

# תוכנה 1 שייעור 1

סיון טולדו

```
public class Lecture1Program1 {  
    public static void main(String[] args) {  
        System.out.println("Hello");  
        System.out.println("Software");  
        System.out.println(1);  
    }  
}
```

```
public class Lecture1Program1 {  
    public static void main(String[] args) {  
        System.out.println("Hello");  
        System.out.println("Software");  
        System.out.println(1);  
    }  
}
```

(זה ג'אוה)

```
public class Lecture1Program1 {  
    public static void main(String[] args) {  
        System.out.println("Hello");  
        System.out.println("Software");  
        System.out.println(1);  
    }  
}
```

```
Hello  
Software  
1
```

```
public class Lecture1Program1 {  
    public static void main(String[] args) {  
        System.out.println("Hello");  
        System.out.println("Software");  
        System.out.println(1);  
    }  
}
```

החלקים **באדום** הם תבנית שבשלב הזה נשתמש בה כקבועה. בהמשך נלמד את המשמעות של כל חלק בה. ובכל זאת: `main` היא פונקציה שמערכת ההפעלה מריצה אותה כשמריצים את הקובץ.

```
public class Lecture1Program1 {  
    public static void main(String[] args) {  
        System.out.println("Hello");  
        System.out.println("Software");  
        System.out.println(1);  
    }  
}
```

שתי מחרוזות (טקסט שמופיע בגוף התוכנית)  
ושלם אחד. קבועים שמופיעים בתוכנית נקראים  
ליטרלים (literals).

```
public class Lecture1Program1 {  
    public static void main(String[] args) {  
        System.out.println("Hello");  
        System.out.println("Software");  
        System.out.println(1);  
    }  
}
```

משתנה גלובלי (זמין בכל מקום בתוכנית)  
שמתייחס לעצם שמייצג חלון או קובץ  
שאפשר להדפיס לתוכו טקסט.

```
public class Lecture1Program1 {  
    public static void main(String[] args) {  
        System.out.println("Hello");  
        System.out.println("Software");  
        System.out.println(1);  
    }  
}
```

שירות של העצם שמייצג חלון/קובץ טקסט  
שמבצע הדפסה של שורה אחת ועובר לשורה  
הבאה (קיצור של print line).  
שירות הוא סוג של פונקציה, אבל פונקציה  
שהיא חלק מעצם.



```
public class Lecture1Program2 {  
    public static void main(String[] args) {  
        Integer x = 4;  
        Integer y;  
        y = 5;  
        Integer z;  
        z = x + y;  
        System.out.println(z);  
        System.out.println(x+y+z);  
    }  
}
```

תוכנית חדשה (מספר 2).

שלושה משתנים,  $x$ ,  $y$ ,  $z$ , שלושתם מטיפוס `Integer`.  
בג'אווה למשתנים יש טיפוס (`type`). חייבים להגדיר  
משתנים ואת הטיפוס שלהם לפני השימוש.

```
public class Lecture1Program2 {  
    public static void main(String[] args) {  
        Integer x = 4;  
        Integer y;  
        y = 5;  
        Integer z;  
        z = x + y;  
        System.out.println(z);  
        System.out.println(x+y+z);  
    }  
}
```

השמת ערך למשתנה.

אפשר לבצע השמה במשפט ההגדרה (x).

```
public class Lecture1Program2 {  
    public static void main(String[] args) {  
        Integer x = 4;  
        Integer y;  
        // y = 5;  
        Integer z;  
        z = x + y;  
        System.out.println(z);  
        System.out.println(x+y+z);  
    }  
}
```

חייבים לבצע השמה לפני שמשתמשים במשתנה.  
התוכנית לא תקינה כי ההשמה ל- $y$  היא בהערה.

נקבל הודעת שגיאה

the local variable  $y$  may not have been initialized

```
public class Lecture1Program2 {  
    public static void main(String[] args) {  
        Integer x = 4;  
        Integer y;  
        // y = 5;  
        Integer z;  
        z = x + y;  
        System.out.println(z);  
        System.out.println(x+y+z);  
    }  
}
```

חייבים לבצע השמה לפני שמשתמשים במשתנה.  
התוכנית לא תקינה כי ההשמה ל- $y$  היא בהערה.

נקבל הודעת שגיאה

the local variable  $y$  may not have been initialized

```
public class Lecture1Program3 {
```

```
    public static
```

```
    Integer plusone(Integer x) {
```

```
        return x+1;
```

```
    }
```

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

```
        Integer x = 4;
```

```
        Integer y = plusone(x);
```

```
        System.out.println(x);
```

```
        System.out.println(y);
```

```
    }
```

```
}
```

הגדרה של פונקציה שמקבלת ומחזירה Integer.

מילת המפתח static אומרת שזו פונקציה ולא

שירות של עצם. על ההבדל נדון בהמשך הקורס.

```
public class Lecture1Program3 {
```

```
    public static
```

```
    Integer plusone(Integer x) {
```

```
        return x+1;
```

```
    }
```

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

```
        Integer x = 4;
```

```
        Integer y = plusone(x);
```

```
        System.out.println(x);
```

```
        System.out.println(y);
```

```
    }
```

```
}
```

ארגומנטים של פונקציה הם משתנים מקומיים  
והשם שלהם הוא מקומי; אין להם קשר למשתנים  
אחרים עם אותו שם בתוכנית.

```
public class Lecture1Program5 {  
  
    public static void modify(StringBuffer s) {  
        s.append("-Aviv");  
    }  
  
    public static void main(String[] args) {  
        StringBuffer s = new StringBuffer("Tel");  
        modify(s);  
        System.out.println(s);  
    }  
}
```

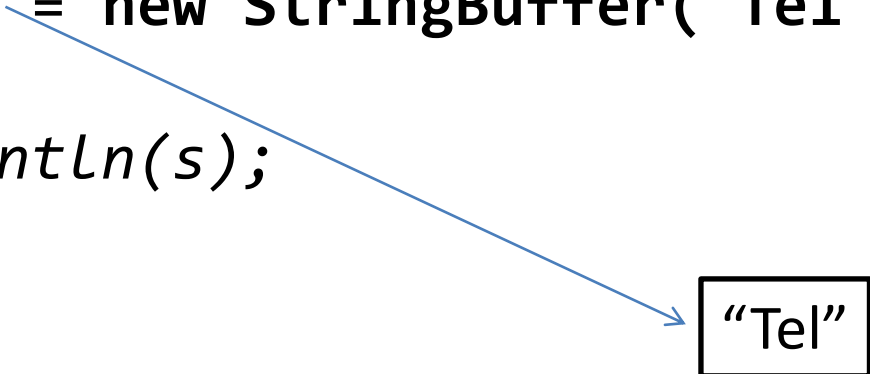
משתנה s שמתייחס לעצם חדש מטיפוס StringBuffer (מחרוזת שאפשר לשנות) שמכיל בהתחלה Tel.

```
public class Lecture1Program5 {  
  
    public static void modify(StringBuffer s) {  
        s.append("-Aviv");  
    }  
  
    public static void main(String[] args) {  
        StringBuffer s = new StringBuffer("Tel");  
        modify(s);  
        System.out.println(s);  
    }  
}
```

הפרוצדורה modify מפעילה על העצם s מתייחס אליו את השירות append שמוסיף למחרוזת. מה יודפס?



```
public class Lecture1Program5 {  
  
    public static void modify(StringBuffer s) {  
        s.append("-Aviv");  
    }  
  
    public static void main(String[] args) {  
        StringBuffer s = new StringBuffer("Tel");  
        → modify(s);  
        System.out.println(s);  
    }  
}
```



A blue arrow originates from the variable `s` in the line `StringBuffer s = new StringBuffer("Tel");` and points to a rectangular box containing the text `"Tel"`.

```
public class Lecture1Program5 {  
  
    public static void modify(StringBuffer s) {  
→      s.append("-Aviv");  
    }  
  
    public static void main(String[] args) {  
        StringBuffer s = new StringBuffer("Tel");  
        modify(s);  
        System.out.println(s);  
    }  
}
```

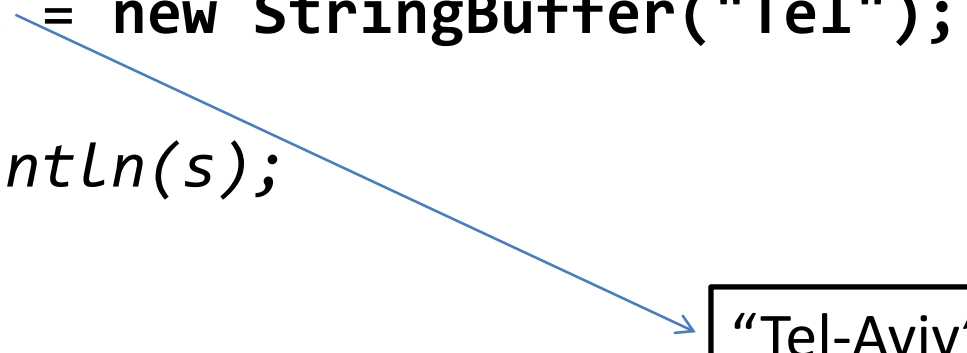
Diagram illustrating the flow of data in the code:

- The `main` method creates a `StringBuffer` object `s` with the initial value `"Tel"`.
- The `main` method calls the `modify` method, passing `s` as an argument.
- The `modify` method receives `s` as a parameter and appends `"-Aviv"` to it.

```
public class Lecture1Program5 {  
  
    public static void modify(StringBuffer s) {  
        s.append("-Aviv");  
    }  
  
    public static void main(String[] args) {  
        StringBuffer s = new StringBuffer("Tel");  
        modify(s);  
        System.out.println(s);  
    }  
}
```

Diagram illustrating the execution flow of the code. The `main` method calls `modify(s)`, passing the `s` variable. The `modify` method then appends `"-Aviv"` to the `s` object. The final output is `"Tel-Aviv"`.

```
public class Lecture1Program5 {  
  
    public static void modify(StringBuffer s) {  
        s.append("-Aviv");  
    }  
  
    public static void main(String[] args) {  
        StringBuffer s = new StringBuffer("Tel");  
        modify(s);  
        System.out.println(s);  
    }  
}
```



→ "Tel-Aviv"

# Immutable Objects

- בתוכנית האחרונה השתמשנו ב-StringBuffer כי אפשר לשנות עצמים מהטיפוס הזה (למשל להאריך את המחרוזת)
- הטיפוס הנפוץ String הוא immutable; עצם כזה מקבל תוכן (ערך) כשיוצרים אותו ואי אפשר לשנות אותו
- כנ"ל לגבי Integer ודומיו ( Short, Long, Double, Float, Boolean, Character )
- רוב העצמים בתוכניות ג'אווה כן ניתנים לשינוי

```
import java.util.*;
```

```
public class Lecture1Program6 {
```

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

```
        HashSet<Integer> set = new HashSet<Integer>();
```

```
        set.add(1);
```

```
        set.add(2);
```

```
        for (Integer i: set) {
```

```
            System.out.println(i);
```

```
        }
```

```
    }
```

```
}
```

התוכנית משתמשת בטיפוסים

מהספרייה java.util.

```
import java.util.*;

public class Lecture1Program6 {

    public static void main(String[] args) {
        HashSet<Integer> set = new HashSet<Integer>();
        set.add(1);
        set.add(2);
        for (Integer i: set) {
            System.out.println(i);
        }
    }
}
```

טיפוס מהספריה שמייצג קבוצה של משהו-ים,  
כאן קבוצה של Integers.

```
import java.util.*;

public class Lecture1Program6 {

    public static void main(String[] args) {
        HashSet<Integer> set = new HashSet<Integer>();
        set.add(1);
        set.add(2);
        for (Integer i: set) {
            System.out.println(i);
        }
    }
}
```

עצם חדש שמייצג קבוצה ריקה (של Integers).



```
import java.util.*;

public class Lecture1Program6 {

    public static void main(String[] args) {
        HashSet<Integer> set = new HashSet<Integer>();
        set.add(1);
        set.add(2);
        for (Integer i: set) {
            System.out.println(i);
        }
    }
}
```

לולאה בתחביר מיוחד לקבוצות שעוברת על כל  
איברי הקבוצה. זה תחביר מקוצר ל...

```
import java.util.*;

public class Lecture1Program7 {

    public static void main(String[] args) {
        HashSet<Integer> set = new HashSet<Integer>();
        set.add(1);
        set.add(2);
        Iterator<Integer> iter = set.iterator();
        while (iter.hasNext()) {
            System.out.println(iter.next());
        }
    }
}
```

תחביר שמתמש בשירותים של HashSet ושל  
Iterator. נחזור לתחביר המקוצר.

```
import java.util.*;

public class Lecture1Program8 {

    public static void modify(HashSet<Integer> s) {
        s.add(7);
    }

    public static void main(String[] args) {
        HashSet<Integer> set = new HashSet<Integer>();
        set.add(1);
        modify(set);
        for (Integer i: set) {
            System.out.println(i);
        }
    }
}
```

מה יודפס?

```
import java.util.*;

public class Lecture1Program8 {

    public static void modify(HashSet<Integer> s) {
        s.add(7);
    }

    public static void main(String[] args) {
        HashSet<Integer> set = new HashSet<Integer>();
        set.add(1);
        modify(set);
        for (Integer i: set) {
            System.out.println(i);
        }
    }
}
```

מה יודפס?

```
import java.util.*;

public class Lecture1Program9 {

    public static void modify(HashSet<Integer> s) {
        s = new HashSet<Integer>();
        s.add(7);
    }

    public static void main(String[] args) {
        HashSet<Integer> set = new HashSet<Integer>();
        set.add(1);
        modify(set);
        for (Integer i: set) {
            System.out.println(i);
        }
    }
}
```

ועכשיו?

# משתנים והתייחסויות

- משתנים הם שמות בתוך פונקציות/פרוצדורות/שירותים
- כל המשתנים שראינו עד עכשיו הם התייחסויות. הם מצביעים לעצם אבל לא מכילים אותו
- כאשר מעבירים משתנה כארגומנט או מעתיקים אותו למשתנה אחר (השמה), נוצרת עוד הצבעה לאותו עצם
- השמה או העברה כארגומנט יוצרת עותק של ההתייחסות, לא של העצם

```
public class Lecture1Program10 {  
  
    public static void main(String[] args) {  
        Integer[] array = new Integer[3];  
        array[0] = 1;  
        array[1] = 22;  
        array[2] = 333;  
        for (Integer i: array) {  
            System.out.println(i);  
        }  
    }  
}
```

המשתנה array מתייחס למערך בגודל 3.

מערך: עצם פשוט שממפה שלמים בין 0 לגודל המערך פחות 1 למשתנים. המערך הזה ממפה למשתנים שמתייחסים כולם ל-Integer. תחביר מיוחד.

```
public class Lecture1Program10 {  
  
    public static void main(String[] args) {  
        Integer[] array = new Integer[3];  
        array[0] = 1;  
        array[1] = 22;  
        array[2] = 333;  
        for (Integer i: array) {  
            System.out.println(i);  
        }  
    }  
}
```

המשתנה array מתייחס למערך בגודל 3.

מערך: עצם פשוט שממפה שלמים בין 0 לגודל המערך פחות 1 למשתנים. המערך הזה ממפה למשתנים שמתייחסים כולם ל-Integer. תחביר מיוחד.



```
public class Lecture1Program11 {  
  
    public static void main(String[] args) {  
        Integer[] array = new Integer[3];  
        array[0] = 1;  
        array[1] = 22;  
        array[2] = 333;  
        for (Integer index=0; index<array.length; index++)  
        {  
            System.out.println( array[ index ] );  
        }  
    }  
}
```

תחביר חלופי ללולאה. יותר מסובך אבל יותר  
גמיש. שימו לב להגדרת המשתנה index בתוך  
הסוגריים של הלולאה.

```
public class Lecture1Program10 {  
  
    public static void main(String[] args) {  
        Integer[] array = new Integer[3];  
        array[0] = 1;  
        array[1] = 22;  
        // array[2] = 333;  
        for (Integer i: array) {  
            System.out.println(i);  
        }  
    }  
}
```

מה יודפס? (ההשמה לאיבר השלישי בהערה)

```
public class Lecture1Program10 {  
  
    public static void main(String[] args) {  
        Integer[] array = new Integer[3];  
        array[0] = 1;  
        array[1] = 22;  
        // array[2] = 333;  
        for (Integer i: array) {  
            System.out.println(i);  
        }  
    }  
}
```

הערך null מייצג התייחסות שאינה מצביעה  
לשום דבר. כל התייחסות יכולה "להכיל" null.

1

2

null

```
public class Lecture1Program10 {  
  
    public static void main(String[] args) {  
        Integer[] array = new Integer[3];  
        array[0] = 1;  
        array[1] = 22;  
        array[2] = null;  
        for (Integer i: array) {  
            System.out.println(i);  
        }  
    }  
}
```

אותה תוצאה בדיוק. אברי מערך מאותחלים ל-null, אבל אפשר לעשות השמה של null.

```
1  
2  
null
```

```
public class Lecture1Program10 {  
  
    public static void main(String[] args) {  
        Integer[] array = new Integer[3];  
        array[0] = 1;  
        array[1] = 22;  
        array      = new Integer[2];  
        for (Integer i: array) {  
            System.out.println(i);  
        }  
    }  
}
```

מה יודפס?

```

public class Lecture1Program10 {

    public static void main(String[] args) {
        Integer[] array = new Integer[3];
        array[0] = 1;
        array[1] = 22;
        array      = new Integer[2];
        for (Integer i: array) {
            System.out.println(i);
        }
    }
}

```

התוכן של המערך החדש בגודל 2. array הוא

משתנה שמתייחס למערך, אבל הוא יכול

null

null

להתייחס למערכים שונים בזמנים שונים.

# משתנים פרימיטיביים

- ג'אווה היא שפה מתוחכמת, אבל עם שרידים של פרימיטביות
- אחד השרידים הוא משתנים פרימיטיביים מהטיפוסים int, byte, short, long, char, double, float, boolean (בלי אות גדולה בהתחלה)
- משתנים כאלה מכילים ערך ולא מצביעים עליו
- השימוש בהם נפוץ מאוד
- הם עשויים לשפר ביצועים
- אבל יש למשתנים הללו התנהגות שונה!

```
public class Lecture1Program13 {
```

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

```
        Integer i1, i2, i3;
```

```
        int      i4, i5, i6;
```

```
        i1 = new Integer(4);
```

```
        i2 = new Integer(4);
```

```
        i3 = i2;
```

```
        i4 = 4;
```

```
        i5 = 4;
```

```
        i6 = i5;
```

```
        System.out.println( i1 == i2 );
```

```
        System.out.println( i2 == i3 );
```

```
        System.out.println( i4 == i5 );
```

```
        System.out.println( i5 == i6 );
```

```
    }
```

```
}
```

מה יודפס?



```
public class Lecture1Program13 {
```

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

```
        Integer i1, i2, i3;
```

```
        int     i4, i5, i6;
```

```
        i1 = new Integer(4);
```

```
        i2 = new Integer(4);
```

```
        i3 = i2;
```

```
        i4 = 4;
```

```
        i5 = 4;
```

```
        i6 = i5;
```

```
        System.out.println( i1 == i2 ); false
```

```
        System.out.println( i2 == i3 ); true
```

```
        System.out.println( i4 == i5 ); true
```

```
        System.out.println( i5 == i6 ); true
```

```
    }
```

```
}
```

מה יודפס?

```
public class Lecture1Program12 {
```

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

```
        String s1, s2, s3;
```

```
        s1 = new String("Tel");
```

תרגיל דומה,

```
        s2 = new String("Tel");
```

עם מחרוזות

```
        s3 = s2;
```

```
        System.out.println( s1 == s2 );
```

```
        System.out.println( s2 == s3 );
```

```
    }
```

```
}
```

```

public class Lecture1Program12 {
    public static void main(String[] args) {
        String s1, s2, s3;
        s1 = new String("Tel");
        s2 = new String("Tel");
        s3 = s2;
        System.out.println( s1 == s2 );           false
        System.out.println( s2 == s3 );           true
    }
}

```

המשתנים s1, s2 מתייחסים לעצמים שונים עם תוכן זהה  
המשתנים s2, s3 מתייחסים לאותו עצם; ההתייחסויות

זהות

```
public class Lecture1Program12 {  
  
    public static void main(String[] args) {  
        String s1, s2, s3;  
        s1 = new String("Tel");  
        s2 = new String("Tel");  
        s3 = s2;  
        System.out.println( s1 == s2 );           false  
        System.out.println( s2 == s3 );           true  
        System.out.println( s1.equals(s2) );      true  
    }  
}
```

כדי לבדוק האם התוכן של שני עצמים זהה, צריך להשתמש בשירות של אחד מהם

# תחנת רענון

מה זו ג'אווה?

על מה הקורס הזה?

## מה זו ג'אווה?

- ג'אווה היא שפת תכנות שפותחה באמצע שנות ה-90 ועברה אבולוציה איטית מאז
- בשימוש נרחב מאוד בתעשייה (למשל רוב אפליקציות אנדרואיד, הרבה תוכנות לשרתים, וכו').
- מבוססת על 3 עקרונות עיקריים
  - תכנות מונחה עצמים (עם לקחים משפות כמו ++C)
  - תוכנות נכתבות לפלטפורמה וירטואלית אחת שיכולה לרוץ על חלונות, לינוקס, מק, וכו'; באנדרואיד הפלטפורמה שונה (ספריות אחרות לגישה למערכת ההפעלה)
  - השפה מסייעת למזעור באגים שמתגלים רק בזמן ריצה (הרבה בדיקות סטאטיות, ניהול זיכרון אוטומטי)

# מטרות הקורס

- לימוד והבנה מעמיקה של רעיונות שמסייעים לפיתוח תוכנות גדולות ומורכבות (בעתיד, לא בקורס)
  - תכנות מונחה עצמים
  - חוזים
  - בדיקות תוכנה לפני ועל ידי הרצה
  - ...
- פיתוח מיומנויות תיכנות טובות בג'אווה
  - שפה עשירה עם ספריות עשירות
  - טוב כי קל לעשות דברים
  - אבל דורש יחסית הרבה לימוד וידע כדי להגיע ליכולות סבירות

# ידע לעומת הבנה

- אי אפשר לתכנת טוב בג'אווה ושפות דומות בלי להבין טוב את העקרונות
- גם אי אפשר בלי להבין את העקרונות של ספריות התוכנה העיקריות ולהכיר את הטיפוסים העיקריים בהן
  - אבל לא צריך לזכור פרטים או טכניקות אזוטריות
  - הפרטים כתובים בתיעוד
  - טכניקות אזוטריות אפשר למצוא ב-stackoverflow וכו'
  - אבל צריך לדעת די הרבה כדי להבין את המקורות הללו

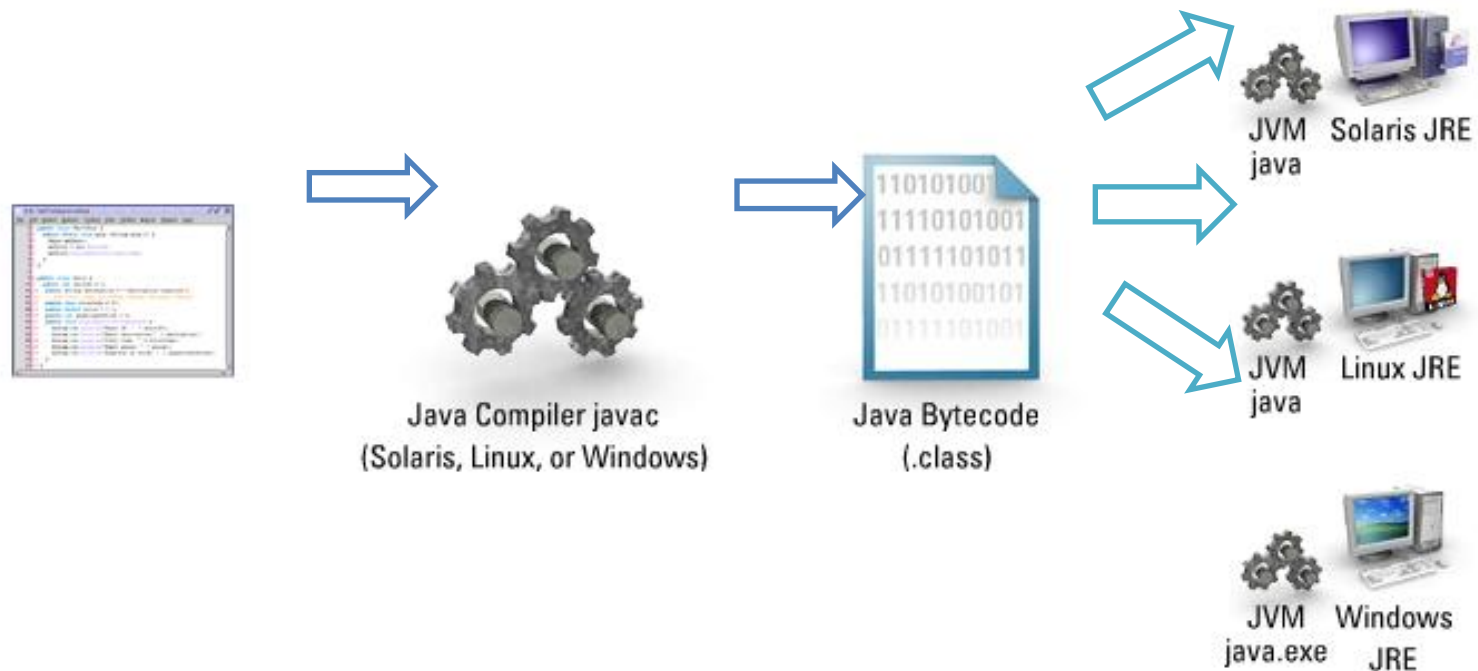


# ג'אווה עבורי

- שפה שאני מתכנת בה הרבה
- מתכנת גם ב-C אבל בעיקר כשאין לי ברירה
- שפה שאני מרגיש בה שאני פרודוקטיבי
- ושכיף לי לתכנת בה: יותר קל להיות יצירתי ובדרך כלל יותר קל להבין קוד (כולל קוד שלי ששכחתי)
- כלי פיתוח מעולים (אקליפס אבל גם סביבות אחרות) שעוזרים לכתוב קוד, למצוא קוד, לשנות קוד (נכון גם לשפות דומות כמו C#, פחות לשפות כמו C)

# הרצת תוכנה בג'אווה

- באקליפס ודומותיה: לחצי על Run; הכי פשוט שאפשר
- מאחורי הקלעים:



# הרצת תוכנה בג'אווה (המשך)

- קובץ קוד המקור Lecture1Program1.java מומר על ידי מהדר (קומפיילר) לקובץ בינרי (לא מיועד לקריאה על ידי בני אדם) בשם Lecture1Program1.class, לפעמים לכמה כאלה
- לעיתים קרובות אורזים קבצי class ביחד במארז עם סיומת jar
- תוכנה שנקראת "מכונת ג'אווה וירטואלית" או JVM מריצה את הפונקציה main שבקובץ ה-Class (או באחד מהם שבתוך jar)

## יבילות (פורטבילות)

- התוכנה של המכונה הוירטואלית ספיציפית למערכת הפעלה מסויימת (לינוקס, חלונות, מק)
- קבצי ה-class וה-jar פועלים בכל מכונה וירטואלית כזאת בכל מערכת הפעלה (בערך)
- לעיתים קרובות אותה תוכנה בדיוק (או כמעט) רצה על כמה מערכות הפעלה שונות
- קובץ class (גם בתוך jar) מאפשר למפתחים לכתוב תוכנה שמשמשת בטיפוסים שמוגדרים בו, לא רק להריץ אותו כמו שהוא; מקל מאוד על פיתוח והפצת ספריות תוכנה

# יוצא מן הכלל

- אפליקציות אנדרואיד מבוססות על ג'אווה
- אבל עם מודל ריצה קצת שונה
- חלק מהספריות שחושפות את היכולות של מערכת ההפעלה (גישה לקבצים וכו') שונות; חלק זהות
- האריזה של יישומים היא שונה
- לא יידון בקורס הזה וגם לא כל כך עקרוני

## ג'אווה ושפות אחרות

- אפשר לקרוא מג'אווה ל-C, בעיקר לצורך קישור לספריות תוכנה קיימות או לשירותים של מערכת ההפעלה שהפלטפורמה הוירטואלית לא תומכת בהם
- יש עוד שפות שמתמשות באותה מכונה וירטואלית בדיוק (ה-JVM) כמו Groovy, Scala, ועוד
  - לכולן יש מאפיינים מסויימים שירשו כמו ג'אווה מה-JVM
  - אבל חלקן שונות מאוד מג'אווה
  - בדרך כלל קל לקרוא להן מגא'ווה ולהיפך
  - נסו ותהנו
- עוד כמה חיבורים מעניינים (למשל לקרוא לגא'ווה מ-Matlab)

# חזרה לחלק הטכני

```
public class Lecture1Program13 {
```

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

```
        i = 7;
```

```
        int j;
```

```
        j = 7;
```

```
        j = 3.14;
```

```
        Integer k = 4;
```

```
        k = 5;
```

```
        k = "Tel-Aviv";
```

```
    }
```

```
}
```

התוכנית לא לגמרי  
תקינה. איפה התקלות?



```
public class Lecture1Program13 {
```

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

```
        i = 7;
```

```
        int j;
```

```
        j = 7;
```

```
        j = 3.14;
```

```
        Integer k = 4;
```

```
        k = 5;
```

```
        k = "Tel-Aviv";
```

```
    }
```

```
}
```

חייבים להגדיר משתנים ואת

הטיפוס שלהם

הם יכולים להכיל או להתייחס

רק לפרימיטיבים/עצמים עם

טיפוס מתאים.

```
public class Lecture1Program13 {
```

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

```
        i = 7;
```

```
        int j;
```

```
        j = 7;
```

```
        j = 3.14;
```

```
        Integer k = 4;
```

```
        k = 5;
```

```
        k = "Tel-Aviv";
```

```
    }
```

```
}
```

האם קומפיילר "חכם" היה יכול

לטפל בתוכנית הזאת בלי להתלונן?

כן, זה בערך מה שקורה

ב-Python, ב-Scala, ועוד

בג'אווה הפילוסופיה היא לדרוש מהתוכניתנית  
להצהיר על הרבה דברים כדי שהקומפיילר ימצא  
חלק מהבאגים לפני שמריצים את התוכנה

```
public class Lecture1Program13 {
```

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

```
        int i;
```

```
        ii = 7;
```

```
        ...
```

```
    }
```

```
}
```

למשל, הקומפיילר של גא'ווה

מוצא באגים כאלה; ב-Python

התוכנית תרוץ, אבל לא תתנהג

נכון כי יש פה באג

# פילוסופיית הטיפוסים של גא'ווה:

## שני עקרונות ראשונים

- **Strong Typing:** לכל משתנה יש טיפוס, והוא יכול להתייחס רק לעצמים מתאימים
  - בהמשך נראה ש-"התאמה" זה מושג מאוד גמיש, אבל העיקרון הוא עדיין עיקרון
  - ב-Python זה דינמי
- **Explicit Typing:** התוכניתן/ית חייבים להצהיר על הטיפוס של משתנה, הקומפיילר לא מניח ולא מנסה להסיק כלום
  - ב-Scala למשל הקומפיילר מסיק את הטיפוס אם אפשר

# שלמים (int, Integer)

```

public class Lecture1Program15 {

    public static void main(String[] args) {
        int i = 1;
        i = -2000000000;           32 סיביות במשלים 2
        i++;                       שתי דרכים לקדם ב-1
        i += 1;
        i = i ^ 7;                 אופרטורים בינריים על
        i = i & 3;                 הייצוג (or ,and ,xor)
        i = i | -34;
        i = Integer.MAX_VALUE;
    }
}

```

# שלמים קצרים וארוכים

Primitive	Object	Literals	Representation
byte	Byte	7	8-bit 2's complement
short	Short	-8	16-bit 2's complement
int	Integer	123, -9;	32-bit 2's complement
long	Long	123L, 1	64-bit 2's complement
char	Character	65, 'א'	16-bit unsigned (~Unicode character)

# הסיפור העצוב על char

- המתכננים של גא'ווה החליטו שתוים ומחרוזות יוכלו לייצג טקסט בכל שפה; רעיון מעולה – גם בגוף התוכנית (בליטרלים; רעיון פחות מעולה)
- הם השתמשו בתקן שנקרא Unicode לייצוג טקסט; גם רעיון מעולה
- באותו זמן כל תווי יוניקוד מופו למספרים של 16 סיביות, וככה הגדירו בג'אווה את התווים
- בהמשך ראו ביוניקוד שזה לא מספיק, הגדירו תוים שממופים למספר מעל  $2^{16}-1$  (האזהרה היתה שם תמיד), וכעת char לא תמיד יכול לייצג תו (int כן)

# מספרים ממשיים מקורבים (float, double)

```
public class Lecture1Program16 {  
  
    public static void main(String[] args) {  
        double x = 3.14;  
        x = -7;  
        x = Math.cos( Math.PI / 8 );  
        x = Double.MAX_VALUE;  
        x = Double.NEGATIVE_INFINITY;  
        x = Double.NaN;  
        // NaN is not a number (Like null or 0/0)  
        float y = 3.14f;  
    }  
}
```



```
public class Lecture1Program17 {  
    public static void main(String[] args) {  
        int    i = 3;  
        long   l = -8;  
        double d = 3.14;  
        float  f = 2.7f;  
        l = i;  
        i = l;  
        d = l;  
        f = l;  
        f = d;  
    }  
}
```

**איזה מההשמות**

**תקינות ואיזה לא?**

**החוקים קצת מסובכים**

```
public class Lecture1Program17 {  
  
    public static void main(String[] args) {  
        int    i = 3;  
        long   l = -8;  
        double d = 3.14;  
        float  f = 2.7f;  
        l = i;  
i = l;  
        d = l;  
        f = l;  
f = d;  
    }  
}
```

The diagram consists of three grey rectangular callout boxes with white text. Blue arrows point from these boxes to specific lines in the code. The first box points to the line ~~i = l;~~. The second box points to the lines d = l; and f = l;. The third box points to the line ~~f = d;~~.

הייצוג תמיד אפשרי

תיתכן שגיאת עיגול! אבל בסדר...

המרה אסורה שעלולה לאבד מידע

```
public class Lecture1Program17 {
```

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

```
        int    i = 3;
```

```
        long   l = -8;
```

```
        double d = 3.14;
```

```
        float  f = 2.7f;
```

```
        l = i;
```

אופרטור ההמרה (cast)

```
        i = (int) l;
```

מרשה לשפה לבצע השמה

```
        d = l;
```

שעלולה לאבד מידע; האופרטור

```
        f = l;
```

```
        f = (float) d;
```

מזהיר שהתוכניתן/ית מודע

```
    }
```

לסכנה ומאשר/ת בכל זאת

```
}
```

## זנב ללימוד עצמי: אופרטורים

```

+ - * / %           // arithmetic
~ & | ^ << >> <<< // bitwise
! && ||           // boolean (short-circuit)
= += -= ...       // assignment, op-assign
?:               // conditional
==              // equality
++ --           /* unary prefix/postfix */

```

- שימו לב לאסוסיאציה (ימין/שמאל)
- שימו לב לקדימות
- שתיהן בדרך כלל אינטואיטבי ( $x+y*z$ ) אבל אם לא בטוחים, צריך סוגריים (לא לבדוק את הכלל)

# זנב ללימוד עצמי: אופרטורים

+ - * / %	// arithmetic
~ &   ^ << >> <<<	// bitwise
! &&	// boolean (short-circuit)
= += -= ...	// assignment, op-assign
?:	// conditional
==	// equality
++ --	/* unary prefix/postfix */

הערה עד סוף שורה

הערה עד שסוגרים אותה

# איך ללמוד?

- הרצאות ותרגולים

- תרגילים

- ספרים ואתרים

– המון על השפה והספריות, חפשו מה מתאים לכם

– מעט על רעיונות גבוהים בהנדסת תוכנה; המומלץ הוא

Program Development in Java מאת Liskov ו-Guttag

(היא חתנת פרס טיורינג!)

– Tutorial מקיף באתר של Oracle

– stackoverflow וכו' לפתרון סוגיות טכניות ספציפיות