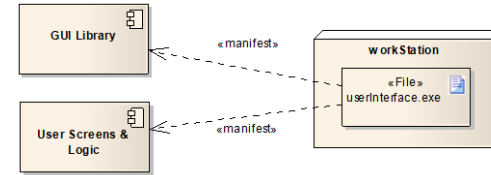
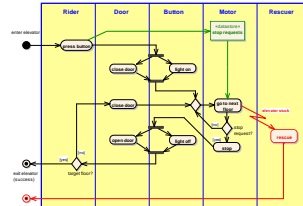
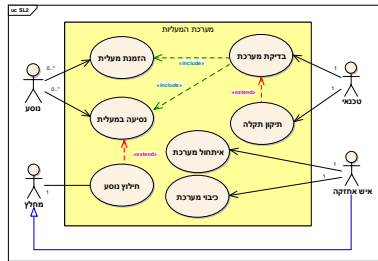


תכן תוכנה מונחה-עצמים 1 (מודל מחלקות)

Object-Oriented Software Design



פעילות תכן התוכנה



• קלט

– ארכיטקטורת התוכנה

• מודל רכיבים (component model)

– מפרט תהליכי התוכנה

• activity models

• use case model

• תוצרים עיקריים

– מבנה וארגון התוכנה

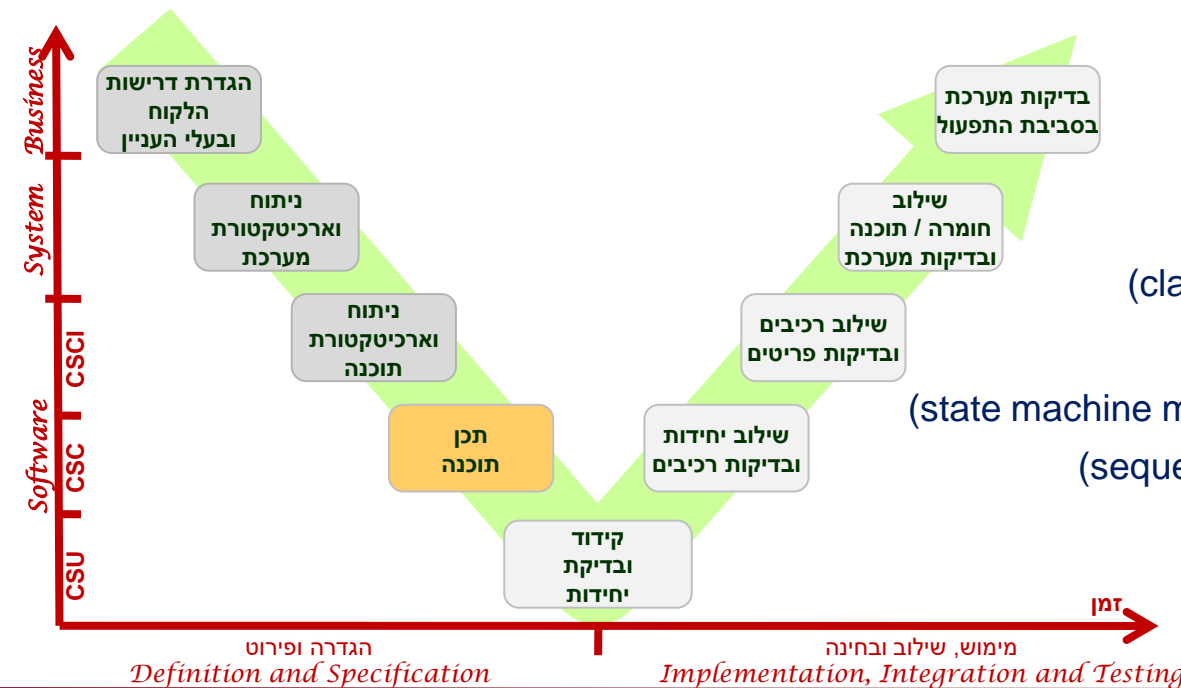
• מודל מחלקות (class model)

– תהליכי תוכנה מפורטים

• מודל מכונת מצבים (state machine model)

• מודל רצפים (sequence model)

רמת עניין



דרישות פונקציונליות ודרישות לא פונקציונליות - תזכורת



דרישות לא פונקציונליות

- מגדירות מאפיינים ואילוצים על אופן מימוש תכולת הפתרון
 - מקבלות מענה כאשר הפתרון הנבחר (התכן/המימוש) עומד במאפיינים ובאילוצים המוגדרים

דרישות פונקציונליות

- מגדירות את תכולת הפתרון
 - מקבלות מענה ספציפי וישיר בתוך הפתרון (התכן/המימוש)

דרישות/החלטות המשפיעות על התכן



- התכן הוא הפעילות בה נקבעים פרטי המימוש – נדרש לקבל החלטות לגבי חלופות מימוש שונות

- הבסיס לקבלת ההחלטות

– אילוצים

- אילוצי תכן שנקבעו בדרישות
- אילוצים ניהוליים שנקבעו בדרישות
- אילוצי המפתח (זמינות טכנולוגית, זמינות משאבים, דרישות שימוש חוזר וכו')
- החלטות שהתקבלו בשלבים קודמים
 - ארכיטקטורה מערכתית
 - חומרה

- התכן נועד להציע דרך למימוש הדרישות הפונקציונליות תוך מתן מענה לדרישות הלא-פונקציונליות

דרישות/החלטות אופייניות עבור תכן התוכנה

- החומרה
 - סוג המחשבים, ארכיטקטורת החומרה, תקשורת
- מערכת ההפעלה
 - אופן הקריאה לשרותי מערכת ההפעלה מתוך התוכנה, תהליכים
- שפת התכנות
 - טיפוסים, מבני בקרה, פונקציות בנויות, תמיכה בתהליכים, ...
- סטנדרטים
 - למשל TCP/IP, .Net, CORBA
- שילוב מוצרי מדף
 - למשל GIS = Geographic Information System
- שילוב מערכת קיימת (legacy system)
 - בסיסי נתונים קיימים, חומרה קיימת

תוכן העניינים

➤ דרישות / החלטות המשפיעות על התכן

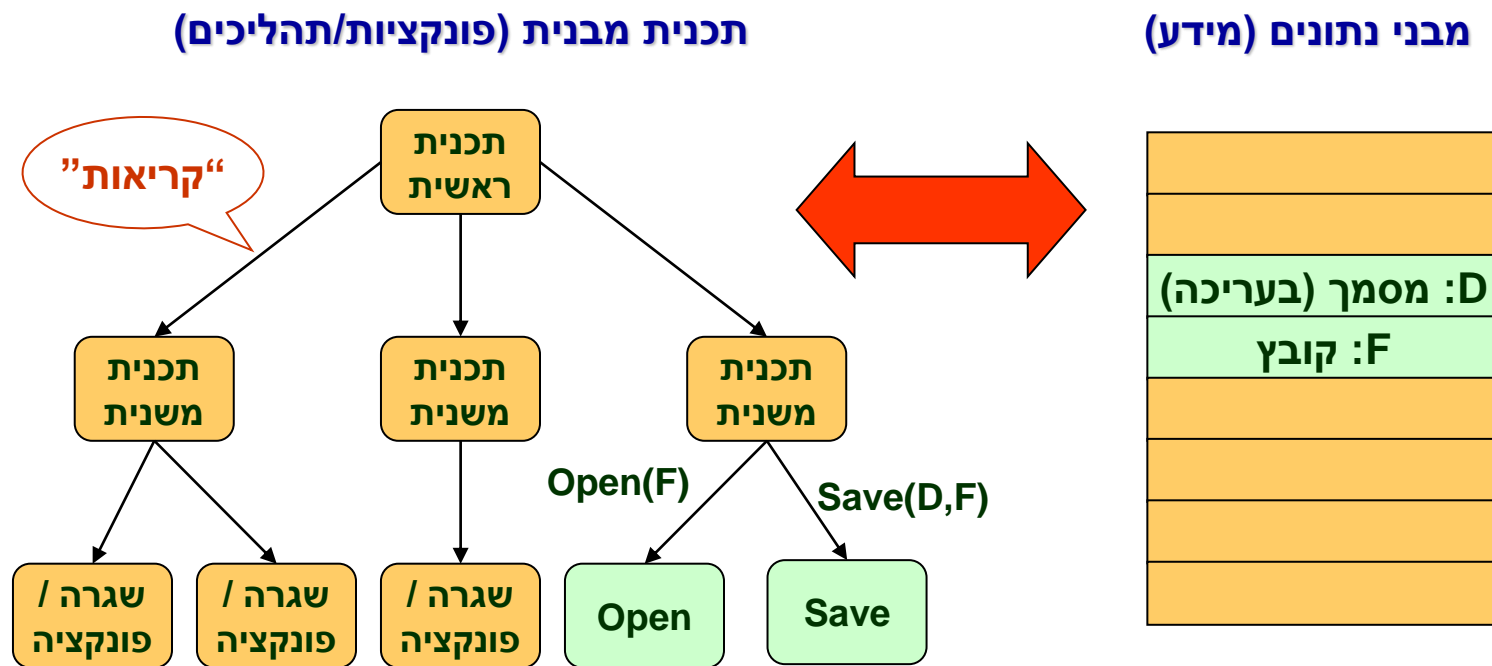
➤ מבנה וארגון התוכנה

➤ תבנית מונחית-העצמים

➤ מודל המחלקות (class model)

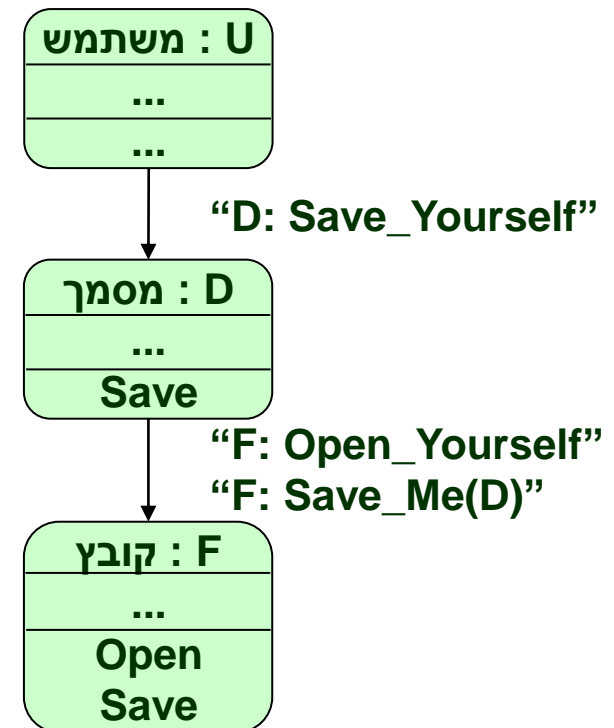
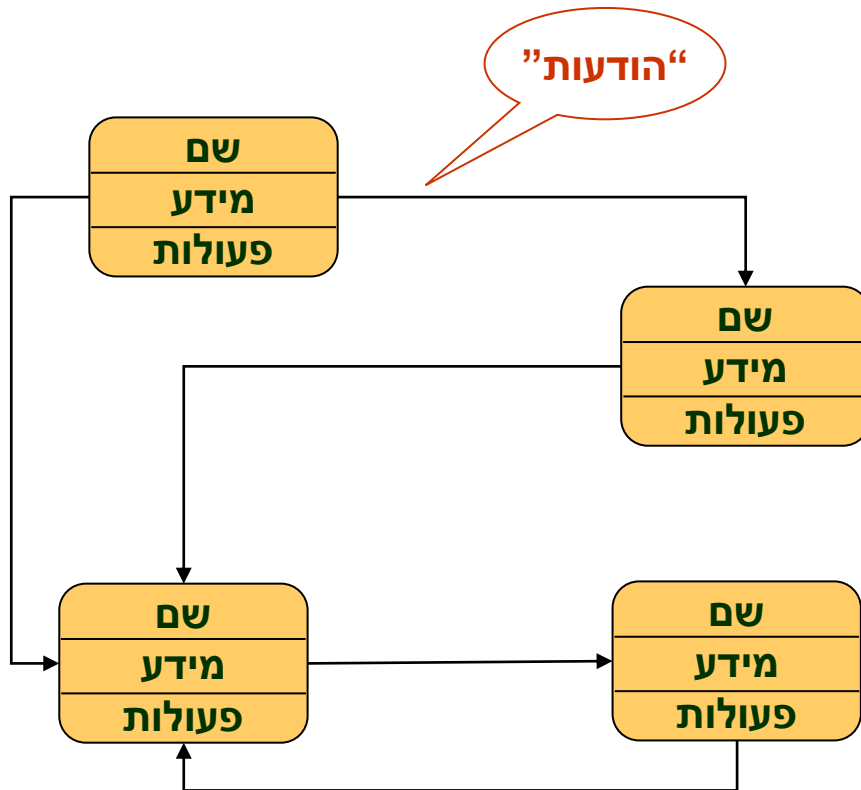
תוכנה מבנית (structured) [קלאסית]

- מבנה
 - הפרדת המידע (data) מהפונקציונאליות
- ביצוע
 - “עץ” קריאות בין מודלים
 - כל מודול מבצע מניפולציות על המידע



תוכנה מונחית עצמים

- מבנה
 - המידע והפונקציונאליות כמוסים (encapsulated) בתוך "עצמים"
- ביצוע
 - העברת "הודעות" בין אובייקטים



תבנית מונחית העצמים (Object Oriented paradigm) – מושגי יסוד

- עצם (object)

- ישות בדידה

- גבולות וזיהוי מוגדרים

- מכיל בתוכו (encapsulates) מצב והתנהגות

• **Properties** - תכונות לתיאור מצב = מבני נתונים - data members, attributes

• **Behavior** - התנהגות = פעולות / פונקציות - member functions, methods

- מחלקה (class)

- מתאר (descriptor) של קבוצת עצמים,

בעלי מאפיינים משותפים:

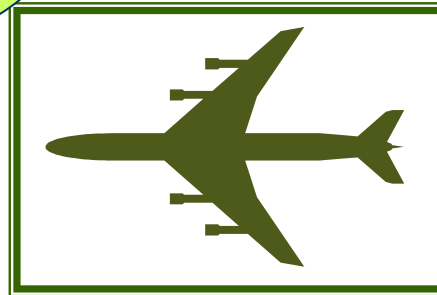
- תכונות

- פעולות

- יחסים

- התנהגות

מחלקות קיימות בקוד אך ורק
בזמן ההגדרה
עצמים קיימים בזכרון אך ורק
בזמן הריצה



עצמים ומחלקות

- עצמים (objects)

- היחידות הבסיסיות של התוכנה
- כל עצם מנהל את המידע שבאחריותו באמצעות הפונקציונאליות שהוקנתה לו
- עצמים קיימים בזיכרון המחשב בזמן ריצת התוכנית
 - ניתן לבנות/להרוס עצמים באופן דינמי תוך כדי ריצה
 - constructor = פונקציה הבונה עצם חדש
 - destructor = פונקציה ההורסת (מוחקת) עצם קיים
- לכל עצם יש מפתח גישה ייחודי (handle, pointer), הניתן לו ברגע בנייתו

- מחלקות (classes)

- התבניות על פיהן נוצרים עצמים חדשים
- התבנית מכילה 3 "תאים" (compartments)
- המחלקות מוגדרות ב**קוד** ע"י כותב התוכנה
- עצמים הם מופעים ספציפיים (instances) של מחלקות

שם
מאפיינים (מבני נתונים)
פעולות (פונקציונאליות)

יצירה ותפעול של עצמים

יצירת עצם "מכונית" חדש

```
theBlueCar = new(Car)
```

איתחול פרטי המכונית

```
theBlueCar.maker = "mazda"
```

```
theBlueCar.model = "CX-7"
```

רישוי וטסט

```
theBlueCar.licensePlate = "12-345-67"
```

```
theBlueCar.testDate = 08/09/2010
```

מכירה

```
theBlueCar.sellTo(Lior)
```

```
function sellTo(X) : owner = X
```

Car
+ maker: string + model: string + licensePlate: string + testDate: Date - owner: Person
+ sellTo(Person) : void + getOwner(int) : Person + testIsValid(Date) : boolean



theBlueCar : Car

```
maker = "mazda"  
model = "CX-7"  
licensePlate = "12-345-67"  
testDate = 08/09/2010  
owner = Lior
```

תוכן העניינים

➤ דרישות / החלטות המשפיעות על התכן

➤ מבנה וארגון התוכנה

➤ תבנית מונחית-העצמים

➤ מודל המחלקות (class model)

תרשים מחלקות (Class Diagram) - תחביר

ClassName
- privateAttribute: Type + publicAttribute: Type
- privateMethod(X:TypeX, Y:TypeY) : ReturnType + publicMethod(X:TypeX, Y:TypeY) : ReturnType

- מחלקה

- שם המחלקה

- מאפיינים (משתנים)

- מאפיין פרטי: ניתן לגשת אליו רק מתוך המחלקה עצמה

- מאפיין ציבורי: ניתן לגשת אליו גם מבחוץ

- מתודות (פונקציות)

- מתודה פרטית: ניתן לקרוא לה אך ורק מתוך המחלקה עצמה

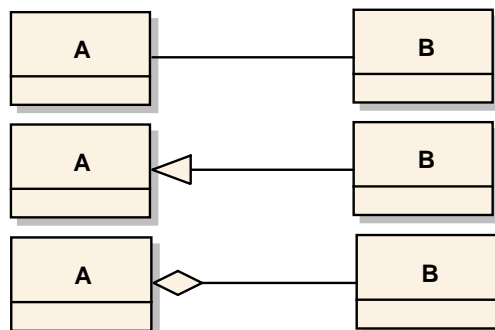
- מתודה ציבורית: ניתן לקרוא לה גם מבחוץ

- קשרים

- זיקה (association)

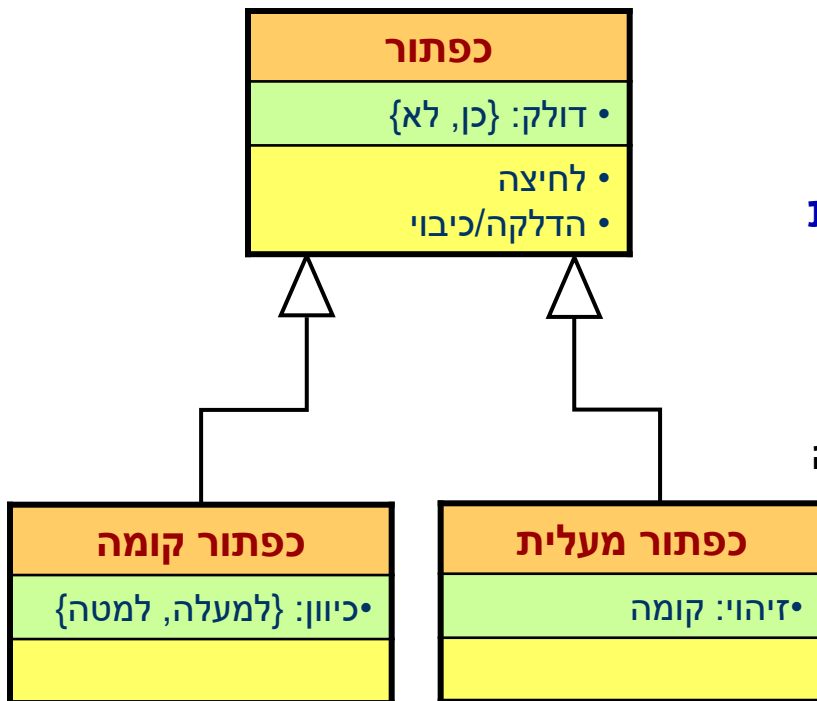
- הורשה (Inheritance)

- הקבצה (aggregation)



יחס הורשה (inheritance) / הכללה (Generalization)

- מחלקה B יורשת/מכלילה את מחלקה A:
 - B מכילה את כל המאפיינים של A
 - B מכילה את כל הפעולות של A
 - בנוסף, B מכילה מאפיינים ופעולות משל עצמה
- “B is-a A”
- B היא תת-מחלקה (sub-class) של A
 - מינוח לא מוצלח, כי B מכילה יותר מאשר A
- **יחס ההורשה יוצר מבנה היררכי של מחלקות**
- **מחלקה אבסטרקטית**
 - מחלקה שלא ניתן ליצור ממנה עצמים
 - כל העצמים נוצרים ממחלקות היורשות אותה
 - לדוגמה: “כלי רכב”

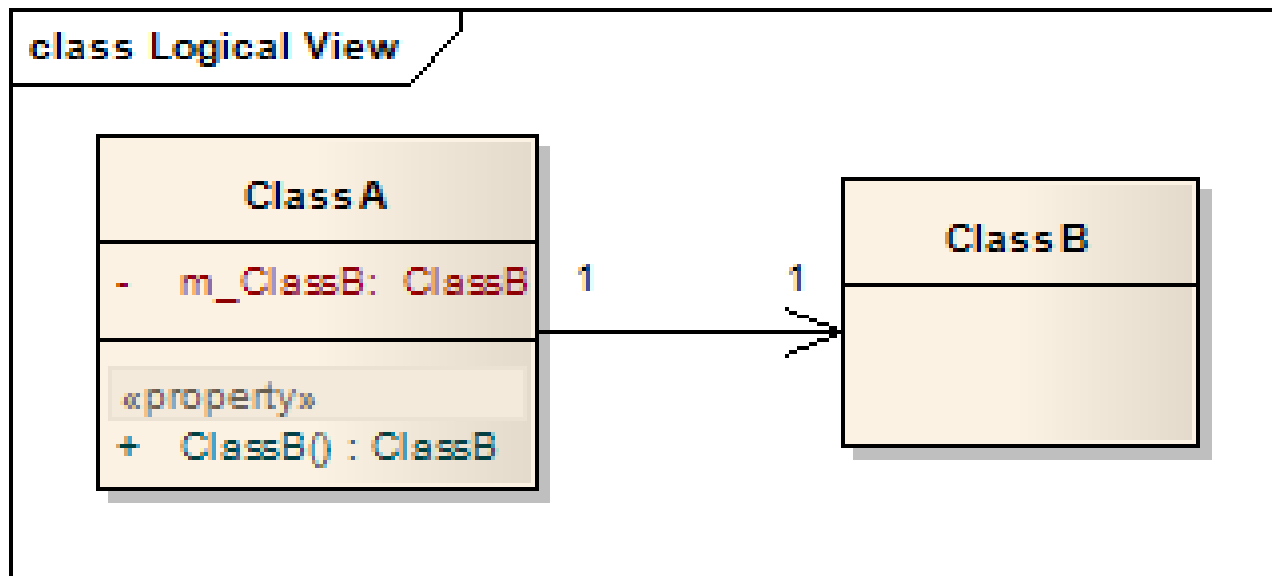


בעיות ביחסי הורשה

- הורשה מרובה (multiple inheritance)
 - מחלקה אחת יורשת משתי מחלקות שונות
 - הבעיה: עלולות להיווצר סתירות במאפיינים/פעולות
 - הפתרון: רוב שפות התכנות אינן מרשות הורשה מרובה (מבנה של עץ)
- הורשה עמוקה מדי
 - $X \rightarrow \dots \rightarrow C \rightarrow B \rightarrow A$
 - הבעיה: קושי במעקב אחר הקשר (קשיי תחזוקה)
 - הפתרון: "שבירת" העץ בנקודות בהן הזיקה חלשה יותר
- הורשה מדומה
 - לדוגמה: ריבוע הוא סוג של מלבן, לכן מלבן \rightarrow ריבוע
 - הבעיה: למלבן יש שני מאפיינים (אורך, רוחב) ולריבוע רק אחד (צלע)
 - הפתרון: להגדיר את ההורשה על בסיס תכונות משותפות (מאפיינים/פעולות)

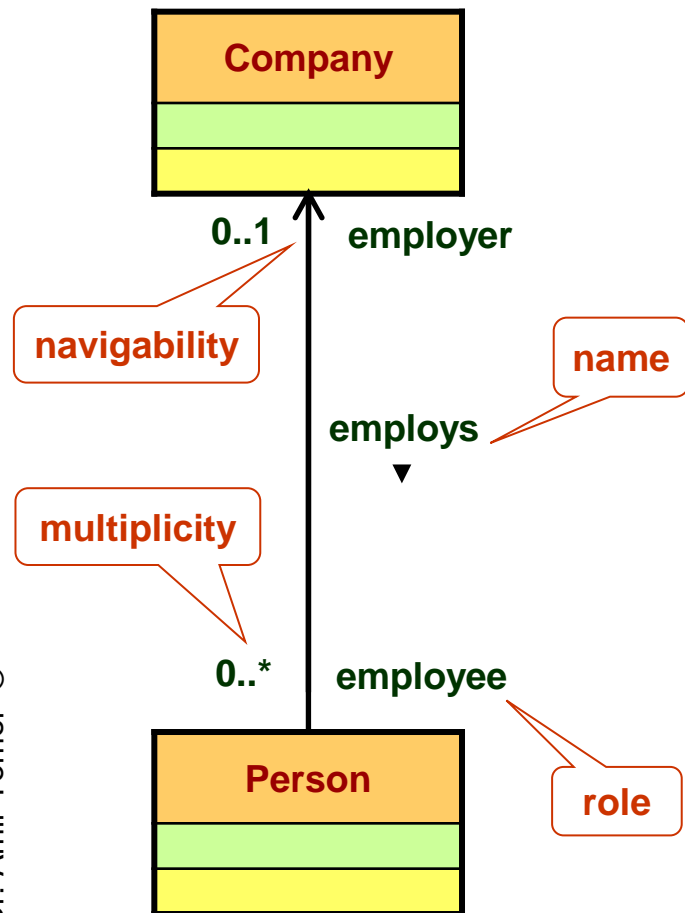
יחס הכלה חזק - Strong association

- מחלקה A קשורה למחלקה B כדי להראות שיש לה reference למופע של מחלקה B



זיקה - association

- יחס בין מחלקות המגדיר "היכרות" בין עצמים ממחלקות אלה (Company, Person)
 - "היכרות" = מצביעים הדדיים

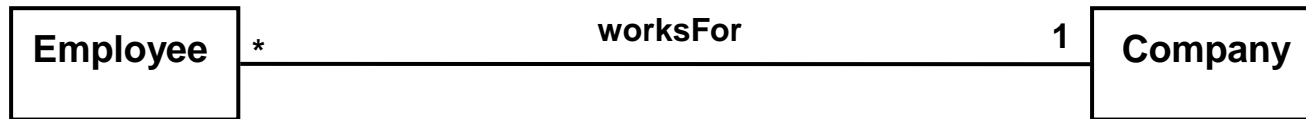


- מאפיינים המאפשרים הגדרה ברורה יותר של זיקה
 - שם (name) + כיוון
 - "Company employs Person"
 - תפקיד (role)
 - "Company" is the employer
 - "Person" is the employee
 - ריבוי (multiplicity)
 - "Company employs 0 or more Persons"
 - "Person is employed by 0 or 1 company"
 - ניווט (navigability)
 - Person "knows" who is its Company
 - Company does not know its Persons
 - יכול להיות מצב בו הקשר דו-כיווני.

יחסי זיקה: Many –to- one

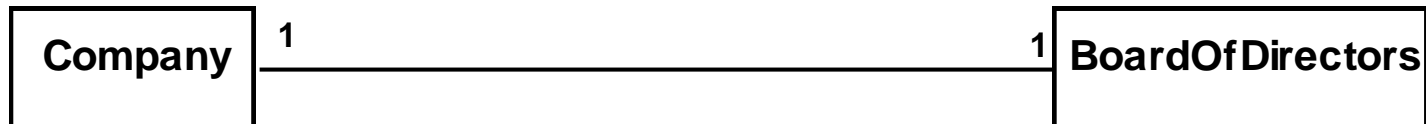
- יח Many –to- one

לדוגמא: לחברה עובדים רבים, עובד מועסק בחברה אחת.



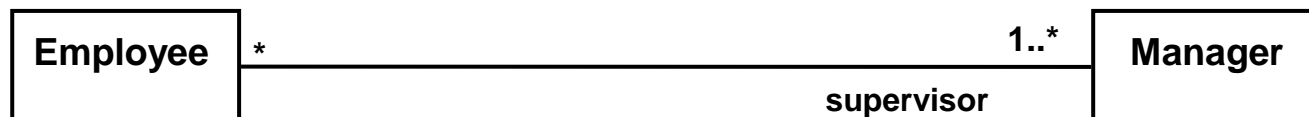
- יח One-to-one

לדוגמא: לחברה יש בדיוק board of directors אחד.



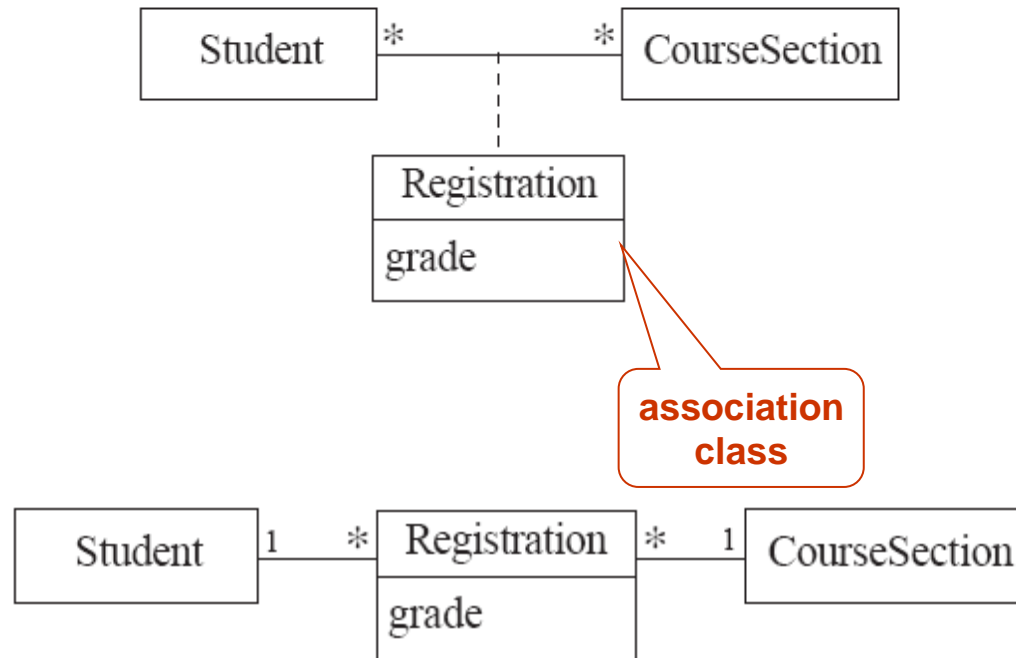
- יח Many-to-many

לדוגמא: עובד יכול לעבוד עבור מספר מנהלים, למנהל יכולים להיות מספר עובדים. ייתכן שלמנהל לא יהיו עובדים כלל.

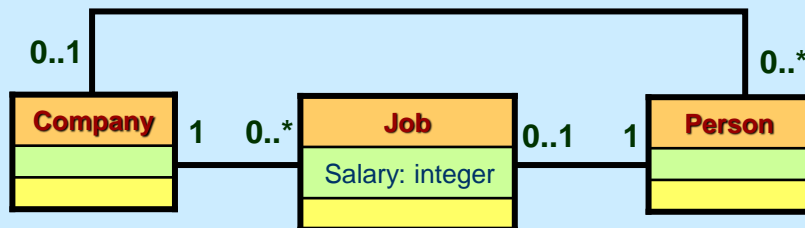
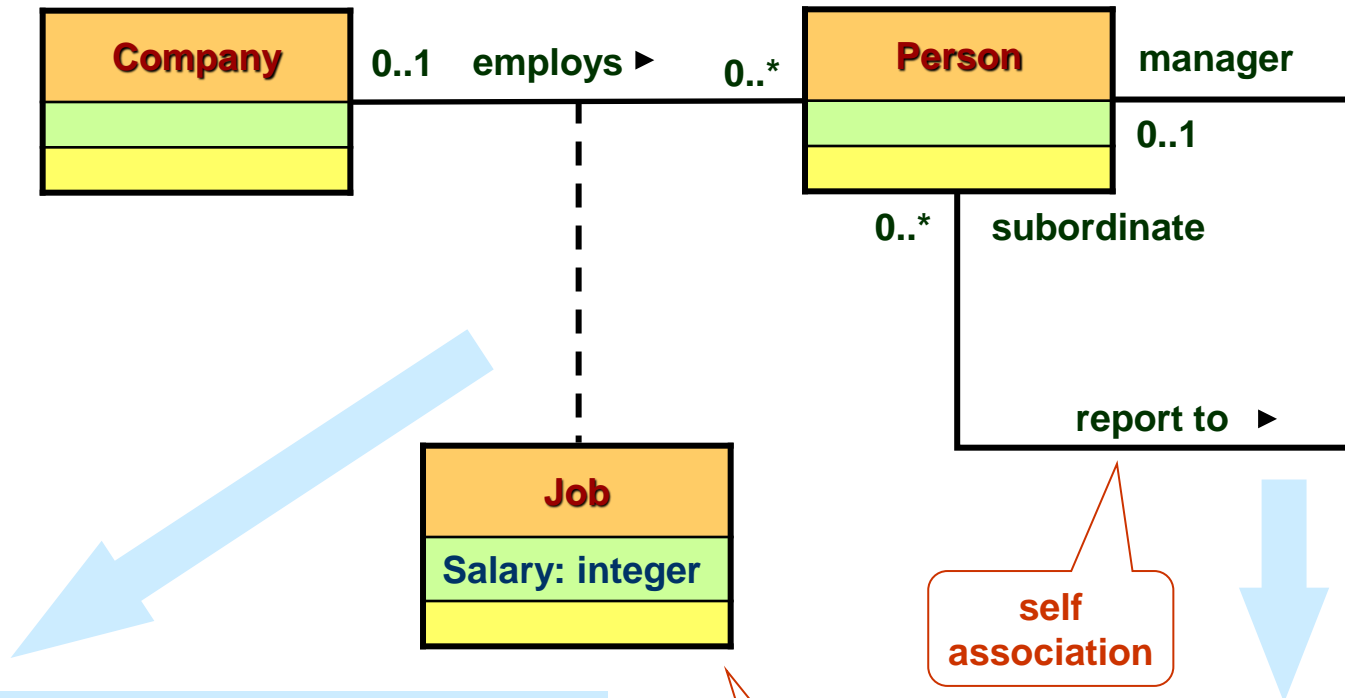


דוגמא: יחס Many-to-many

- סטודנט יכול ללמוד מספר קורסים.
- בקורס לומדים הרבה סטודנטים.



זיקה עצמית ומחלקת זיקה



הקבצה (aggregation)

- סוג מיוחד של זיקה
 - כל עצם ממחלקה B "מכיל" עצם (עצמים) ממחלקה A
- שני סוגי הקבצה:

– הקבצת הרֶכֶּב (composite aggregation)

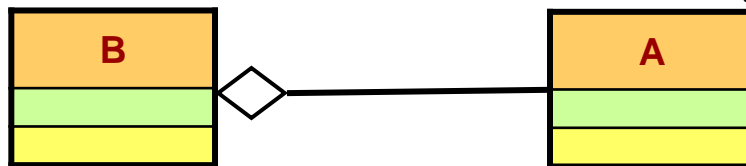


- A הוא חלק בלתי נפרד מ-B
- B אחראי לקיומו של A
- שמות נוספים:

– whole-part aggregation

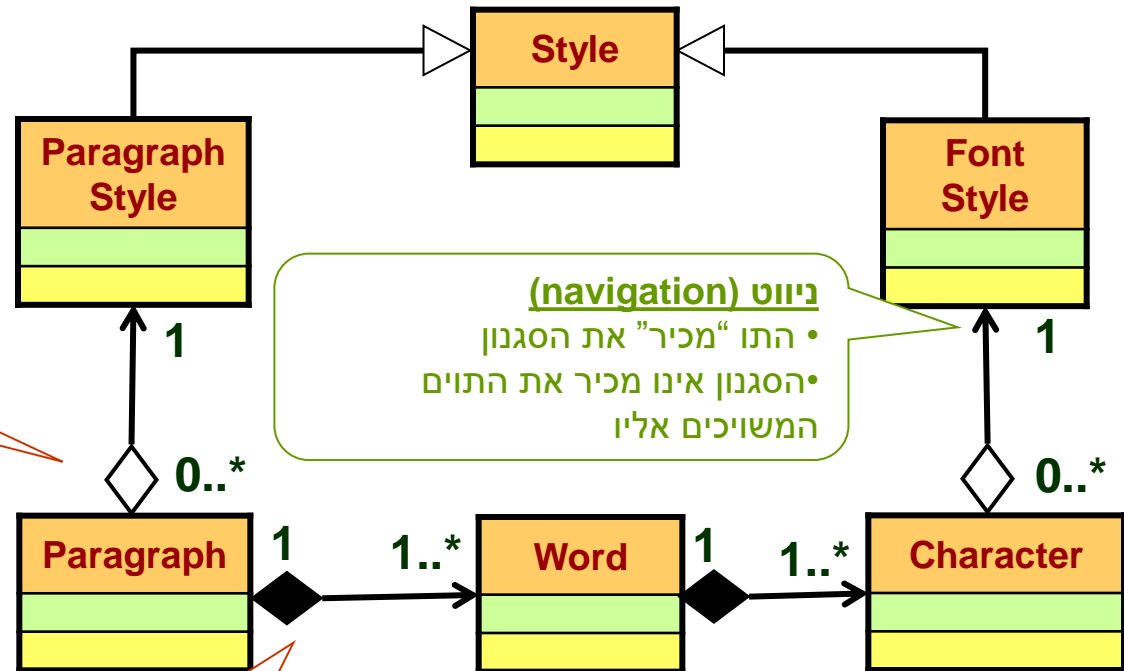
– non-shared aggregation

– הקבצת שיתוף (shared aggregation)



- A הוא חלק מ-B, אבל
 - קיומו של A אינו תלוי בקיומו של B
 - A יכול להיות משותף בין עצמים אחרים

הקבצה - דוגמה



הקבצת שיתוף (shared)

- לפסקה יש סגנון-פסקה אחד
- סגנון יכול להיות משותף למספר פסקאות
- הסגנון הוא ישות עצמאית, וקיומו אינו מותנה בקיום פסקאות
- מחיקת פסקה אינה מוחקת את הסגנון

ניווט (navigation)

- התו "מכיר" את הסגנון
- הסגנון אינו מכיר את התוים המשויכים אליו

הקבצת הרכב (composite)

- פיסקה מורכבת ממילה אחת לפחות
- כל המילים של הפסקה שייכות אך ורק לפסקה זו
- קיומה של המילה מותנה בקיומה של הפסקה
- מחיקת פסקה מוחקת את כל המילים המרכיבות אותה.

1. A Numbered Title

This is the first *paragraph* of this document. It contains 17 words and 80 non-blank characters.

מודל מחלקות של מרחב הבעיה

- כבר בשלב הניתוח המערכתי ניתן לבנות מודל מובנה של עצמים, המייצגים את מונחי מרחב הבעיה והקשרים ביניהם

– PDOM = Problem Domain Object Model

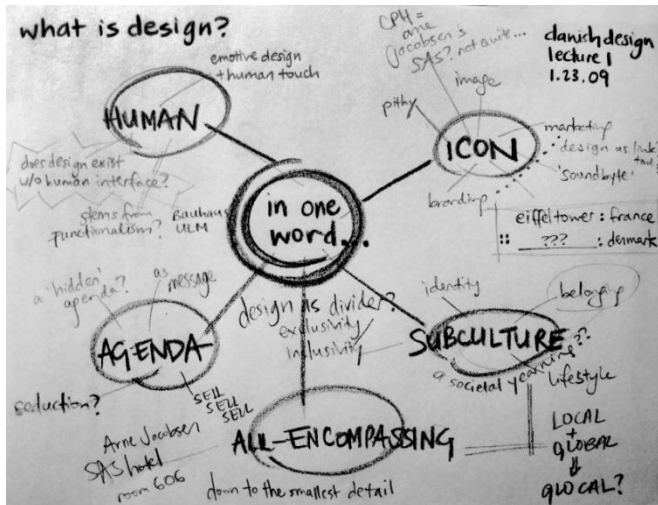
- המטרה
 - הבהרה וחידוד של המונחים והיחסים ביניהם
- שימושים

– יישוב סתירות ואי-בהירויות במפרטי הלקוח

– מילון מונחים של המערכת

– בסיס למודל מחלקות עבור התוכנה

- ישויות המידע בהן נדרשת התוכנה לטפל



מערכת המעליות – איתור ראשוני של מחלקות

מערכת הכוללת 3 מעליות משרתת בניין משרדים בן 10 קומות.

בכל קומה, פרט לקומת הקרקע ולקומה העליונה, נמצאים שני כפתורים – אחד לעליה ואחד לירידה. בקומת הקרקע נמצא כפתור אחד לעליה בלבד; בקומה העליונה נמצא כפתור אחד לירידה בלבד.

משתמש הנמצא בקומה כלשהי ורוצה לנסוע במעלית לוחץ על הכפתור המתאים לכיוון הנסיעה המבוקש. אם לא היה דלוק קודם לכן, נדלק הכפתור בעקבות הלחיצה. מעלית כלשהי הנמצאת בכיוון הנסיעה המבוקש תגיע לקומה, תוך דקה לכל היותר. עם הגעתה תיפתח הדלת והכפתור יכבה. בתוך כל מעלית נמצאים 10 כפתורים – אחד עבור כל קומה. כמו כן נמצאים במעלית כפתור לעצירת חירום וכפתור להזעקת חילוץ.

משתמש הנמצא בתוך המעלית ורוצה להגיע לקומה כלשהי לוחץ על הכפתור המתאים. אם לא היה דלוק קודם נדלק הכפתור בעקבות הלחיצה ולמעלית נוספת בקשה לעצירה בקומה המתאימה. כאשר תגיע המעלית לקומה המבוקשת היא תיעצר, הדלת תפתח והכפתור המתאים לקומה יכבה. לאחר השהיה תיסגר הדלת והמעלית תמשיך בפעולתה.

המערכת בנויה ופועלת על פי דרישות הבטיחות והאמינות שנקבעו בתקן המחייב למעליות. איש האחזקה של הבניין הוכשר לחלץ נוסעים תקועים באמצעות פאנל הפעלה מיוחד הנמצא בחדר המכונות.

על פי תקנות משרד העבודה נבדקת המערכת אחת לששה חודשים בידי טכנאי מוסמך. בזמן הבדיקה לא ניתן להשתמש במערכת.

מערכת המעליות – איתור ראשוני של מחלקות

מערכת הכוללת 3 מעליות משרתת בניין משרדים בן 10 קומות.

בכל קומה, פרט לקומת הקרקע ולקומה העליונה, נמצאים שני כפתורים – אחד לעליה ואחד לירידה. בקומת הקרקע נמצא כפתור אחד לעליה בלבד; בקומה העליונה נמצא כפתור אחד לירידה בלבד.

משתמש הנמצא בקומה כלשהי ורוצה לנסוע במעלית לוחץ על הכפתור המתאים לכיוון הנסיעה המבוקש. אם לא היה דלוק קודם לכן, נדלק הכפתור בעקבות הלחיצה. מעלית כלשהי הנמצאת בכיוון הנסיעה המבוקש תגיע לקומה, תוך דקה לכל היותר. עם הגעתה תיפתח דלת והכפתור יכבה.

בתוך כל מעלית נמצאים 10 כפתורים – אחד עבור כל קומה. כמו כן נמצאים במעלית כפתור לעצירת חירום וכפתור להזעקת חילוץ.

משתמש הנמצא בתוך המעלית ורוצה להגיע לקומה כלשהי לוחץ על הכפתור המתאים. אם לא היה דלוק קודם נדלק הכפתור בעקבות הלחיצה ולמעלית נוספת בקשה לעצירה בקומה המתאימה. כאשר תגיע המעלית לקומה המבוקשת היא תיעצר, הדלת תפתח והכפתור המתאים לקומה יכבה. לאחר השהיה תיסגר הדלת והמעלית תמשיך בפעולתה.

המערכת בנויה ופועלת על פי דרישות הבטיחות והאמינות שנקבעו בתקן המחייב למעליות. איש האחזקה של הבניין הוכשר לחלץ נוסעים תקועים באמצעות פאנל הפעלה מיוחד הנמצא בחדר המכונות.

על פי תקנות משרד העבודה נבדקת המערכת אחת לששה חודשים בידי טכנאי מוסמך. בזמן הבדיקה לא ניתן להשתמש במערכת.

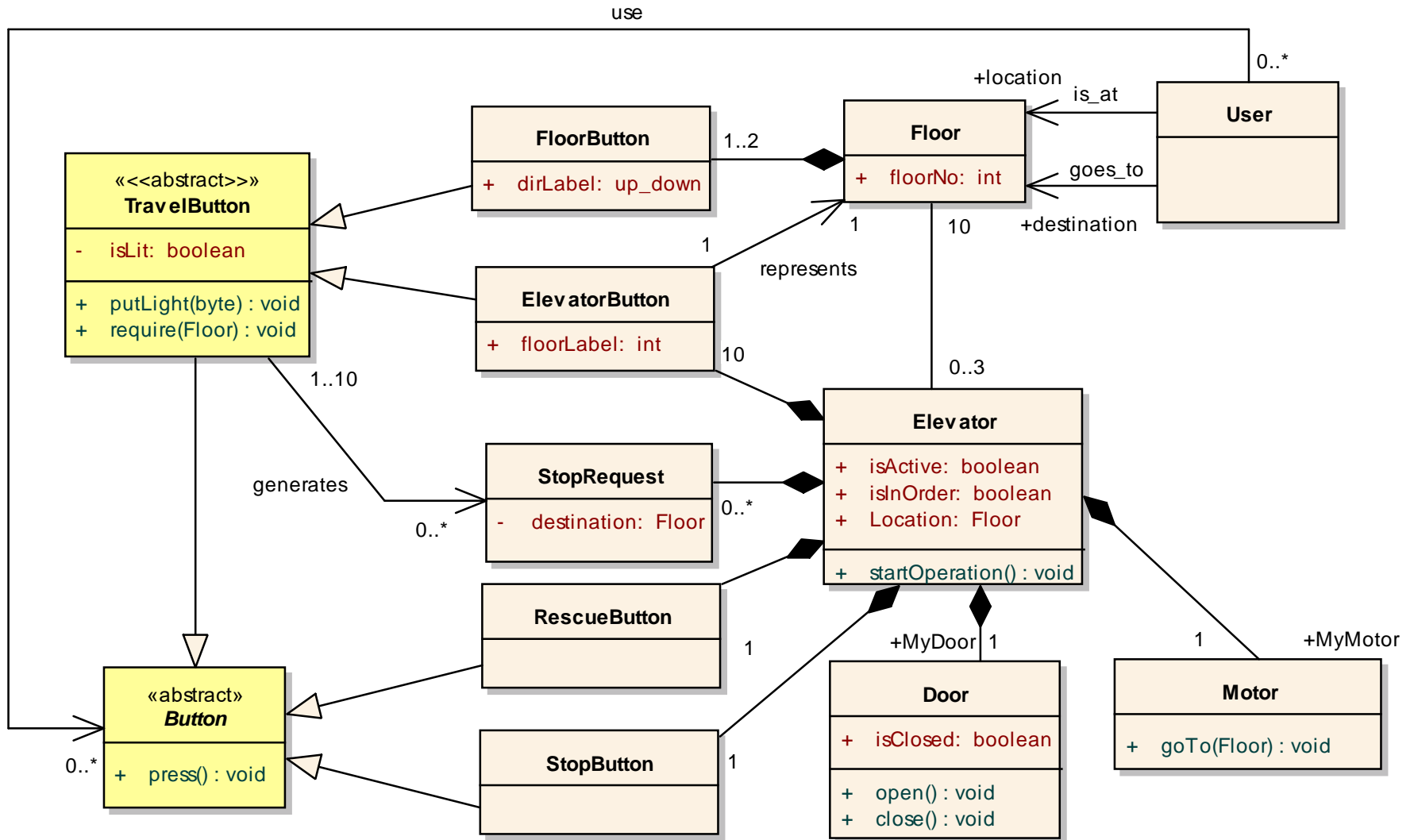
אפיון ראשוני (חלקי) של מחלקות פוטנציאליות

• מקורות:

- מסמכי הלקוח
- Use Case Model



מערכת המעליות - PDOM



- על בסיס סיפור הלקוח
 - **זהה מחלקות במרחב הבעיה**
 - הגדר מאפיינים ופעולות לכל מחלקה
 - **ערוך תרשים מחלקות במרחב הבעיה (PDOM)**

עקיבות הדרישות הפונקציונליות למודל המחלקות

- המחלקות שבמודל המחלקות אמורות לספק את כל הפונקציונליות המערכתית
- מכל דרישה פונקציונלית בטבלת הדרישות יש להצביע למחלקה או למחלקות הרלוונטיות
 - משתתפות בדרישה תפעולית (OR)
 - לדוגמה: "אם לא היה דלוק קודם נדלק הכפתור בעקבות הלחיצה" ← כפתור
 - מספקות את מבני הנתונים עבור דרישות המידע (DR)
 - לדוגמה: "בכל קומה יהיו שני כפתורים" ← קומה
- מכל מחלקה במודל המחלקות יש להצביע על הדרישות הפונקציונליות הרלוונטיות לה

מודל מחלקות ברמת התוכנה

- ארכיטקטורת התוכנה הגדירה רכיבי תוכנה וממשקים ביניהם

- תכן מפורט של התוכנה: מבנה והתנהגות התוכנה של כל רכיב

- רכיב תוכנה (software component) כמערכת

- המטרה

- לספק את הפונקציונאליות שהוקצתה לרכיב בתהליכים המערכתיים

- המרכיבים

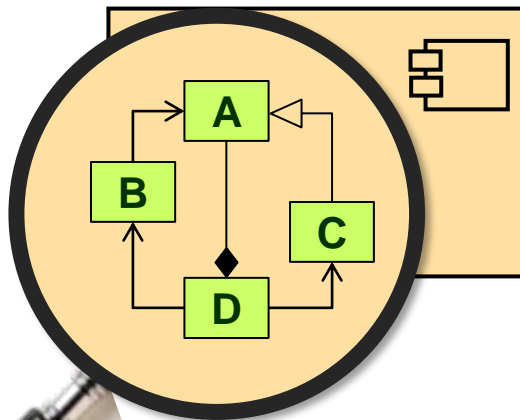
- מחלקות (בזמן ריצה: העצמים הנוצרים מהן)

- המבנה

- מודל מחלקות

- הפעולה המשותפת

- אינטראקציה בין עצמים (תתואר באמצעות sequence diagrams)



מועמדים לעצמים בתוכנה

- עצמים המייצגים ישויות פיזיות (מנוע, דלת, עמדת עבודה, ...)
 - מאפיינים: פרמטרים ונתונים לגבי הישות, קלט/פלט
 - מתודות: פונקציונאליות פיזית
- העצם המייצג משמש, למעשה, כממשק שבין התוכנה לישות הפיזית
- עצמים המייצגים ישויות לוגיות (תהליך, שירות, ...)
 - מאפיינים: פרמטרים ונתונים לגבי הישות, קלט/פלט
 - מתודות: פעולות המשמשות את התהליך/השירות
- עצמים המייצגים ישויות מידע (מאגרי נתונים, רשימות, תורים, ...)
 - מאפיינים: רכיבי המידע שבאחריות הישות
 - מתודות: פעולות על המידע (אחסון, שליפה, עדכון, ...)
- עצמים המייצגים עצמים הנמצאים ברכיב תוכנה אחר
 - "שיקוף" של העצמים החיצוניים
 - מימוש ממשקי תוכנה-תוכנה דרך תווך של חומרה

בניית מודל מחלקות לרכיב תוכנה

- מקורות

– PDOM

- מחלקות שצריכות להיות באחריות הרכיב
- מחלקות-אב שמהן ניתן לגזור מחלקות לרכיב

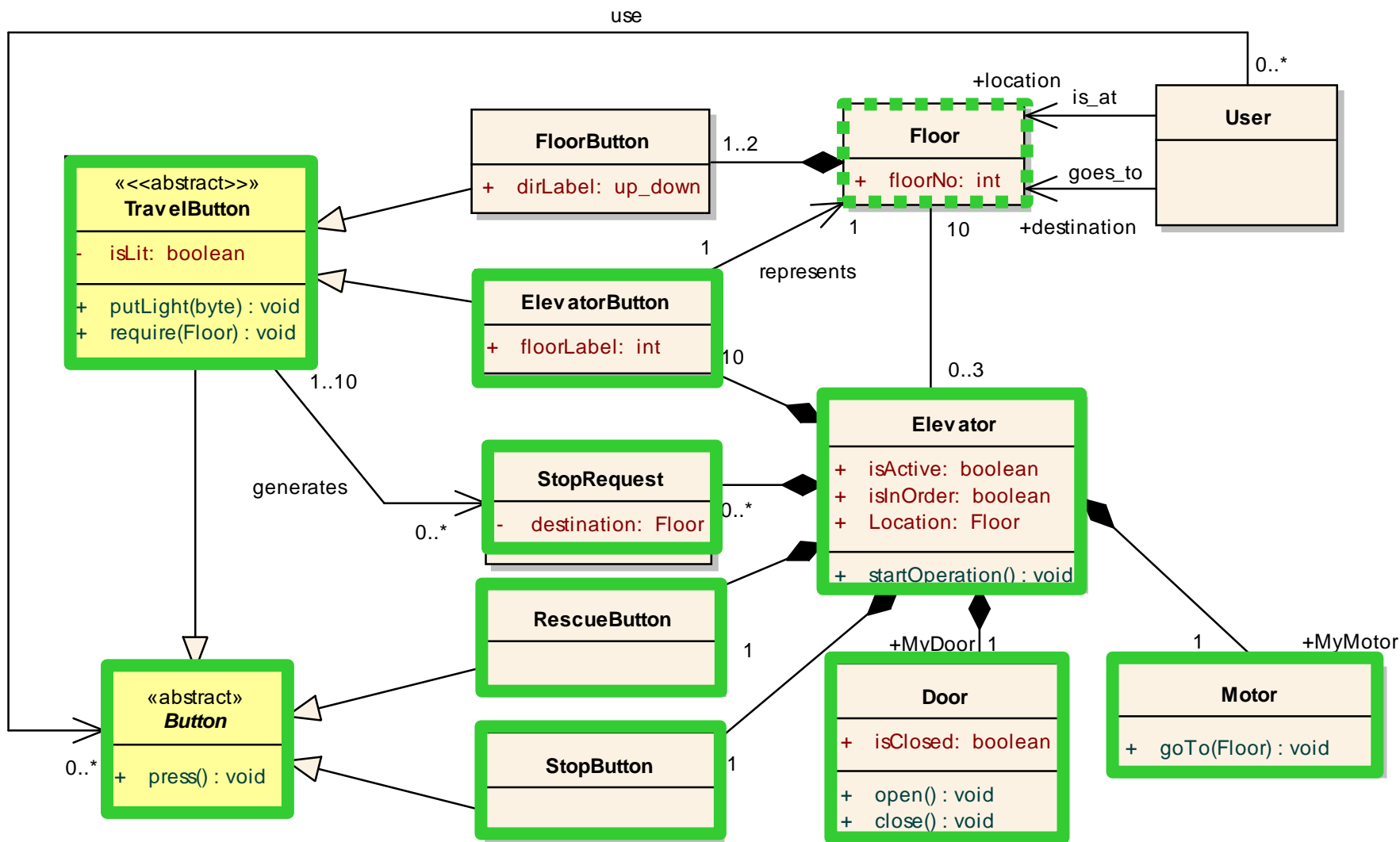
– SRS

- תהליכי תוכנה (Use Cases, Activity Diagrams)
- הקצאת הפונקציונאליות של הרכיב למחלקות (מאפיינים ומתודות)

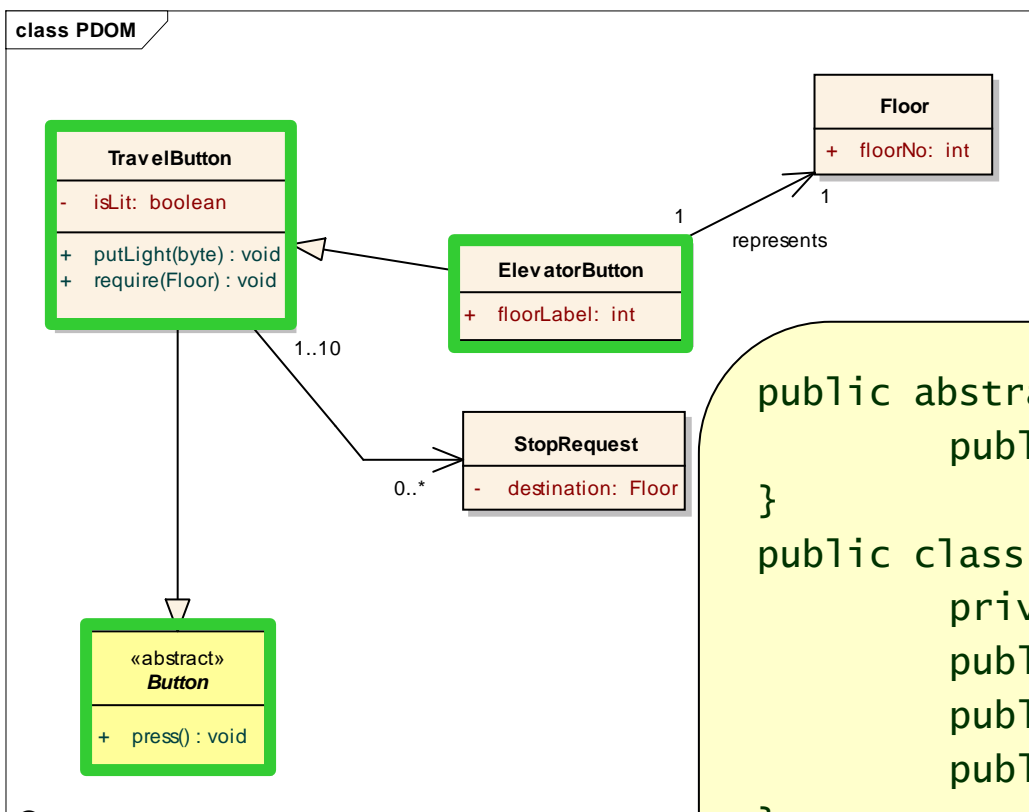
– ספריות תוכנה

- מחלקות "מן המדף" אותן ניתן לשלב בתוכנה הייעודית

רכיב "תפעול מעלית" – מחלקות רלוונטיות מה-PDOM



הפקת קוד ממודל המחלקות (Java)



```
public abstract class Button {
    public void press(){ }
}

public class TravelButton extends Button {
    private boolean isLit;
    public StopRequest m_StopRequest;
    public void putLight(byte on_off){ }
    public void require(Floor floor){ }
}

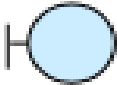


public class ElevatorButton extends TravelButton {
    public int floorLabel;
    public Floor m_Floor;
}
```

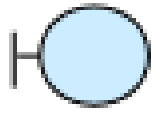
ארכיטקטורה של שלושת הרבדים - 3 tier model

- כפי שלמדנו, גישה מקובלת במערכות עתירות-תוכנה: חלוקה לרבדים.
- פעמים רבות מחלקים לשלושה רבדים:
 - רובד הנתונים (data tier):
 - רכיבים האחראיים לאיחסון ולאיחזור נתונים.
 - הרכיבים נקראים גם **אובייקטי ישות - Entity object**
 - רובד היישומים (application tier)
 - רכיבים האחראיים לפונקציות העיבוד השונות במערכת
 - הרכיבים נקראים גם **אובייקטי בקרה - Control object**
 - רובד התצוגה (presentation tier)
 - רכיבים של ממשקי-משתמש האחראים לאינטראקציה עם עמדות ההפעלה
 - הרכיבים נקראים גם **אובייקטי ממשק - Boundary object**

מודל שלושת הרבדים בתרשים המחלקות

- בתרשים המחלקות יש לציין עבור כל מחלקה לאיזה רובד היא משתייכת.
- ניתן לסמן באמצעות סימון של stereotype (<<שם הרובד >>) או באמצעות סימן גרפי ע"פ הטבלה:

סימן	<< שם >>	סימן גרפי
רכיב ממשק	<<Boundary>>	
רכיב בקרה	<<Control>>	
רכיב ישות	<<Entity>>	



רכיב ממשק Boundary

- קיימים שני סוגים של רכיבי ממשק:

– ממשק למערכות חיצוניות - יתואר במונחים של פרוטוקולים.

– ממשק למשתמש אנושי. כיום בד"כ רכיב GUI .

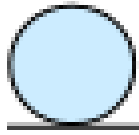
- באמצעות רכיבים אלה השחקנים מתקשרים עם המערכת.

- כל שחקן זקוק לרכיב ממשק (או לרכיבי ממשק) לצורך פעולותיו עם המערכת.

- הרכיבים מתרגמים את פעולות השחקנים לאירועים במערכת, ואת אירועי המערכת לעצמים המוצגים בפני השחקנים.

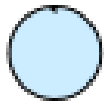
- רכיב ממשק יכול לשרת מספר שחקנים. לשחקן יכולים להיות מספר רכיבי ממשק, במידה וקיימים מספר אספקטים בהם השחקן מתקשר עם המערכת, השונים באופן מהותי.

- רכיב ממשק יכול לשמש כמעביר מסרים בלבד.



רכיב ישות Entity

- רכיב ישות שומר על המידע של המערכת לאורך זמן.
- פעולות שנהוג למקם ברכיב ישות: קריאה וכתיבה של תכונותיו, פעולות התלויות במצב הרכיב, פעולות יצירה והסרה ופעולות הקשורות לקשרים עם רכיבי ישות אחרים.



רכיב בקרה Control

- מגשר בין רכיבי ממשק לרכיבי ישות. תפקידו לשלוט בביצוע תהליכים.
- רכיב בקרה הוא לעיתים ישות חולפת המתקיימת במשך ביצוע תהליך (use case).
- בתהליך הניתוח בד"כ מוקצה רכיב בקרה אחד לכל use case.