בס"ד

משימה 4 – שמים הכל ביחד

זמן הגשה:

15/8/24 21:00

התוכנית תיווצר בפרוייקט הapps בתוך transportationApp (להוסיף, לא למחוק את מה שעשיתם).

כללי המשימה:

* במשימה הזו נחיל את כל הדברים שלמדנו.
* מלבד מה שמתבקש בהוראות – למבצע המשימה יש את החופשיות לעשות את הדברים כראות עיניו.
* העתקת קוד לא ראויה. גם לא שמישהו אחר יעשה את המשימה.
* ניתן להתייעץ וללמוד מכל מקור שהוא.
* מתוך התיאור צריך ליצור את האובייקטים.

# מה הסיפור לקוח?

הלקוח רוצה תוכנה-מודול נוסף בתוך תוכנת התחבורה. שתעקוב ותתעד את פרטי משלוח רכבי המשא.

נכון להיות התוכנה תתעד את הפרטים של אונייה משא, מטוס משא, ורכבת משא, כאשר התוכנה מנהלת תיעוד של העמסת פריטים על רכבי המשא והעברתם מנמל לנמל. בהמשך פרטים יותר מפורטים.

בתוכנה יהיו כלולים מס' מחלקות וחלקם אבסטרקטיות ומספר ממשקים. (abstract & interface)

# מה התוכנה תכיל (סיפור לקוח):

CargoVehicle (המלצה-לא חייב לקרוא לזה ככה) –

רכב משא שמכיל נפח איחסון (צריך להיכלל אורך, רוחב, גובה) , משקל נשיאה. זה **אבסטרקטי** אין כזה דבר רכב משא. אוניה, מטוס, ורכבת משא, יורשים אותו. רכב משא מממש את הממשק IContainable .

מכיל מתודה load שלוקחת מוצר כלשהו – רהיט, מכשיר חשמלי או מוצר כללי. ומעלה אותו. זה אמור לקרות מתוך נמל מסויים.

המוצר נארז, נבדק האם יכול להעלות לרכב, ונרשם עליו הפרטים של הרכב.

באונייה המוצר מתאחסן במכליות. ברכבת בתוך קרונות נפרדים. במטוס זה חדר של איחסון.

הרכב משא מקבל רשימה ראשונית של פריטים שהוא צריך להעמיס.

בנוסף רכב המשא יש לו מתודה שמחזירה האם מוכן ליסוע. התנאי הוא במידה והוא העלה את כל הפריטים, או במידה שלא – וגם שהנהג הראשי יתן לו אישור ליסוע. (יש משתנה אישור).

כל נמל כלשהו הוא איחסון (יוסבר בהמשך), וניתן להעמיס אליו חפצים ולפרוק ממנו (בגרסאות הבאות מחוץ למשימה זו, ניתן להוסיף העברות למחסנים אחרים, או איסוף לקוח).

# פרטי התיכנות

## ממשקים:

IPortable

IContainable

## מחלקות אבסטרקטיות:

StorageStructure – מבנה איחסון

CargoVehicle – רכב משא

התוכנה תתעד מעקב משלוחי סחורה ברחבי העולם.

משלוח מנמל לנמל (גרסה הבאה - ומשמה למחסנים עד שאוסף הלקוח את הפריט).

## מבני איחסון (ראה בהמשך תיאור מחלקה)

התוכנה תכיל עצמים **מחסן +** **נמל** (כרגע הנמל יהיה לכל כלי הרכב) שיורשים אובייקט אבסטרקטי – "מבנה איחסון" שמכיל את התכונות שמגדירות את הארץ שלו, את היכולת תכולה שלו במשקל, ונפח.

בכל מחסן יש כמות משתנה של חפצים מאחוסנים. ( IPortable)

## רכב משא

ישנה מחלקה **אבסטרקטית** רכב משא, שמכילה את התכונות: נהג, כמות משקל מקסימלית, כמות נפח מקסימלית, האם-יכול-ליסוע, האם-עומס-יתר, הנמל הבא. הנמל הנוכחי. ID של הנסיעה הנוכחית – מס' סידורי רנדומלי (אין צורך לטפל בייחודיות כרגע), בין 1000 ל9999, רשימת פריטים מסוג IPortable של רשימת הפריטים שצריכה לעלות למחסן. תשלום צפוי לנסיעה הבאה, IShippingPriceCalculator שמשמש לחישוב הנסיעה ע"י ספק ההובלה. **רכב משא מממש את הממשק IContainable .(חובה).** המרחק בקילומטרים לנמל הבא.

המלצה לניהול התשלום הצפוי (היסטורייה) להשתמש בDictionary .

<https://learn.microsoft.com/en-us/dotnet/api/system.collections.generic.dictionary-2?view=net-8.0>

### הפעולות שחייבות להיות אבסטרקטיות:

הפעולות שנמצאות בתוך Icontainable

Load(IPortable item): bool

מעלה פריט לאובייקט, מחזירה אמת אם הפעולה בוצעה בהצלחה וfalse להיפך.

Load(List<IPortable> items): bool

מעלה את כל הפריטים המוגשים לאובייקט. במידה ובוצע בהצלחה לגמרי (שהכניס את כל הרשימה) מחזיר אמת, אחרת שקר.

UnLoad((:bool

פורק הכל, כנ"ל לגבי הצלחה.

פרוק את כל המשא, פרוק משא לפי רשימה (List) פרוק משא לפי פריט מסויים.

Unload(IPortable item): bool

Unload(List<IPortable> items): bool

פעולה וירטואלית שניתנת לדריסה:

GetPricingList: String

שמחזירה את כל רישום התימחור לפי הפריטים (כמו שיפורט שם), בדומה לקבלה של סופרמרקט.

את המחלקה הזו יורשים רכבת משא, מטוס משא, ואוניית משא. וההסבר עליהם הוא כדלהלן:

### ספינת משא–

מכילה מס' ידוע מראש של מכולות שמממשות (חובה) את **IContainable**. (מכולה = container)

### מטוס משא- מממש את IContainable

מכיל חדר אחד לאיחסון



### רכבת משא

**מכילה** הרבה קרונות (ייתכנו גם מסוגים שונים) שמממשות את IContainable .

### מה צריך רכב משא לעשות?

צריך במתודה אחת, להעמיס את הפריטים מהרשימה לתוך האיחסון שלו. LoadCargo כנ"ל. לעבור לפי הסדר של הפריטים.

בכל פריט שהועמס, צריך להוסיף למחיר הצפוי את המחיר. (לקחת בחשבון גם משקל ונפח)

במידה והגיע למקסימום – לעצור. זה סוף המתודה.

צריך לאשר שהרכב מוכן לנסיעה. אישור לנסיעה קורה רק אחרי שניסו פעם אחת להעמיס את הדברים לתוך הרכב, וכשהנהג עצמו מאשר את הנסיעה. **יש לבדוק כשמשנים את המשתנה שקשור לאישור, שזה הנהג מאשר אותו.**

**ברגע שמאשרים נסיעה,** צריך לחשב שוב את כל הפריטים לפי המחיר הצפוי ולקבוע סופית.

במידה והנמל הבא null עדיין, הפעולות של חישוב מחיר, ואישור נסיעה לא רלוונטיות ואמורות להעצר בזמן. וכן סע ליעד הבא.

הפעולה **סע ליעד הבא**, לוקחת את הרכב מהנקודה הזו לנקודה הבאה. ואז - הנקודה הבאה נהיית הנוכחית.

מעכשיו ההעמסות והפריקות יקרו מול הנמל הנוכחי.

פעולה נוספת:

פרוק משא – מס' פעולות –

פרוק את כל המשא, פרוק משא לפי רשימה (List) פרוק משא לפי פריט מסויים.

הפעולה פורקת את הנתונים שמועברים לה (בהתאם למתודה) מסירה אותם מהרכב, ומכניסה אותם לתוך הנמל.

לגבי חישוב המחיר – ראה בהמשך. IShippingPriceCalculator

## מחלקה Driver

כשמה - מייצגת נהג פשוט.

שם משפחה, פרטי. מס' זהות.

סוג הנהג – מיוצג ע"י enum עם אחד הערכים בהקשר של סוג רכבי **משא**: מכונית פרטית, אופנוע, משאית, מטוס, רכבת, אונייה (CargoType )

מתודה Approve שמקבלת רכב משא עם יעד הבא, ומאשרת אותו.

* לפי דרישות התוכנה, במידה והנהג רוצה לאשר, אין מי שיעצור אותו, לכן חסימות לפעולה הזו, אמורות לקרות ממקום אחר, ולא מפה.

## מחלקת מבנה איחסון

מחלקה abstract מבנה איחסון. מממשת את IContainable

יורשות שלה: נמל, מחסן.

היא עצמה מכילה:

מיקום – ארץ, עיר, רחוב, מס'.

נמל מכיל enum שמייצג נמל של מה הוא מאחד הערכים שקשורים לנהג – סוג רכב המשא.

מאפשר להכניס אליו דברים כמו שמצויין בממשק Icontainable

בכל מחסן יש כמות משתנה של חפצים מאחוסנים. ( IPortable) צריך לממש מתודות שגורמות לזה להתנהל הגיוני. לא מעבר.

זה לגרסאות הבאות (לא למשימה הזו):

~~מכיל בעצמו מערך עם ערכים לא משתנים של מחסנים. (array not List)~~

~~במידה ומכניסים אליו דברים, זה נכנס למחסן הראשון שפנוי וכן הלאה. (אין צורך כרגע להגדיר מתודה שמבצעת הכנסה מסויימת למחסן.~~

# ממשק IContainable – מגדיר האם האובייקט יכול לאחסן.

מכיל את המתודות הבאות, עם הסברים מה הם אמורות לעשות בהגיון הפשוט.

עליך לדאוג שכל האובייקטים שמממשים את זה, אכן מבצעים בצורה נכונה את הפעולות האלו.

Load(IPortable item): bool

מעלה פריט לאובייקט, מחזירה אמת אם הפעולה בוצעה בהצלחה וfalse להיפך.

Load(List<IPortable> items): bool

מעלה את כל הפריטים המוגשים לאובייקט. במידה ובוצע בהצלחה לגמרי (שהכניס את כל הרשימה) מחזיר אמת, אחרת שקר.

UnLoad((:bool -

פורק הכל, כנ"ל לגבי הצלחה.

פרוק את כל המשא, פרוק משא לפי רשימה (List) פרוק משא לפי פריט מסויים.

Unload(IPortable item): bool

Unload(List<IPortable> items): bool

**מובן מאליו שבאובייקט שמממשים את הממשק הדברים צריכים לקרות בהתאם, במידה וזה פריטים נוכחיים להוציא אותם מהאובייקט וכו'**

IsHaveRoom(): bool

האם יש מקום.

IsOverload(): bool –

כשמו – האם עומס יתר.

GetMaxVolume() – קבל את הנפח המקסימלי של האובייקט

GetMaxWeight() – קבל את המשקל המקסימלי שהאובייקט יכול לשאת

GetCurrentVolume() – כמשמעו (= כמו שזה נשמע)

GetCurrentWeight() - כמשמעו

# ממשק Iportable – מגדיר האם האובייקט ניתן להעמסה.

IPortable –

**כל המידות צריכות להיות עשרוניות. מדובר במידת סנטימטר.**

GetArea - שטח

GetSize(): double[3] – w,l,h = width, length, height -

GetVolume - נפח

GetWeight - משקל

PackageItem – ארוז את הפריט

IsPackaged

UnPackage

IsFragile – האם שביר.

GetLocation – מחזירה את המבנה איחסון הנוכחי שהיא נמצאת בו

IsLoaded – האם נמצאת על רכב משא

# מחלקות שממשות את פריט IPortable.

פריט חשמלי

פריט כללי – מיועד לפריטים כלליים שעדיין לא נוצר להם אובייקט מותאם.

רהיט – אבסטרקט שיורשות אותו כיסא ושולחן וספה. תעשו משתנים ומתודות בקטנה לפי הצורך.

# IShippingPriceCalculator

אנחנו יוצרים את הממשק הזה, כי היום אנחנו משתמשים בחישוב של ספק מסויים לפי זמן מסויים, בעתיד ייתכן שנשנה את זה, או יהיו עוד ספקים, ואז המחיר יצטרך להשתנות.

CalculatePrice(IPortable item, int travelDistance) – מחזיר את החישוב בתור מס' עשרוני לפי פריט

CalculatePrice(List<IPortable> items, int travelDistance) – מחזירה את החישוב לפי פריטים בתור מס' עשרוני.

שיטת החישוב כיום:

**צריך ליצור מחלקה מסויימת שמממשת את הדברים האלה או אחד לכולם, או לכל רכב משא בנפרד**

כל חפץ מקבל מס' מסויים של יחידות לפי (מוסיף בכל תנאי עוד) –

1. נפח – על כל 100 סמ"ק – יחידה אחת
2. משקל – על כל קילו – יחידה אחת
3. במידה ושביר (כולל חפצים שיכולים להנזק במידה ומוטחים כמו מוצרים חשמליים) – כל היחידות מוכפלות.

לאחר מכן – מס' היחידות כפול מס' הקילומטרים \* תעריף לק"מ.

התעריפים ב$:

רכבת משא – 5

אונייה - 20

מטוס – 50

# בדיקות

שיהיו לפחות 15 בדיקות שמתארות ובודקות 3 נסיעות.

בונוס – לא במקום הבדיקות –

יכולים לפתח כמו ממשק שניתן לבחור איזה נמל, ולהוסיף מתוך רשימת אייטמים, ליצור חדשים וכו' ליצור ממשק של קונסול.

# קישורים חשובים:

לגבי list :

<https://learn.microsoft.com/en-us/dotnet/api/system.collections.generic.list-1?view=net-8.0>