

תכנות מונחה עצמים

בשפת C++

אוהד ברזילי
אוניברסיטת תל אביב

טיפוסי נתונים מופשטים (Abstract Data Types)

ההרצאה מבוססת על מצגת של פרופ' עמירם יהודאי ע"פ הספר
Object-Oriented Software Construction,
.2nd edition, by Bertrand Meyer (Prentice Hall)
כל הזכויות שמורות למחברים

מבנה של מערכות תוכנה

- זיהוי מרכיבי המערכת (מודולים)
- המשולש: מעבדים – פעולות – ישויות
- פירוק top-down פונקציונלי מצוין לתאור אלגוריתמים
- אינו מתאים למערכות גדולות שיתוחזקו לאורך זמן:
 - פונקציה ראשית (מהו ה-top)
 - בעיית הרציפות
 - חסרה תמיכה בשימוש חוזר

מבנה של מערכות תוכנה

□ פירוק מבוסס עצמים ומחלקות מאידך, עונה על התכונות הבאות:

- יכולת הרחבה
- יציבות
- שימוש חוזר
- בעל תאימות גבוהה



בניית תוכנה מונחית עצמים (OOSC)

□ OOSC היא שיטה לפיה תוכנה מאורגנת ברכיבים, אשר הוסקו מטיפוסי העצמים שבהם היא מטפלת

□ מפתחות:

■ מציאת טיפוסי העצמים הרלוונטיים

■ תאור טיפוסי העצמים

■ היחסים והמכנים המשותפים שבין העצמים

■ איך הטיפוסיים מגדירים את מבנה התוכנה

מציאת העצמים

- מרחב הבעיה ומרחב הפתרון
- הדמייה (simula 67)
- עצמים ושמות עצם בשפה טבעית (חיישן, מכונית, משכורת, אדם)
- שימוש חוזר בעצמים קיימים
- דרוש נסיון



תאור של מערכות מבוססות עצמים

□ תאור העצמים וטיפוסיהם:

■ התאור אינו תלוי בייצוג (מימוש)

■ פונקציות בדלת האחורית

□ תאור היחסים בין העצמים:

■ יחס של ספק-לקוח (client-server או client-supplier)

■ יחס ירושה

תאור של עצמים

□ תאור כזה צריך להיות:

- מדויק ושאינו משתמע לשתי פנים
- מלא (או מלא כרצוננו)
- לא מפורט מדי

כדי להחביא את המימוש נתאר את העצמים על פי פעולותיהם (פונקציות) כיצורים מתמטיים טהורים:

טיפוס נתונים מופשט (*ADT*)

טיפוס נתונים מופשט

- תאור המכיל את המרכיבים הבאים:
 - טיפוסים (או הכללה שלהם)
 - פונקציות
 - תנאי קדם (precondition)
- פונקציה עם תנאי קדם היא פונקציה חלקית
 - אקסיומות
- התאור לא יכיל מימוש
- טיפוס: הינו אוסף של עצמים אשר מאפיינים אותם אותו פונקציות, אקסיומות ותנאי קדם

דוגמא: טיפוס מחסנית

$STACK<T>$ □

(בספרות מופיע בתור: $STACK[G]$)

□ T הוא פרמטר פורמלי כללי. בהגדרת מופע יסופק פרמטר אקטואלי.



□ למשל: $STACK<ACCOUNT>$

או $STACK<STACK<ACCOUNT>>$

פעולות המחסנית

□ **יוצרים** (בנאים, constructors)

■ make (new) – צור מחסנית חדשה ריקה

□ **שאלות** (selectors, accesors, queries)

■ item (top) – החזר את האיבר העליון (אם המחסנית אינה ריקה)

■ empty – האם המחסנית ריקה?

□ **פקודות** (modifiers, transformers, mutators)

■ put (push) הוסף איבר למחסנית

■ remove (pop) – הסר את האיבר העליון (אם המחסנית אינה ריקה)

פעולות המחסנית

- ניתן לסווג את סוג הפעולות על טיפוס נתונים T לפי מיקומו של הטיפוס בהגדרת הפונקציה:
 - **יוצרים** (בנאים, constructors)
 - T מופיע רק מימין לחץ
 - **שאלות** (selectors, accesors, queries)
 - T item מופיע רק משמאל לחץ
 - **פקודות** (modifiers, transformers, mutators)
 - T מופיע גם מימין וגם משמאל לחץ

פונקציות

- *put*: $STACK \langle T \rangle \times T \rightarrow STACK \langle T \rangle$
- *remove*: $STACK \langle T \rangle \rightarrow STACK \langle T \rangle$
- *item*: $STACK \langle T \rangle \rightarrow T$
- *empty*: $STACK \langle T \rangle \rightarrow BOOLEAN$
- *new*: $\rightarrow STACK \langle T \rangle$



אקסיומות

For any $x: T, s: STACK \langle T \rangle$,

A1 • $item (put (s, x)) = x$

A2 • $remove (put (s, x)) = s$

A3 • $empty (new)$

A4 • **not empty** ($put (s, x)$)

תנאי קדם

□ לטיפול בפונקציות חלקיות:

- **@pre: not empty** (*s*)
remove (*STACK*<*T*> *s*)

- **@pre: not empty** (*s*)
item (*STACK*<*T*> *s*)

הסקה על פי אקסיומות

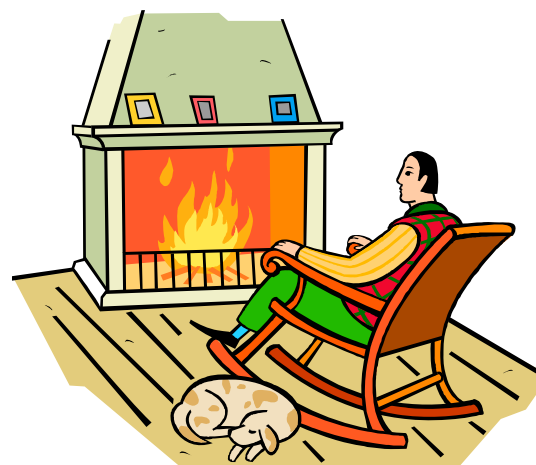
$S1 = new$

$S2 = put(S1, X1)$

$S3 = put(S2, X2)$

$S4 = remove(S3)$

$E2 = item(S4)$



הסקה על פי אקסיומות (2)

$S1 = new$

$S2 = put (S1, X1)$

$S3 = put (S2, X2)$

$S4 = remove (S3) = remove (put (S2, X2)) = S2$

$E2 = item (S4)$



הסקה על פי אקסיומות (3)

$S1 = new$

$S2 = put (S1, X1)$

$S3 = put (S2, X2)$

$S4 = S2$

$E2 = item (S4) = item (S2) = item (put (S1, X1))$
 $= X1$



המחסנית כטיפוס נתונים מופשט

TYPES

- $STACK \langle T \rangle$

FUNCTIONS

- $put: STACK \langle T \rangle \times T \rightarrow STACK \langle T \rangle$
- $remove: STACK \langle T \rangle \rightarrow STACK \langle T \rangle$
- $item: STACK \langle T \rangle \rightarrow T$
- $empty: STACK \langle T \rangle \rightarrow BOOLEAN$
- $new: \rightarrow STACK \langle T \rangle$

המחסנית כטיפוס נתונים מופשט (2)

AXIOMS

For any $x: T, s: STACK [T]$

- A1 • $item (put (s, x)) = x$
- A2 • $remove (put (s, x)) = s$
- A3 • $empty (new)$
- A4 • **not empty** ($put (s, x)$)

PRECONDITIONS

- **@pre:** **not empty** (s)
 $remove (STACK<T> , s)$
- **@pre:** **not empty** (s)
 $item (STACK<T> , s)$



עוד על טיפוס נתונים מופשט

- טיפוס נתונים מופשט הוא מפרט (specification) חופשי משיקולי עיצוב המערכת ומימושה
- לדוגמא: זמן לעומת מקום
- מה המשמעות של שלמות ההגדרה של טיפוס
- מה המשמעות של עקביות (consistency) ההגדרה
- בהקשר הזה נרצה לדבר על שלמות מספקת (sufficient) (completeness)

ביטויי טיפוס נתונים מופשט

- **הגדרה:** ביטוי ADT בנוי כהלכה (well formed ADT) expression) משתמש בפונקציות של טיפוס הנתונים בתחביר נכון (מספר הארגומנטים וסוגם)
- **הגדרה:** הביטוי בנוי כהלכה $f(x_1 \dots x_n)$ יקרא תקין אם כל x_i הם תקינים וערכיהם מספקים את תנאי הקדם של f .
- **דוגמאות:**
 - $push(x, s)$ - תקין
 - $push(x)$ - לא תקין
 - $push(new(), x)$ - לא תקין
 - $pop(new())$ – בנוי כהלכה אך לא תקין

ביטויי טיפוס נתונים מופשט

□ הגדרה: ביטוי שאילתה הוא ביטוי שהפונקציה החיצונית ביותר שלו היא שאילתה

- `empty (put (put (new, x1), x2))`
- `item (put (put (new, x1), x2))`



ביטויי טיפוס נתונים מופשט

□ הגדרה: ביטוי שאילתה הוא ביטוי שהפונקציה החיצונית ביותר שלו היא שאילתה

- `empty (put (put (new, x1), x2)) = false`
- `item (put (put (new, x1), x2)) = x2`



שלמות מספקת

□ טיפוס נתונים מופשט T הוא בעל שלמות מספקת אם אפשר להכריע ע"פ אקסיומות בלבד עבור כל ביטוי בנוי כהלכה e :

■ האם e תקין

■ את ערכו של e (אם הוא ביטוי שאילתה תקין) ללא תלות ב- T

□ דוגמא:

$$\begin{aligned} & \text{top}(\text{pop}(\text{push}(a, \text{push}(b, \text{pop}(\text{push}(c, \text{push}(d, \text{new}())))))))) \\ &= \text{top}(\text{push}(b, \text{push}(d, \text{new}()))) = b \end{aligned}$$

עקביות של טיפוס נתונים מופשט

□ טיפוס נתונים מופשט T הוא עקבי אם אפשר לחשב עבור כל שאילתה בנויה כהלכה e ערך אחד לכל היותר

□ הערות:

■ שלמות מספקת ועקביות הן תכונות משלימות (ערך אחד לכל היותר \ לכל הפחות)

■ בדרך כלל (בעולם "האמיתי") תכונות אלו בלתי כריעות, ואולם בקורס נצליח להוכיח אותם עבור מבחר טיפוסים (כגון STACK)

בניית תוכנה מונחית עצמים (OOSC)

- הגדרה חלופית: OOSC היא שיטה לפיה מערכת תוכנה היא אוסף מובנה של מימושים של טיפוסים נתונים מופשטים (או מופשטים למחצה)



מטיפוסים מופשטים למחלקות

- מחלקה היא מימוש (אולי חלקי) של טיפוס נתונים מופשט
- מחלקה ממומשת במלואה תקרא מחלקה קונקרטית או מחלקה אפקטיבית
- מחלקה הממומשת בחלקה (או שאינה ממומשת כלל) תקרא מחלקה דחויה או מחלקה מופשטת או ממשק של מחלקה (Java)
- למחלקות דחויות תפקיד מפתח בניתוח המערכת ובסיווג הישויות והטיפוסים

מטיפוסים מופשטים למחלקות (2)

- מ ADT למחלקה אפקטיבית: הכלילי את המפרט בהגדרת המחלקה, בחרי ייצוג וממשי כל פונקציה כשגרת מחשב או נתון
- מהגדרה פונקציונלית למימוש אימפרטיבי:
 - הסירי את טיפוס הנתונים המופיע כארגומנט ו\או כערך מוחזר
 - פקודות הן פונקציות המשנות את המצב הפנימי של האובייקט
- אקסיומות ותנאי קדם ישמרו במסגרת המחלקה
 - תיעוד, הערות, כלי עזר

7 צעדים לאושר מונחה עצמים

ע"פ B. Meyer

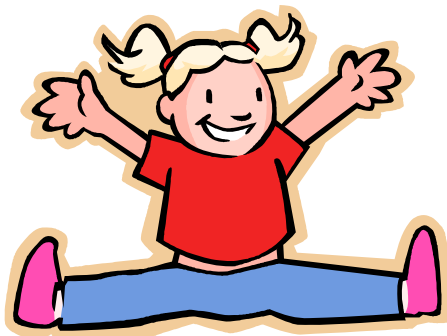
- **מבנה מודולרי מונחה עצמים** – המערכות יחולקו לרכיבים על בסיס מבני הנתונים
- **הפשטה** – אובייקטים הם מימושים של מבני נתונים מופשטים
- **ניהול זכרון אוטומטי** – garbage collection (לא קיים כחלק משפת C++)
- **מחלקות** – כל טיפוס לא פשוט הוא מודול. כל מודול הוא טיפוס



7 צעדים לאושר מונחה עצמים

ע"פ B. Meyer

- ירושה – מחלקה עשויה להיות מוגדרת במונחים של הרחבה או הגבלה של מחלקה קיימת
- רב צורתיות וקישור דינאמי – הפניות מיועדות למגוון של עצמים. לפעולות יש מימושים שונים במחלקות שונות
- ירושה מרובה וירושה חוזרת – מחלקה יכולה לרשת מיותר ממחלקה אחת



מיון של שפות תכנות

ע"פ Peter Wegner

□ שפות מבוססות עצמים – רכיבים עם הסתרת מידע

■ Ada 83, Modula-2

□ שפות מבוססות מחלקות – עצמים + מחלקות

■ CLU

□ שפות מונחות עצמים - עצמים + מחלקות + ירושה

■ Simula, Smalltalk, C++, Eiffel, Ada 95, Java.