

קורס JAVA

מרצה: ד"ר עסאקלה שאדי

לא לשכוח לפתוח את הטלפונים בסוף השיעור

תקשורת במהלך הקורס

• המייל שלי: shadi@braude.ort.org.il (זמינות כל היום)

• מטלות: הגשת המטלות תהיה בתוך האתר.

• עבודה במעבדה – תרגול.

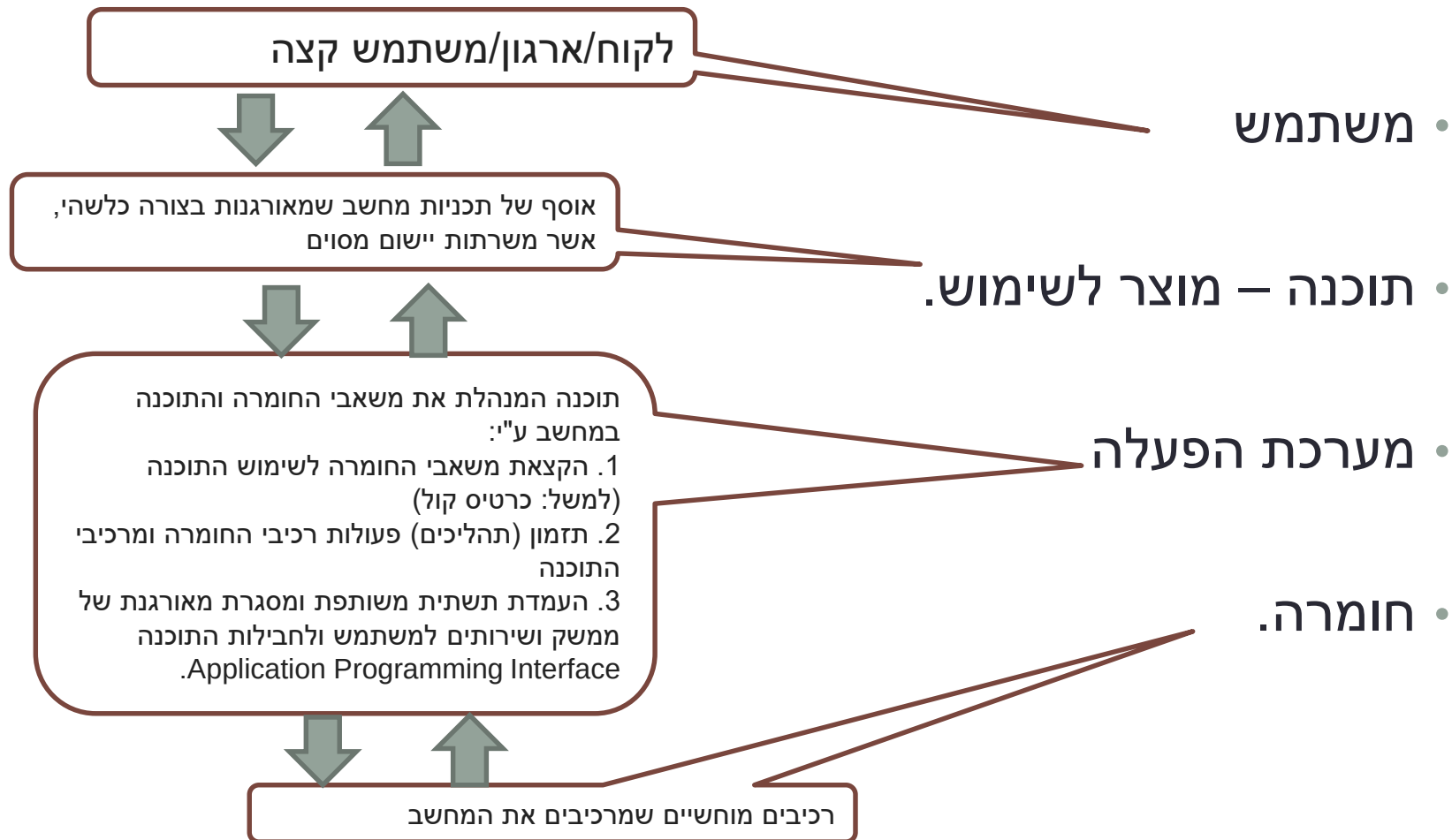
• העלאת מצגות.

מבוא – הנדסת תוכנה

- ענף הנדסת תוכנה - Software Engineering עוסק בפיתוח תוכנה.
- הנדסת תוכנה מתייחסת לכל תהליך הפיתוח בשלמותו. לעומת זאת, תכנות מתמקד בדרך כלל בכתיבת התוכנה.
- בהנדסת תוכנה, תהליך הפיתוח מלווה בתהליכים ארגוניים ועסקיים, לעומת תהליך התכנות שיכול להיות תהליך אישי בלבד.
- מהנדס התוכנה - מומחה לכתיבה ועיצוב תוכנה.
- מהנדס התוכנה מגשר בין הלקוחות (המציאות) לבין המחשב (המערכת) ע"י קיום דו-שיח עם הלקוח ותרגום צרכיו ל- "שפה" פורמאלית בתחום המחשוב.

מבוא – מושגים ששמענו!!!

- בעולם התוכנה יש מושגי יסוד רבים, אבל השחקנים המרכזיים הם:



מבוא

- מערכת – הגדרה

מרכיבים אשר פועלים במשותף לצורך השגת מטרה (או יותר) מוגדרת מראש. כל רכיב יכול להיות מערכת – תהליך רקורסיבי. מערכת ממוחשבת – מורכבת מחומרה ותוכנה.

תוכנה לא עובדת ללא חומרה!!!!

- מערכת עתירת תוכנה (SIS - System Intensive Software) - מערכות תוכנה Software systems – הצורך ודוגמאות.
- מערכות מורכבות – חלוקה ליחידות שכל אחת אחראית על מימוש משלה, עם מינימום תלות זו בזו.
- תכנות מונחה עצמים OOP – הנחת יסוד: ביצוע במשימה של המערכת המורכבת ניתן לחלוקה בין ישויות קטנות יותר. כל ישות נקראת "עצם" (Object) – **יורחב בהמשך.**
- תכניות גדולות מכילות בדרך כלל מאות אלפי שורות קוד. תכניות אלה מטפלות בבעיות רבות ומגוון רב של נושאים. דוגמא: מערכות לניהול מוסד לימודים, מערכות לניהול בנקים, משחקי מחשב ועוד.

מערכות תוכנה – תכנון

- מה זה מוצר תוכנה?
- צורות של מוצר תוכנה: מוצר עצמי, מוצר משוכן (משובץ) שמגיע עם רכיב חומרה, מוצר Software as a service – saas שהרוב הם מוצרי רשת כמו Office 365.
- תכנון בניית המוצר: רעיון – פיתוח – מוצר שעונה על הצורך – תהליך אחזקה – תיקונים ושדרוגים – מה הלאה?
- תהליך הבנייה: צורך – ארכיטקטורה וניתוח תרחישי פעולה – איזה ממשקים ומודולים – תהליך הבנייה.
- איפה אתם כמתכנתים רואים את עצמכם?

מערכות תוכנה – דרישות יסוד

- מפרט – מכיל פירוט לצורך בבניית התוכנה, תפקיד המערכת ואיך היא אמורה לפעול. המפרט מתעדכן בדרך כלל בשלב הפיתוח. בסוף הוא יכיל את כל התיעוד כולל דרישות ההתקנה וההפעלה וכו'. המפרט בדרך כלל נכלל בחוזה הסופי, והוא מהווה חוזה בעצמו.
- נכונות של המערכת – המערכת המתוארת במפרט צריכה להיות נכונה ומתנהגת בהתאם למוזכר בו ומותאמת למטרתה. בדרך כלל במערכות תוכנה גדולות מתגלים באגים והתנהגויות לא לפי המפרט.
- עמידות במצבי קצה מעמיסים. למשל בניית מערכת לניהול קניות, כמה קניות ניתן לבצע ביחידת זמן מסוימת?
- ממשק ידידותי.
- תגובה בזמן מינימלי
- תחזוקה – הוספה/עדכון

מערכות תוכנה

- רוב המערכות המפותחות כיום בעולם הן עתירות תוכנה
- מערכות תוכנה נמצאות:
 - ברמת הרכיב/המכלול, ברמת המערכת (סנכרון בין רכיבים שמרכיבים את המערכת), ברמת המערך (System of System) - פעילות משותפת בין מערכות שונות.
- הנדסת (כולל תכנון ובנייה) מערכות עתירות תוכנה נמצאת בשני מישורים מרכזיים:
 - פיתוח מערכת תוכנה עם התחשבות במקומה במערכת הכוללת
 - פיתוח מערכת תוכנה עם התחשבות בהגדרות ומאפייני מערכת התוכנה עצמה.

מערכות תוכנה – בנייה ואמינות

- למה זה חשוב? מערכות שקרסו עקב תקלות, בנייה מחודשת ועלויות תיקון. מה זה באג? איפה הוא נמצא? ולמה הוא נוצר?
- היבט וסקירה של כל ארכיטקטורת התוכנה. הפשטה לרכיבים, היחס בין הרכיבים והתקשורת.
- אמינות, לחזות תקלות, לצמצם למינימום. קיום תקנים לחיזוי.
- גמישות של תוכנה. ניתנת לשינוי (עלות נמוכה – רק פיתוח)
- מהו תפקיד המתכנת?
- חלוקה לתכניות, מינימום פגיעה מערכתית. תכנות מודולרי – בהמשך.
- תכנון מעמיק של שלב הפיתוח.

שלב הפיתוח – כתיבת הקוד

- בקיאות בשפה
- היכרות עם סביבת הפיתוח
- שימוש נכון במשתנים
- היכרות עם דרכי הפעולה במלואם – בחירת המודל המתאים לפי המערכת העתידה.
- תיעוד – **חשוב ביותר (דרישת חובה לקורס)**
- עבודה בצוות – שילוב בין מחלקות
- אופטימיזציה בבניית הקוד.
- מה פתוח ללקוח ומה סגור? יורחב בהמשך

מהי JAVA

- דבר ראשון - "אנחנו אוהבים java" היא השפה הטובה ביותר בעולם!! 😊
- הייחודיות של השפה:
- ג'אווה הינה שפת תכנות מוכוונת-עצמים (Object Oriented), זו שפה עילית שפותחה על ידי חברת SUN.
- מטרתם המקורית של מפתחי השפה הייתה ליצור שפת תכנות אשר תשמש לפיתוח תכניות להפעלת מכשירי חשמל. בשנת 95 עם התפתחות ה-web השפה החלה לצבור תאוצה הודות לאפשרות לכתיבת applets.

מהי JAVA

המוטו של השפה:

Write Once –Run Anywhere !

אי תלות בפלטפורמה שעליה התכנית רצה

- המטרה העיקרית שעמדה לנגד עיניהם של מפתחי השפה הייתה ליצור שפה אשר מאפשרת לכתוב את התכנית פעם אחת ולאחר מכן להריץ אותה בכל מחשב, מבלי לבצע שינויים.

איך זה עובד בעצם?

- את הקוד שנכתב בשפת java מריץ מפרש (interpreter).
- חסרונות:
 - מאט את מהירות הריצה.
 - טעויות מתגלות רק בזמן ריצה.
- לצורך כך הוסיפו ב java שלב נוסף, הידור (קימפול *compilation*)
- המהדר מבצע עיבוד מקדים של קוד התוכנית (קובץ טקסט) ויוצר קובץ חדש שנקרא byte code והסיומת שלו היא .class.
- בתהליך הקומפילציה נבדק התחביר של הקוד והשגיאות המתגלות מדווחות למתכנת.

המכונה המדומה

Java Virtual Machine

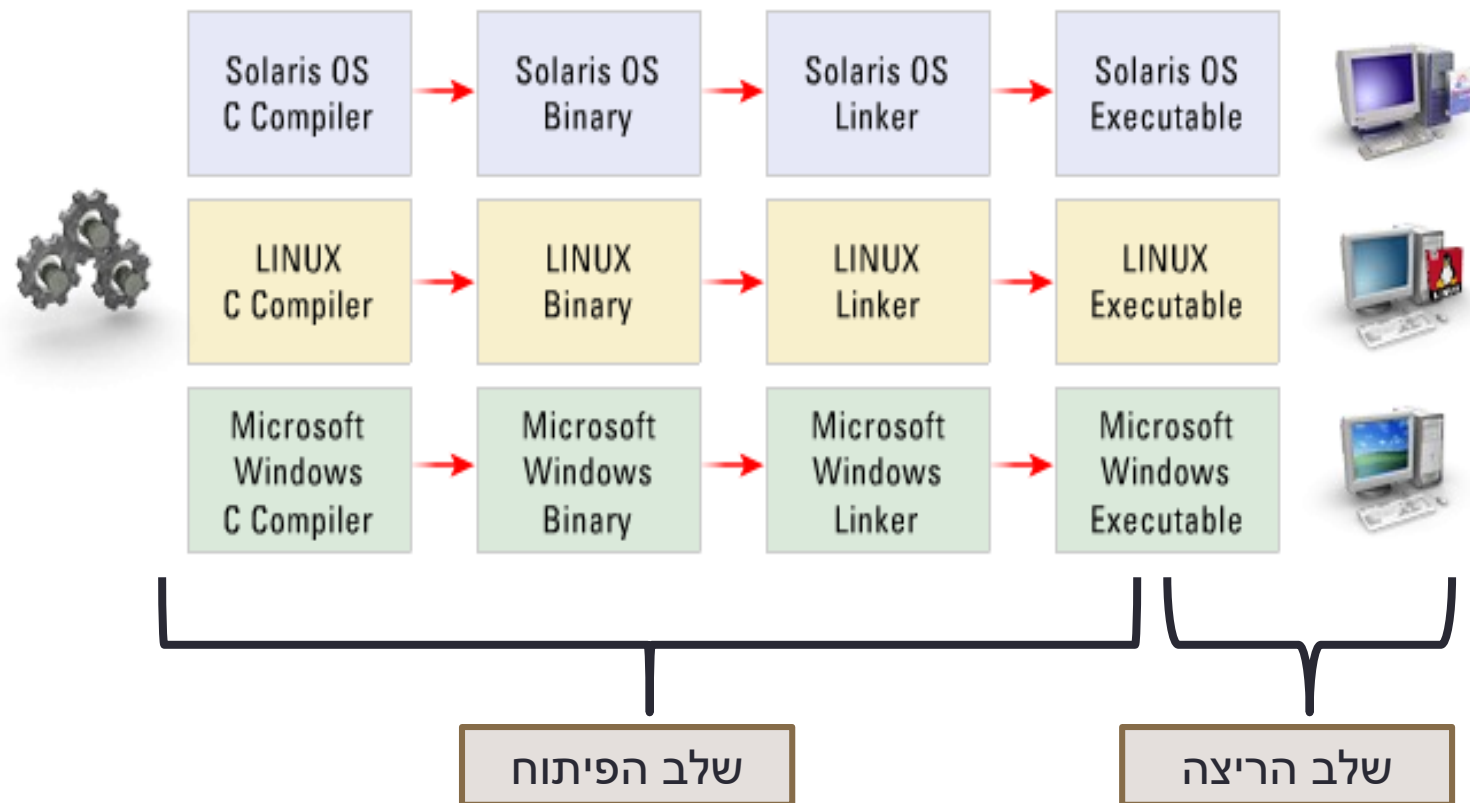
- הקובץ המהודר מכיל הוראות ריצה ב"מחשב כללי" ולא במחשב ספציפי או פלטפורמה ספציפית.
- עבור כל סביבה (פלטפורמה) נכתב מפרש מיוחד שיודע לבצע את התרגום מהמחשב הכללי, המדומה, למחשב המסוים שעליו מתבצעת הריצה.
- המפרש נעשה ע"י ספקי תוכנה, שזה תפקידם, עבור רוב סביבות הריצה הנפוצות.

Write Once –Run Anywhere !

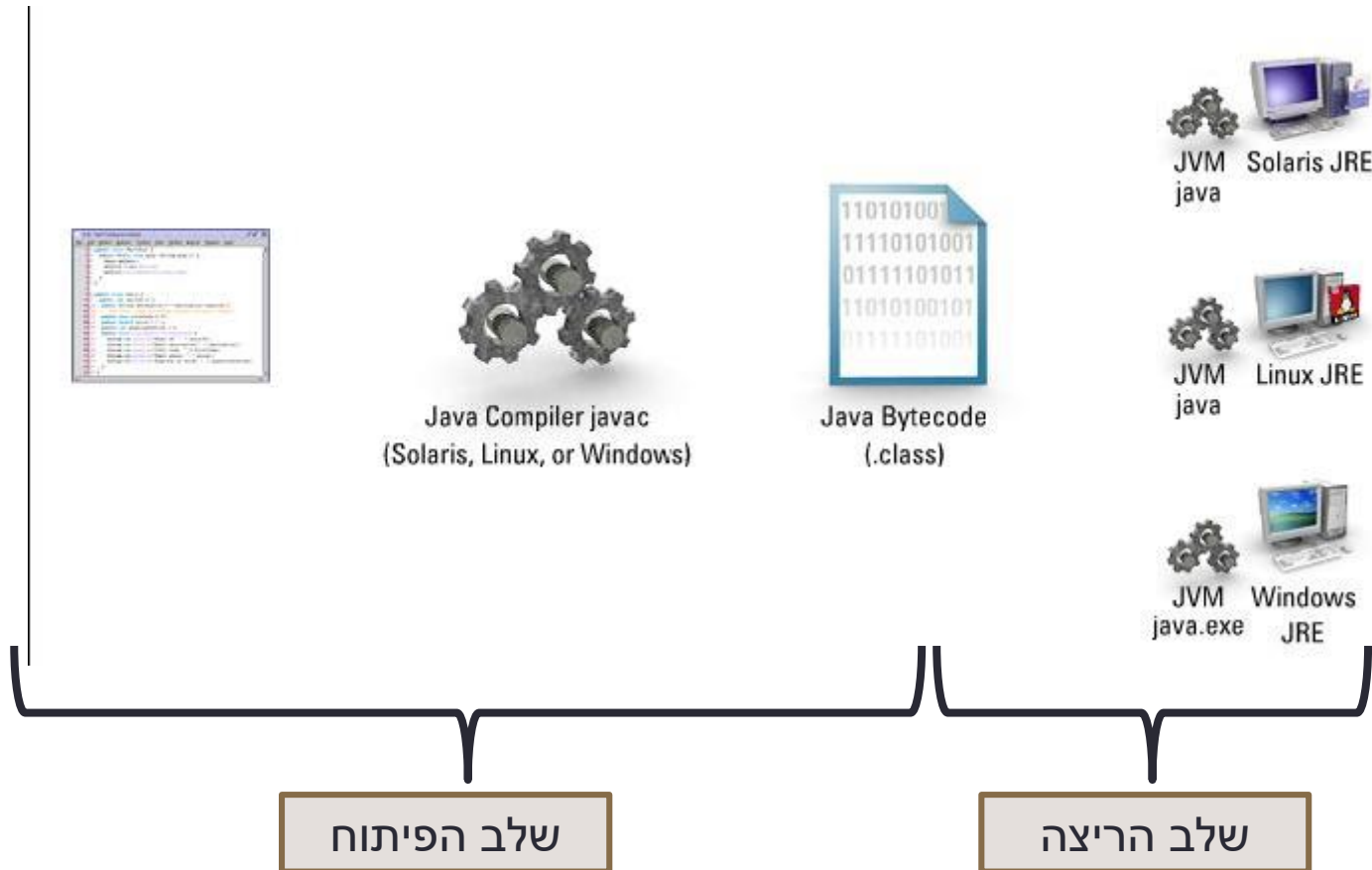
- מדוע אין המהדר יוצר קובץ שתואם בדיוק לחומרת המחשב ?
וכך היה נחסך זמן ריצה וגם הקובץ היה מובן וקריא יותר?!
- זאת מכיוון שאיננו יודעים בדיוק על איזה מחשב תרוץ תוכנית ה java שכתבנו.
- תוכניות java חוצות סביבות
- סביבה = חומרה + מערכת הפעלה

תוכנית שנכתבה והודרה במחשב מסוים, תוכל לרוץ בכל מחשב אשר מותקן בו מפרש לjava .

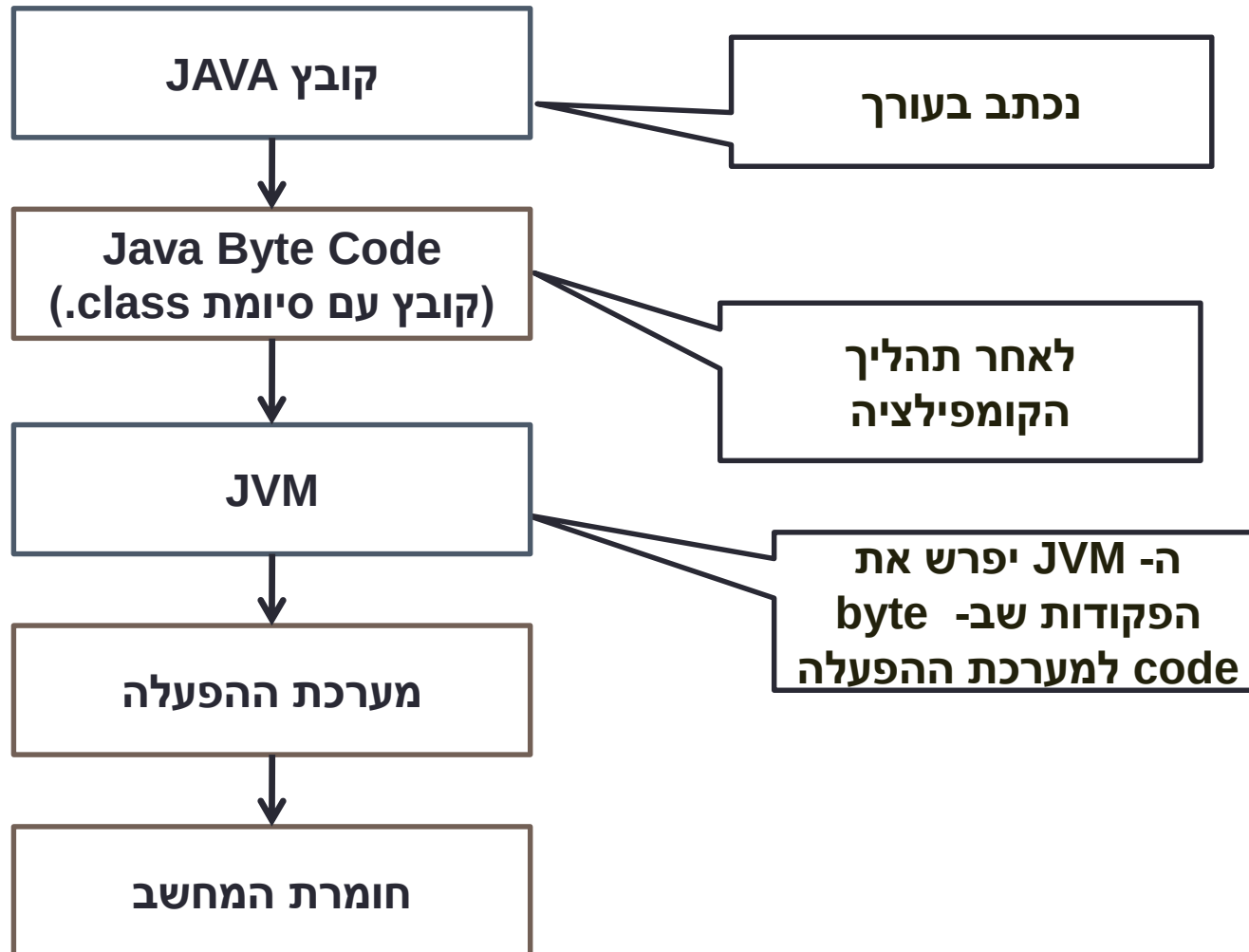
תלות בסביבה



עצמאות סביבתית



תהליך כתיבה והרצה של תוכנית ב-JAVA



סביבת הפיתוח

- סביבת הפיתוח הבסיסית של JAVA נקראת JDK (Java Development Kit) וכוללת את התוכנה והכלים להם זקוק מתכנת JAVA בשביל להדר, לנפות משגיאות ולהריץ תוכניות JAVA. גרסה עדכנית של JDK ניתן להשיג מ-
<http://www.oracle.com/technetwork/java/javase/download>
.S
- סביבת הפיתוח הבסיסית כוללת שימוש בעורך טקסט פשוט עם זאת ישנן סביבות פיתוח ויזואליות כגון: eclipse, JBuilder, JCreator ואחרות.
- בקורס אנו נשתמש בסביבת העבודה הויזואלית eclipse שניתנת להורדה מ- <http://www.eclipse.org/downloads>

תכנות והרצה בסביבת Jdk

- לשם כתיבת קובץ קוד מקור ניתן להשתמש בכל תוכנת עורך תמלילים סטנדרטית.
- הקובץ יישמר בסיומת `java`
- הקובץ יכיל מחלקה אחת בלבד ששמה יהיה כשם הקובץ
- בקובץ המכיל את המחלקה הראשית תהיה פונקציית `main` הבאה:
 - `public static void main (String[ ] args)`
- לאחר סיום כתיבת הקוד יש לקמפל אותו. ביצוע קימפול דרך חלון `dos (cmd)` נעשה באמצעות הפקודה:
`javac FileName.java`

תכנות והרצה בסביבת Jdk

- הערות:
- הקובץ `FileName.java` הוא הקובץ המכיל את המחלקה `FileName` אשר מכילה את פונקציית `-main`.
- יש לגשת ב- `cmd` לתיקייה המכילה את ה- `FileName`
- יש להגדיר ב- `cmd` את ה- `path` של הקומפיילר בצורה הבאה:
- `Set path = C:\Program Files\Java\jdk***\bin`
- כאשר `jdk***` הינה הגרסה של `java` המותקנת במחשב
- במידה וישנן בעיות בקוד, נקבל פלט המכיל אותן. כל פיסקה תכיל את הפרטים הבאים: שם הקובץ ומספר השורה בה נתגלתה השגיאה; תיאור קצר של השגיאה; ציטוט של השורה השגויה עם הדגשת המקום בו ארעה השגיאה.

JDK - JRE

• Java Runtime Environment – JRE :

- מספקת ספריות סטנדרטיות ואת ה-JVM.
- לכל מערכת הפעלה יהיה JRE שמותאם עבורה.
- ללא התקנה של JRE במחשב לא ניתן להריץ אפליקציות JAVA.
- JRE שונים יכולים לספק JVM שיכולים להיבדל למשל בדברים הבאים:
- אלגוריתם שחרורי הזיכרון (ה-Garbage Collector), יוסבר בהמשך.
- אלגוריתם להרצת התהליכים במחשב.

• Java Development Kit – JDK :

- מכילה בתוכה JRE
- כוללת כלי קומפילציה ו-Debugger.

ערכות פיתוח

- JAVA SE (Standard Edition):

- פיתוח אפליקציות שולחניות

- מכילה ספריות סטנדרטיות, ספריות גרפיות, ממשקי משתמש GUI, תקשורת DB,

- JAVA EE (Enterprise Edition):

- מכילה ספריות לעבודה של שרתים, תכנות מבוזר, אבטחה email.

- JAVA ME (Micro Edition):

- לעבודה עם מעבדים קטנים, בדרך כלל בתחום המכשירים החכמים של הסלולרי.

😊 נתחיל להכיר את JAVA

נהוג לקרוא למחלקה ראשית שכוללת את
ה main בשם Program

```
public class Program{
```

לשים לב שה- main נמצא בתוך מחלקה.
העבודה היא עם אובייקטים.

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

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

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

```
}
```

פקודות פלט והדפסה על המסך

```
}
```

פעולות פלט

- פלט (הדפסה) ל- console יבוצע ע"י שימוש בפונקציה print או בפונקציה println הכוללת ירידת שורה
- פונקציות אלו שייכות לצינור out של המחלקה System
- *System.out.println* – היא בעצם קריאה למתודה (זימון מתודה-*method call*). אנו משתמשים כאן בשמה המלא של המתודה. (qualified name) שמכיל את "הנקודה".
- פעולת שרשור מתבצעת באמצעות הסימן +. זה כולל ערכי משתנים. לדוגמא:

```
int x =7;
```

```
System.out.println("x=" + x);
```



יודפס x=7

טיפוסים בסיסיים ב-java

• ב java יש 8 טיפוסים פרימיטיביים

Type	Values	Number Of Bits	Default Value
boolean	true or false	1	false
byte	integers	8	0
char	values Unicode	16	\x0000
short	integers	16	0
int	integers	32	0
long	integers	64	0
float	real numbers	32	0.0
double	real numbers	64	0.0

הגדרת טיפוסים בסיסיים (משתנים)

• בדומה לשפת C, אנחנו מגדירים משתנים באופן הבא:

סוג. טיפוס המשתנה

שם המשתנה

• דוגמא:

- `int a,b;`
- `float x,y,z;`

השמה

- בjava ההשמה מתבצעת מימין לשמאל, כלומר קודם מחושב צד ימין ולאחר נכתב ערך הביטוי לצד שמאל:

`<variable> = <expression>;`

לדוגמא:

```
public class Program {
```

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

```
        int x,y;
```

```
        x=5;
```

```
        y=x+7;
```

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

```
    }
```

```
}
```

סדר

• רמה 1: (), רמה 2: ++, --, רמה 3: %, /, *, רמה 4: -, +, רמה 5: =

```
public class Program {
```

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

```
    int x=10,y=20,c=30,res;
```

```
    res = (x+y) + c/3;
```

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

```
}
```

```
}
```

פלט: 40

המרת טיפוסים

- מה יקרה אם ננסה לכתוב לתוך משתנה מטיפוס מסוים ערך מטיפוס אחר?!

- תלוי במקרה:

- אם ההמרה בטוחה (לא יתכן איבוד מידע), ההמרה בד"כ תצליח. המרה בטוחה נקראת הרחבה (widening).

```
int i = 20;
```

```
long x = i;
```

אנלוגיה!

האם בטוח לשפוך דלי שקיבולתו 8 ליטר לתוך דלי שקיבולתו 4 ליטר?!

לא בטוח! אף על פי שלפעמים זה יצליח, למשל אם יהיו בדלי המקורי רק 2 ליטר.

המרת טיפוסים

• בעזרת פעולת ההמרה (Casting), לשים לב לכמה נקודות:

א. מספרים שלמים הם גם שייכים לקבוצת הממשיים. כלומר אם בחלוקה נוסיף נקודה זה יעשה את ההמרה. דוגמא:

```
public class Program {
```

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

```
int x=5,y=10;
```

```
double z;
```

```
z= (x+y)/2;
```

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

```
z=(x+y)/2.0;
```

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

```
}
```

```
}
```

לא יעבוד עם float

יודפס 7

יודפס 7.5

המרה מפורשת

- אם ההמרה לא בטוחה זו בד"כ שגיאת קומפילציה ונדרשת המרה מפורשת. כלומר ציון הטיפוס החדש בסוגריים לפני הערך.

```
public class Program {  
  
    public static void main(String[] args) {  
  
        int x=5,y=10;  
        float z;  
        z= (x+y)/2;  
        System.out.println(z);  
        z=(float) (x+y)/2;  
        System.out.println(z);  
    }  
}
```

אופרטורים אונריים

$x++$ $x--$	מחזיר את X ומקדם. מוריד אותו ב 1
$++x$ $--x$	מקדם. מוריד ב 1 ואז מחזיר את הערך החדש
-	מספר נגדי, הפיכת סימן
~	הפיכת כל סיביות מספר שלם
!	הפיכה של ערך בוליאני

הערות

התוכנית מיועדת להיקרא ע"י המחשב כלומר ע"י הקומפיילר, אך גם ע"י תוכניתנים.
הערות הן טקסט בתוכנית שמיועד לקוראים אנושיים.

```
/**
```

```
* This is the Program class I've ever written
```

```
* @author Course Lecturer
```

```
*/
```

```
public class HelloWorld {
```

```
/* This is the entry point of my application.
```

```
* as you could see not so interesting...
```

```
*/
```

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

```
        system.out.println("Hello World"); // prints "Hello World "
```

```
    }
```

```
}
```

סוגי הערות בjava

• ב java שלושה סוגי הערות:

- הערה עד סוף השורה //
- הערה רגילה, יכולה להתפרס על מספר שורות */
- הערת תיעוד, יכולה להתפרס על מספר שורות /**

כתבו הערות על מנת לתאר את הקוד:

הערות אודות המובן מאליו רק מכבידות

```
i++; //add one to i
```

אבל הערה טובה יכולה לחסוך הרבה זמן למי שקורא את הקוד

קלט מהמשתמש

- כדי לבצע קליטת נתונים נשתמש במחלקה מיוחדת שנקראת Scanner.
 - מחלקה זו אינה חלק מהשפה אלא מודל חיצוני שנדרש לייבא לתוכנית. את הייבוא אנו מבצעים ע"י הפקודה `import` שנרשמת כשורה ראשונה בתוכנית ואחריה את שם המחלקה ומיקומה.
 - המחלקה Scanner נמצאת בתת-תיקייה `util` של סביבת העבודה `java` מכאן שההוראה לייבוא תיראה בצורה הבאה:
- ```
import java.util.Scanner;
```
- מכיוון ש-Java היא שפה מוכוונת עצמים כדי לקלוט נתונים (שימוש ב-Scanner) עלינו ליצור עצם (אובייקט) מסוג Scanner בצורה הבאה:

```
Scanner input = new Scanner (System.in);
```

# פעולות קלט - Scanner

- ההוראה :

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

- נרשמת בגוף התוכנית לפני הוראת הקלט הראשונה והיא מאפשרת לקלוט נתונים מן המשתמש ולשמור אותם במשתנים שהוגדרו מראש. ההוראה המאפשרת לתוכנית לקרוא מידע מהמסך שנכתב ע"י המקלדת נראית:
- `varName = input.next***();`
- כאשר במקום `***` נרשם טיפוס הנתונים. עבור `int` יירשם `nextInt()` עבור `double` יירשם `nextDouble()` עבור מילה יירשם `next()` עבור מחרוזת יירשם `nextLine()`.
- למחלקה Scanner אין פעולה לקליטת תו בודד ולכן אם נרצה לקלוט תו נשתמש בהוראה:
- `input.next().charAt(0);` שלוקחת את התו הראשון מהמילה שנקלטה.

# דוגמא לקלט

קליטת שני מספרים והדפסת הממוצע

```
import java.util.Scanner;

public class Program {

 public static void main(String[] args) {

 int x,y;
 float res;
 Scanner input = new Scanner (System.in);
 System.out.println("Enter the first number:");
 x = input.nextInt();
 System.out.println("Enter the second number:");
 y=input.nextInt();
 res=(float) (x+y)/2;
 System.out.println("The average is:"+ res);

 }
}
```

דוגמא לפלט:

*Enter the first number:6  
Enter the second number:5  
The average is:5.5*

# פעולות קלט - Scanner

- לכל טיפוס יש פקודת (פונקציית) קלט שונה. להלן חלק מהפונקציות השימושיות:

| פונקציה             | טיפוס                    |
|---------------------|--------------------------|
| nextInt()           | int                      |
| nextLong()          | long                     |
| nextFloat()         | float                    |
| nextDouble()        | double                   |
| next().charAt(0)    | char                     |
| next() \ nextLine() | String (word \ sentence) |