SYNTAX
שפה מייצרת חשיבה

# Lesson 2 – Variables & Operators

הנושאים הנלמדים:

משתנים, אופרטורים, ביטויים, המרות וקבלת קלט

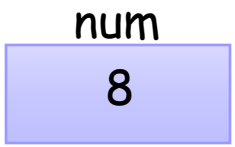
## משתנה – Variable :

- לבקש את שם המשתמש ולאחר מכן לברכו לשלום - "שלום משה" או "שלום דויד", לבקש מספרים ולחבר.
- שמירה בזכרון ושליפה מאוחרת.
- כתובות בזכרון אי אפשר לזכור, שמות כן.
- השם קבוע אבל הערך משתנה.
- לכל משתנה יש שם, טיפוס וערך (הערך יכול להשתנות השאר לא)

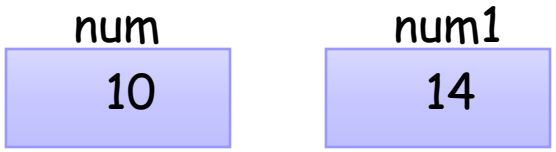
# משתנים - איתחול

```
int num = 8;
```

טיפוס שם ערך




```
int num = 10;
int num1 = 14;
```



# הצהרה והשמה

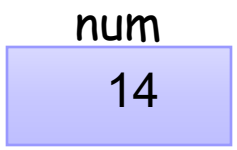
- חלוקה לשני שלבים:

```
int num;
num = 8;
```

- שימוש ב num לפני האיתחול – שגיאת קומפיליציה.

- הסימן = - אופרטור "השמה".
- הביטוי מחושב מימין לשמאל (קודם חישוב ואז השמה).

```
num = 5 + 9;
```



# שימוש במשתנים

```
int num = 3;
int num1 = 5;

num = 8 + 1;
num1 = num + 6;
```

| num | num1 |
|---|---|
| 9 | 15 |

**השמה עובדת מימין לשמאל:**

**1. ראשית מחושב הביטוי בצד ימין.**

**2. שנית, מתבצעת השמה למשתנה בצד השמאלי.**

# שימוש במשתנים

```
public static void main(String[] args) {

    int days = 7; //declaring int variable and initializing it

    System.out.println("A week contains " + days + " days");

    days = 365;

    System.out.println("A year contain " + days + " days");


    int days = 40;//Compile Error! days already declared in this scope
}
```

A week contains 7 days.
A year contains 365 days.

# אופרטורים

- אופרטורים – פעולות, אופרנדים – ערכים, יחד ביטוי.
- הביטוי 4 + 7 מורכב מהאופרטור + הפועל על שני ארגומנטים (4,7) ומחזיר תוצאה (11).
- אופרטור אונרי - ערך אחד (שלילה - !), בינארי - שני ארגומנים (+, -, <), טרנרי - שלושה ארגומנטים (?!).
- ישנם אופרטורים מתמטים (+,-,*,/), אופרטורי השוואה (==, =!, <, >, =<, =>) ואופרטורים לוגיים (&&, ||)

# אופרטורי המתמטיקה

כל האופרטורים של המתמטיקה הם בינארים (מקבלים שני ארגומנטים) והם אותן פעולות שאנו מכירים מהיסודי.

| | |
|---|---|
| חיבור | + |
| חיסור | - |
| כפל | * |
| חילוק | / |
| מודולו | % |

מודולו הוא שארית החלוקה.

מורידים את כמו השלמים שנכנסים ומה שיוצא זה השארית. לדוגמה:

3%2 ← 1

11%3 ← 2

10%5 ← 0

13%7 ← 6

6%7 ← 6

# אופרטורים - דוגמה

```
public static void main(String[] args) {

    int x = 3;

    x = x * 2; //x value is 6

    int y = 2;

    y = 2*x + y; //y value is 14

    System.out.println(y / x);
}
```

x
6

y
14

Output: 2

- int הוא טיפוס של מספרים שלמים בלבד.
- חלוקת שלמים ב Java תמיד תיתן לנו מספר שלם תמיד.
- הטיפוס double (עשרוני) מייצג מספר עשרוני.
- הפתרון הוא בהמרה של אחד המשתנים טרם החלוקה.

# חישוב שעות ודקות

```
public static void main(String[] args) {

    int minutes = 132;

    int hours = minutes / 60;
    int minutesLeft = minutes % 60;

    System.out.println(minutes + " minutes has " + hours +
            " hours, and " + minutesLeft + " minutes");
}
```

132 minutes has 2 hours and 12 minutes

| | Type | Range |
|---|---|---|
| מספרים שלמים | byte | -128 to 127 |
| | short | -32768 to 32767 |
| | int | -2147483648 to 2147483647 |
| | long | -9223372036854775808 to 9223372036854775807 |
| מספרים עם נקודה עשרונית | float | ±1.4E-45 to ±3.4028235E+38 |
| | double | ±4.9E-324 to ±1.79769313486231570E+308 |

# השמה בין טיפוסים שונים

- מה קורה אם מנסים לשים ערך של משתנה מטיפוס אחד במשתנה מטיפוס אחר?
- הדבר מצריך המרה (casting) של הטיפוסים.
- המרה אוטומטית מתבצעת על ידי המחשב כאשר מנסים לעשות השמה מטיפוס צר לרחב (int ל long, float ל double, int ל double וכו')

```
int x = 3;
long y = x;
```

# שלמים ועשרוניים

- double d = 3 / 2

d
1.0

d
1.5

הביטוי בצד ימין מחושב קודם וחלוקת שלמים תמיד תיתן לנו מספר שלם!

- double d = 3.0 / 2

d
1.5

- double d = 3 / 2.0

d
1.5

- double d = (double)3 / 2

casting!!

d
1.0

- double d = (double)(3 / 2)

# השמה בין טיפוסים שונים

- אסור לשים טיפוס "רחב" ב"צר"

```
double d = 3.1
int i = d
```

- אסור לאבד דיוק בלי לקחת אחריות - צריך להצהיר (casting)
- הטיפוסים הם שנבדקים ולא הערך.
- להמיר את הטיפוס לפני ההשמה

```
double d = 3.1
int i = (int)d;
```

ערכו של i הוא 3

# תו - char

- בכדי לייצג תו בודד כלשהו, ישנו הטיפוס char:

```
char c = '$';
```

- ניתן לשרשר תווים למחרוזות, גם הם כמו int עוברים המרה למחרוזת טרם השרשור.

```
System.out.println("Give me 10" + c + ", please");
```

Give me 10$, please

# תו - char

- מאחורי כל תו עומד מספר קבוע ומוגדר מראש.
- למשל התו 'A' שמור תחת המספר 65 והתו 'a' תחת 97.
- לכן כאשר נפעיל את + על שני תווים נקבל מספר('a'+'A'=162)
- בעבר השתמשו בטבלת ה ASSCI היום ב UNICODE

# מחרוזת String

- כדי לייצג יותר מתו בודד נשתמש במחרוזת:

```
String str = "Allot of characters";
```

↑ טיפוס ↑ שם המשתנה ↑ ערך

- מחרוזת היא טיפוס נתונים מיוחד!

- אובייקטים נשמרים באזור שונה בזכרון הנקרא heap בעוד שפרימיטיבים ב stack

# קיצורים

עבור פעולות מאוד נפוצות קיימים קיצורי דרך:

| קיצור | פעולה מלאה |
|---|---|
| x++ | x = x + 1 |
| x-- | x = x -1 |
| x += y | x = x + y |
| x -= y | x = x - y |
| x *= y | x = x * y |
| x /= y | x = x / y |

# קדימות אופרטורים

- נשים לב שיש קדימות של אופרטורים, למשל ההשמה קודמת לקידום וההשגה ולכן:

```
int x = 4;
int y = x++; //the ++ come AFTER the assignment
System.out.println(x); //Prints 5
System.out.println(y); //Prints 4!!

int a = 4;
int b = ++a; //the ++ come BEFORE the assignment
System.out.println(a); //Prints 5
System.out.println(b); //Prints 5!!
```

# קבלת קלט מהמשתמש

- Scanner הוא אובייקט (טיפוס מיוחד) המאפשר לקבל קלט מהמשתמש.
- ראשית יש לייבא אותו לפרוייקט שלנו

import java.util.*; (לרוב זה נעשה אוטומטית עבורנו)

- לאחר מכן נייצר אותו באופן הבא (יוסבר בהמשך הקורס):

**Scanner sc = new Scanner(**System.*in***);**

↑ שם המשתנה (ניתן לתת כל שם חוקי)

- פונקציות שימושיות: sc.next() - > קבלת קלט על הרווח או enter
- sc.nextLine() - > קבלת קלט עד ה Enter
- sc.nextInt(), sc.nextDouble() - > קבלת קלט עד הרווח או enter והמרתו למספר שלם או עשרוני בהתאמה.

# אמירת שלום אישי

```
public static void main(String[] args) {
    Scanner reader = new Scanner(System.in);
    System.out.println("Please enter your name");
    String name = reader.next(); //can use reader.nextLine()
    System.out.println("Hello" + " " + name); //please note the " "
}
```

Input: Moshe
Output: Hello Moshe

# חיבור מספרים

```
public static void main(String[] args) {

    Scanner scanner = new Scanner(System.in);

    System.out.println("Please enter the first number");
    int first  = scanner.nextInt();

    System.out.println("Please enter the second number");
    int second = scanner.nextInt();

    int result = first+second;
    System.out.println(first + " + " + second + " = " + result);
    //OR
    System.out.println(first + " + " + second + " = " + (first+second));

}
```

מה ההבדל ?

Input: 4[space or enter],6

Output: 10

# הדפסת המספר ההפוך

```
public static void main(String[] args) {

    Scanner scanner = new Scanner(System.in);

    System.out.println("Please enter a three digits number");
    int num = scanner.nextInt();

    int units = num % 10; //this gives the units digit
    int tens = num % 100 / 10; //this gives the tens digit
    int hunds = num / 100; //this gives the hundreds digit

    System.out.println("The reverse number is "
            + units + tens + hunds); //the string way

}
```

Input: 561

Output: 165

# בונוס – בסיס בינארי

- ייצוג עשרוני:

$$1452 = 1 \cdot 1000 + 4 \cdot 100 + 5 \cdot 10 + 2 =$$
$$1 \cdot 10^3 + 4 \cdot 10^2 + 5 \cdot 10^1 + 2 \cdot 10^0$$

- ייצוג בינארי:

$$1101 = 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 =$$
$$8 + 4 + 0 + 1 = 13$$

# מעבר מעשרוני לבינארי

- למשל עבור המספר 14:
- מחלקים ב-2, מתקבל 7 עם שארית **0**
- מחלקים ב-2, מתקבל 3 עם שארית **1**
- מחלקים ב-2, מתקבל 1 עם שארית **1**
- מחלקים ב-2, מתקבל 0 עם שארית **1**

- **הייצוג הבינארי הוא 1110**