חבילה.

2.1.3 **Methods**:

- void **addTracking** (**Node** node, **Status** status)

מקבלת אובייקט מטיפוס של **Node** (רכב או סניף) ואובייקט **Status**, יוצרת ומוסיפה אובייקט של מחלקת **Tracking** לאוסף **tracking** במחלקה.

- void **printTracking** ()

מדפיסה את הרשומות השמורות באוסף tracking, כל אובייקט בשורה נפרדת

2.2. SmallPackage – מייצגת חבילות קטנות.

2.2.1 **Fields**:

- **acknowledge**: boolean

שדה זה מקבל ערך **true** אם החבילה דורשת שליחת אישור מסירה לאחר אספקה למקבל.

2.2.2 Constructors:

- **SmallPackage**(Priority priority, Address senderAddress, Address destinationAdress, boolean acknowledge)

בנאי שמקבל כארגומנטים עדיפות וכתובות שולח ומקבל החבילה, ואם יש צורך באישור מסירה.

2.2.3 **Methods**:

- לפי הצורך

2.3. StandardPackage – מחלקה מייצגת חבילות במשקל משתנה מעל קילוגרם אחד.

2.3.1 **Fields**:

- weight: double

משקל החבילה

2.3.2 **Constructors**:

- **StandardPackage** (Priority priority, Address senderAddress, Address destinationAdress,double weight)

בנאי שמקבל כארגומנטים עדיפות, כתובות השולח והמקבל, משקל החבילה.

2.3.3 **Methods**:

- לפי הצורך

2.4. NonStandardPackage– מייצגת את חבילות בגודל לא סטנדרטי.

2.4.1 **Fields**:

- **width**: int
- **length**: int
- **height**: int

2.4.2 **Constructors**:

- **NonStandardPackage**(Priority priority, Address senderAddress,Address destinationAdress,int width, int length, int height)

בנאי שמקבל כארגומנטים עדיפות, כתובות השולח והמקבל, ומידות של חבילה.

2.4.3 **Methods**:

- לפי הצורך

2.5. Truck– מחלקה המייצגת את הרכבים להובלת חבילות.

2.5.1 **Fields**:

- **truckID**: int מספר רץ
  מספר סידורי של רכב, מספור הרכבים מתחיל מ-2000.
- **licensePlate**: String
  מספר מזהה של הרכב (לוחית זיהוי, ראו דוגמת הדפסה)
- **truckModel**: String
  מודל של רכב (ראו דוגמת הדפסה).
- **available**: boolean
  זמינות של רכב (זמין להובלה, השדה מקבל ערך **false** כאשר הרכב יוצא לאיסוף/מסירה של חבילות)
- **timeLeft**: int
  זמן שנותר עד סיום ההובלה (עד ההגעה ליעד)
- **packages**: ArrayList<Package>
  רשימת חבילות להובלה שנמצאות ברכב

2.5.2 **Constructors**:

- public **Truck**()

בנאי ברירת מחדל <u>רנדומלי</u> שמייצר אובייקט עם לוחית זיהוי ומודל של רכב בצורה אקראית.
מודל מורכב מאות M וספרה בין 0 ל-4 (ראו דוגמת הדפסה).
לוחית זיהוי מורכבת משלושה מספרים מופרדים ע"י מקו, לפי התבנית xxx-xx-xxx (ראו דוגמת הדפסה).

- public **Truck**(String licensePlate, String truckModel)

בנאי שמקבל כארגומנטים מספר לוחית זיהוי ומודל של הרכב ומייצר אובייקט.

2.5.3 **Methods**:

- לפי הצורך

2.6. Van – רכב שמבצע איסוף חבילה מכתובת השולח לסניף המקומי ומספק את החבילה מסניף היעד לכתובת המקבל (חבילה אחת בלבד בכל נסיעה). למחלקה אין שדות.

2.6.2 **Constructors**:

- public **Van**()

בנאי ברירת מחדל שמייצר אובייקט אקראי לפי אותם כללים כמו במחלקת האב.

- public **Van**(String licensePlate, String truckModel)

מייצר אובייקט עם פרמטרים נתונים.

2.6.3 **Methods**:

- **work()**

מבצעת יחידת עבודה (בכל פעימה) לפי הדרישות הבאות (תוך כדי שימוש בפונקציות נוספות):

רכב פנוי (available) לא מבצע דבר.

רכב שנמצא במהלך נסיעה מצמצם את הזמן שנותר לסיום הנסיעה (timeLeft) ב-1. אם לאחר הצמצום ערך הזמן שווה לאפס, אזי הנסיעה הסתיימה ורכב ביצע את המשימה לשמה נשלח. אם מטרת הנסיעה הייתה איסוף חבילה מלקוח שולח (COLLECTION), החבילה בשלב זה תעבור מהרכב לסניף, סטטוס החבילה יתעדכן, לרשימת **tracking** של החבילה יתווסף רישום מתאים, ותודפס הודעה על כך שהרכב אסף את החבילה והגיע חזרה לסניף (ראו דוגמת הדפסה). בנוסף, הרכב ישנה את מצבו לפנוי. אם מטרת הנסיעה הייתה מסירת החבילה ללקוח (DISTRIBUTION), החבילה תוסר מרשימת החבילות ברכב, סטטוס החבילה והיסטוריית העברות יתעדכנו בהתאם ותודפס הודעה על כך שהחבילה נמסרה ללקוח. במידה ומדובר על חבילה קטנה עם אופציית שליחת אישור מסירה מופעלת, תודפס הודעה על שליחת אישור מסירה.

2.7. StandardTruck– רכב להעברת חבילות ממרכז מיון לסניפים וחזרה. כל הרכבים מהסוג הזה נמצאים במרכז מיון.

2.7.1 **Fields**:

- **maxWeight**: int

משקל מקסימלי שרכב יכול להוביל.

- **destination**: Branch

סניף יעד / מרכז מיון

2.7.2 **Constructors**:

- public **StandardTruck** ()

בנאי ברירת מחדל שמייצר אובייקט עם מספר לוחית זיהוי ומודל של רכב בצורה אקראית.

- public **StandardTruck**(String licensePlate,String truckModel,int maxWeight)

בנאי שמקבל כארגומנטים: מספר לוחית זיהוי, מודל של הרכב ומשקל מקסימלי.

2.7.3 **Methods**:

- **work()**

מבצעת יחידת עבודה (בכל פעימה) לפי הדרישות הבאות (תוך כדי שימוש בפונקציות נוספות):

רכב שנמצא במהלך נסיעה מצמצם את הזמן שנותר לסיום הנסיעה (timeLeft) ב-1. אם לאחר הצמצום ערך הזמן שווה לאפס, אז הנסיעה הסתיימה, יש להדפיס הודעה על כך שרכב הגיע ליעד (ראו דוגמת הדפסה) ועל הרכב להעביר את כל החבילות שבתוכו לנקודה אליה הגיע (סניף מקומי או מרכז מיון), תוך כדי עדכון סטטוס והיסטוריה של חבילות. במידה והנסיעה הסתיימה במרכז מיון, אז על הרכב לעבור למצב "פנוי".

אחרת, הנסיעה הסתיימה בסניף המקומי, על הרכב להעמיס את החבילות מסניף זה ולקחת אותן למרכז מיון. גם במקרה הזה יש לדאוג לעדכון הסטטוס וההיסטוריה של חבילות. בנוסף, יוגרל זמן נסיעה חדש (ערך בין 1 ל-6) ויעודכן בשדה הרלוונטי. תודפס הודעה על כך שהרכב יצא חזרה מסניף מקומי למרכז מיון.

<u>הערה</u>: כאשר מתבצע איסוף חבילות ע"י רכב ממחלקה זאת, יש לוודא כי משקל החבילות לא עולה על המשקל המרבי שהרכב יכול להוביל. לשם כך יש לסכום את משקל כל החבילות, כאשר משקלן של חבילות קטנות יחושב כקילוגרם אחד.

<u>הערה</u>: רכב פנוי (available) לא מבצע דבר.

2.8. NonStandardTruck– רכב להעברת חבילות בגודל לא סטנדרטי (מטען חריג). כל הרכבים מסוג הזה נמצאים במרכז מיון.

2.8.1 **Fields**:

- **width: int**
- **length: int**
- **height: int**

2.8.2 **Constructors**:

- public **NonStandardTruck** ()

בנאי ברירת מחדל שמייצר אובייקט עם מספר מזהה ומודל של רכב בצורה אקראית.

- public **NonStandardTruck**(String licensePlate, String truckModel, int length, int width, int height)

בנאי שמקבל ארגומנטים: מספר לוחית זיהוי, מודל של הרכב ואורך/רוחב/גובה מקסימלי של מטען שהרכב יכול להוביל.

2.8.3 **Methods**:

- **work()**

- מבצעת יחידת עבודה (בכל פעימה) לפי הדרישות הבאות:
- רכב פנוי (available) לא מבצע דבר.
- רכב שנמצא במהלך נסיעה מצמצם את הזמן שנותר לסיום הנסיעה (timeLeft) ב-1. אם לאחר הצמצום ערך הזמן שווה לאפס, אז הנסיעה הסתיימה ורכב הגיע ליעד. אם מטרת הנסיעה הייתה איסוף חבילה מלקוח שולח (COLLECTION), הרכב יעבור לביצוע אספקה (נסיעה חדשה – הסבר בהמשך), סטטוס החבילה יתעדכן, לרשימת **tracking** של החבילה יתווסף רישום מתאים, ותודפס הודעה על כך שהרכב אסף את החבילה. אם מטרת הנסיעה הייתה מסירת החבילה ללקוח (DISTRIBUTION), החבילה תוסר מרשימת החבילות ברכב, סטטוס החבילה והיסטוריית העברות יתעדכנו בהתאם ותודפס הודעה על כך שהחבילה נמסרה ללקוח, והרכב עובר למצב "פנוי".
- אם הרכב עובר למצב אספקה, עליו לחשב זמן נסיעה חדש. זמן זה מחושב כך:
ערך מוחלט של הפרש בין כתובת השולח לכתובת המקבל מחולק בעשר, ולשארית מוסיפים אחד. (מתייחסים רק למספר רחוב בכתובות). זמן זה מתעדכן בשדה **timeLeft**.

2.9. Branch– מתארת סניף מקומי. שומרת רשימת חבילות המאוחסנות בסניף או מיועדות לאיסוף מכתובת השולח לסניף זה, ורשימת רכבים שאוספים את החבילות מלקוחות השולחים ומספקים את החבילות ללקוחות המקבלים.

2.9.1. **Fields**:

- **branchId**: int מספר רץ
- **branchName:** String
- **listTrucks**: ArrayList <Truck>

אוסף רכבים ששייכים לסניף זה

- **listPackages**: ArrayList <Package>

אוסף חבילות שנמצאות בסניף וחבילות שיש לאסוף משולחים ע"י הסניף הזה.

2.9.2. **Constructors**:

- **Branch**()

בנאי ברירת מחדל, מחשב את המספר הסידורי של הסניף ויוצר את שם הסניף, את שני השדות הנותרים מאתחל לאוספים ריקים.

- **Branch**(String branchName)

בנאי שמקבל שם סניף, מחשב את המספר הסידורי של הסניף, את שני השדות הנותרים מאתחל לאוספים ריקים.

2.9.3. **Methods**

- **work**()

- יחידת עבודה המתבצעת ע"י סניף בכל פעימת שעון המערכת (מותר שימוש בפונקציות נוסופות).
- עבור כל חבילה שנמצאת בסניף, אם היא בסטטוס המתנה לאיסוף מלקוח, מתבצע ניסיון לבצע איסוף – אם יש רכב פנוי, הוא יוצא לאסוף את החבילה. זמן נסיעה מחושב כך: מספר רחוב של השולח מחולק בעשר, לשארית שהתקבלה מוסיפים אחד. הערך שמתקבל מתעדכן ברכב בשדה **timeLeft** ומצבו של הרכב משתנה ל"לא פנוי".
- כנ"ל עבור כל חבילה שממתינה לחלוקה, אם ישנו רכב פנוי, הוא נשלח למסור את החבילה. זמן מחושב לפי אותה נוסחה, אבל לפי כתובת המקבל.
- בשני המקרים יש לדאוג לעדכון סטטוס והיסטוריה של החבילה והדפסות מתאימות.

2.10. Hub – מתארת סניף ראשי.

2.10.1. **Fields**:

- **branches**: ArrayList <Branch>

אוסף אובייקטים של כל הסניפים המקומיים.

2.10.2. **Constructors**:

- Hub()

יוצר אובייקט עם השם "HUB"

2.10.3. **Methods**

- **work()**

יחידת עבודה שמתבצעת בפעימת שעון אחת (מותר שימוש בפונקציות נוספות).
עבור כל משאית סטנדרטית של מרכז מיון, אם המשאית פנויה, היא נשלחת לסניף מקומי כלשהו (על מרכז מיון לדאוג שהנסיעות לסניפים יצאו לפי הסדר של מספרי הסניף. אם משאית אחרונה יצאה לסניף 2, אז המשאית הנוכחית תצא לסניף 3). לצורך הנסיעה המשאית תטען אליה את כל החבילות שממתינות להעברה לסניף שאליו היא נוסעת, כל עוד משקל החבילות יהיה קטן מהמשקל המירבי אותו היא יכולה לשאת. משקל החבילות הסטנדרטיות מופיע בכל חבילה, ומשקל חבילה קטנה הוא קילוגרם אחד. סטטוס החבילות וההיסטוריה שלהן מתעדכנים בהתאם, מודפסת הודעה על כך שהמשאית יוצאת לסניף זה, זמן הנסיעה הוא ערך מוגרל בין 1 ל-10.
אם המשאית הלא סטנדרטית פנויה, ייבדק האם קיימת במרכז מיון חבילה לא סטנדרטית שממתינה לאיסוף ומידותיה מתאימות למשאית. במידה וכן, המשאית תשלח לאסוף את החבילה. איסוף החבילה הלא סטנדרטית מלקוח מתבצע בדיוק לפי אותם כללים כמו שאר החבילות, רק ע"י משאית לא סטנדרטית.

<u>הערה:</u> חבילות שנשארו במרכז מיון עקב הגבלת משקל במשאית, יטופלו בסבבים הבאים, חבילות לא סטנדרטיות שלא נכנסו למשאית לא סטנדרטית בגלל מגבלות גודל, ישארו במרכז מיון עד סוף ההרצה.

MainOffice– אובייקט של מחלקה זאת מנהל את כל המערכת, מפעיל שעון, את הסניפים והרכבים, יוצר את החבילות (מדמה לקוחות) ומעביר אותן לסניפים המתאימים.

2.10.4. **Fields**:

- **clock**: int

מאותחל לאפס, בכל פעימה מקודם באחד. מייצג את כמות הפעימות שעברו מרגע הפעלת המערכת.

- **hub**: Hub

אובייקט של מרכז מיון, המכיל את כל הסניפים הפעילים במשחק.

- **packages**: ArrayList < Package>

אוסף של כל החבילות שקיימות במערכת (כולל כאלה שסופקו כבר ללקוח).

2.10.5. **Constructors**:

- **MainOffice**(int branches, int trucksForBranch)

בנאי שמקבל את כמות הסניפים שיהיו במשחק וכמות רכבים לכל סניף. הבנאי יוצר מרכז מיון (Hub) ומוסיף לו משאיות סטנדרטיות בכמות שבפרמטר trucksForBranch. בנוסף מוסיף למרכז מיון משאית לא סטנדרטית אחת. לאחר מכן יוצר סניפים (Branch) בכמות שמופיעה בפרמטר branches ולכל סניף כזה מוסיף משאיות מסוג Van בכמות התואמת את הפרמטר trucksForBranch.

2.10.6. **Methods**

- void **play**(int playTime)

פונקציה מקבלת כארגומנט את מספר הפעימות אותן תבצע המערכת ומפעילה את הפעימות (tick) כמות פעמים זו.

- void **printReport**()

מדפיסה דו"ח מעקב עבור כל החבילות שקיימות במערכת: עבור כל חבילה מדפיסה את כל תכולת האוסף **tracking** של החבילה (ראו דוגמת הרצה).

- String **clockString**()

מדפיסה את הערך של השעון בפורמט **MM:SS**

- void **tick**()

פעימה. בכל הפעלת פונקציה זאת (בכל פעימה) מתבצעות הפעולות הבאות:
השעון מודפס ומקודם באחת.
כל הסניפים, מרכז מיון ורכבים מבצעים יחידת עבודה אחת (פירוט פונקציה **work** של ממשק **Node**).
כל 5 פעימות נוצרת חבילה חדשה רנדומלית (פירוט בהמשך).
לאחר הפעימה האחרונה מודפסת הודעה על סיום העבודה ("STOP") ולאחר מכן מודפס דו"ח של היסטוריית העברות עבור כל החבילות שנוצרו במהלך הריצה (ראו דוגמת הדפסה).

- void **addPackage**()

מופעלת כל 5 פעימות. יוצרת חבילות באופן הבא:
מגרילה את סוג החבילה (קטנה / סטנדרטית / לא סטנדרטית), את העדיפות, את הכתובות של השולח והמקבל (כך שיתאימו לדרישות במחלקת **Address**), כמו כן בהתאם לסוג החבילה שהוגרל מגרילה את הנתונים הנוספים: עבור חבילה קטנה מוגרל הערך של **acknowledge**, עבור חבילה סטנדרטית מוגרל המשקל – מספר ממשי בין 1 ל-10.
ועבור חבילה לא סטנדרטית מגרילים הגובה (מספר שלם עד 400), הרוחב (מספר שלם עד 500) והאורך (מספר שלם עד 1000).
לאחר הגרלת הנתונים נוצרת חבילה מתאימה ומשויכת לסניף המתאים: חבילות קטנות או סטנדרטיות מועברות לסניף מקומי (שמספרו הוא ערך ה-zip בכתובת השולח), וחבילות לא סטנדרטיות מועברות למרכז מיון.

3. package: program

יש ליצור את הרכיב עם הקוד הנתון ללא שינוייחnnnnnnnnnnnnnnnn

3.1. Game

```
public class Game {

	public static void main(String[] args) {
		MainOffice game=new MainOffice(5, 4);
		game.play(60);
	}
}
```

עבודה נעימה!