חדש חדיש ומחודש

שפת Java לעומת שפת C

Scopes

- בלוק חדש נוצר ע"י: { בלוק חדש }
- ניתן להגדיר משתנים חדשים איפה שרוצים (כמעט)
- טווח הכרה וטווח קיום של משתנים
- כל משתנה יוכר מהנקודה שבה הוגדר ועד לסיום הבלוק *הפנימי* ביותר שבו הוגדר
- להבדיל מ C++ - לא ניתן להגדיר משתנה מקומי בבלוק פנימי אם הוא כבר מוכר בו
- ב Java אין **משתנים גלובלים**. כלומר לא ניתן להגדיר משתנים מחוץ לכל פונקציה, מחלקה או בלוק
- משתנה מקומי (כולל ארגומנט לפונקציה) מסתיר שדה מופע באותו שם (יודגם בהמשך הקורס) – כדי לגשת לשדה יש להשתמש ב `this`

Scopes

```
int x ; // compilation Error! No global x

void f ()
{
  int x ; // local x
  x = 1 ; // assign to local x
  {
      int y = 3;  // local y
      x = 2 ;     // assign to local x
  }
  y = 4 ; // compilation Error! y cannot be resolved
}
```


הגדירי משתנים מקומיים ב scope קטן ככל הניתן

- למשל, במשפט אתחול של `for`:

```
for (int i=0; i<10 ; i++) {...}
```


העמסת פונקציות

- נממש את max המקבלת שני ארגומנטים ומחזירה את הגדול מביניהם
- בשפת C יש לממש max נפרדת (בשם נפרד) עבור כל טיפוס:
  - `double max_double(double x, double y)`
  - `long max_long(long x, long y)`
- העמסת פונקציה מאפשרת הגדרת <u>פונקציות שונות</u> <u>בשם זהה</u> ובלבד שלא תהיה להן אותה חתימה (טיפוס ומספר ארגומנטים)

העמסת פונקציות

- בשפת Java ניתן להגדיר:
  - `double max(double x, double y)`
  - `long max(long x, long y)`
- איזו מהפונקציות תופעל במקרים הבאים:
  - `max(1L , 1L);` `// max(long,long)`
  - `max(1.0 , 1.0);` `// max(double,double)`
  - `max(1L , 1.0);` `// max(double,double)`
  - `max(1 , 1);` `// max(long,long)`

העמסת פונקציות

- המהדר מנסה למצוא את הגרסה המתאימה ביותר עבור כל קריאה לפונקציה על פי טיפוסי הארגומנטים של הקריאה
- אם אין התאמה מדויקת לאף אחת מחתימות המתודה הקיימות, המהדר מנסה המרות (casting) שאינן מאבדות מידע
- אם לא נמצאת התאמה או שנמצאות שתי התאמות "באותה רמת סבירות" או שפרמטרים שונים מתאימים לפונקציות שונות המהדר מודיע על אי בהירות (ambiguity)

שימוש בהעמסה:
ארגומנט עם ערך ברירת מחדל

- פונקצית הספק `int compute(int x)` מבצעת חישוב כלשהו על הארגומנט `x`
- לאחר זמן מה, כאשר הקוד כבר בשימוש במערכת, עלה הצורך לבצע חישוב זה גם בבסיסי ספירה אחרים (החישוב המקורי בוצע בבסיס עשרוני)
- במקום **להחליף** את חתימת הפונקציה להיות `int compute(int x,int base)` **נוסיף** את הפונקציה החדשה כגירסה מועמסת
- כדי לשמור על עקביות שתי הגרסאות הגרסה הישנה תמומש כ- `compute(x,10)`

by value העברה

- בשפת C נתונים מועברים לפונקציה ומוחזרים ומפונקציה ע"י העתקתם (by value)
- גם מצביעים מועתקים!
- לא ניתן לשנות נתון המועבר לפונקציה
- ב Java **נשמרת** שיטת העברה זו
- אף על פי שמצביעים בשפת Java מכונים גם reference (הפנייה או התייחסות) כל העברות הארגומנטים הן by value

טיפוסי נתונים

- בשפת Java קיימים 8 טיפוסי נתונים פרימיטיביים:
  - מספריים:
    - שלמים: **`byte, short, int, long`**
    - עשרוניים: **`float, double`**
  - תווים:
    - בייצוג unicode: **`char`**
  - לוגיים:
    - **`boolean`**
- בשפת Java לא ניתן ליצור מצביעים לנתונים פרימיטיביים
- כל שאר הטיפוסים הם **הרכבות** של טיפוסים פרימיטיביים ומכונים מחלקות (בדומה ל struct בשפת C)
- מערכים ומחרוזות הן מחלקות!

טיפוסי נתונים

- מופע של מחלקה (class) מכונה עצם (object)
- ניתן ליצור מופע של מחלקה אך ורק על ה Heap (הקצאת זיכרון דינאמי)
- ב Java (שלא כמו ב C++) אין הבדל בין המונחים: מצביע, מחוון, pointer, התייחסות, הפנייה, reference ולכן אין משמעות לאופרטור החץ (<-), אופרטור הכתובת (&) ואופרטור ה de-reference (*)
- כדי לפנות לשדה של עצם (struct ב-C) נשתמש תמיד באופרטור הנקודה (.) - דוגמא בהמשך

`const`

- המילה השמורה **`const`** הקיימת בשפת C אינה קיימת בשפת Java
- חלק מהתכונות של **`const`** ניתנות להבעה בעזרת המילה השמורה **`final`** – פרוט בהמשך הקורס

קלט \ פלט

- פלט בעזרת ה"פקודה":

```
System.out.println("Hello World");
```

- הארגומנט לפקודה יכול להיות מכל טיפוס שהוא (פרימיטיבי, מחרוזת או אחר)
- קלט מהמקלדת ב- Java דורש הבנה של ספריית ה Streams של Java וילמד רק בהמשך הקורס

מה חדש?

- הקצאת זכרון דינאמית תעשה בעזרת האופרטור `new` (מילה שמורה) המחליף את `malloc`
- `new` מקבלת כארגומנט שם טיפוס עם סוגריים, מערך או פונקציית אתחול (בנאי)
- לא ניתן ב Java להקצות זיכרון עבור טיפוס פרימיטיבי:

```
int i = new int;        // compilation error!
int[] i = new int[10]; // הקצאת מערך של 10 int
Point p = new Point(); // הקצאת עצם מטיפוס Point
```


לשחרר את ווילי

- שחרור זכרון שהוקצה דינאמית יעשה בצורה אוטומטית ע"י מנגנון ה Garbage Collection
- אין ב Java מקבילה ל `free` של שפת C או `delete` של C++

עבודה עם מערכים

- מערכים הם אוסף נתונים מאותו הטיפוס בעל גודל קבוע מראש (בזמן יצירת המערך)
- אוספי נתונים בגודל משתנה לא יממשו ע"י מערכים אלא ע"י מבנה הנתונים `ArrayList<T>` אשר חוסך למתכנת את ההתמודדות עם ניהול זכרון (אין ב Java פונקציות כדוגמת `realloc()`) הטומן בחובו באגים לעתיד
- מחרוזות (מערכים מטיפוס * char) יוחלפו בטיפוס הסטנדרטי `String`

הערות

- יותר משורה אחת: /* ... */
- עד סוף השורה: //
- בשפת JAVA הוסיפו סוג נוסף - הערות לצורכי תיעוד: /** ... */

הפרה-פרוססור עשה את שלו - הפרה-פרוססור יכול ללכת

- הגדרת קבועים:

```
final static int MAX = 100;
```

- הגדרת סדרת קבועים:

```
enum traffic_lights { RED , YELLOW , GREEN }
```

boolean

- טיפוס חדש לטיפול בערכים בולאנים
- הליטרלים `true` ו- `false` הן חלק מהשפה (מילים שמורות)
- לא ניתן להמיר ערכים בולאנים למספרים שלמים

Keywords

- The following keyword are reserved and cannot be used as identifiers:

abstract	continue	for	new	switch
assert	default	goto	package	synchronized
boolean	do	if	private	this
break	double	implements	protected	throw
byte	else	import	public	throws
case	enum	instanceof	return	transient
catch	extends	int	short	try
char	final	interface	static	void
class	finally	long	strictfp	volatile
const	float	native	super	while