קורס מונחה עצמים בשפת JAVA

מרצה: עסאקלה שאדי

לא לשכוח לפתוח את הטלפונים בסוף השיעור

תקשורת במהלך הקורס

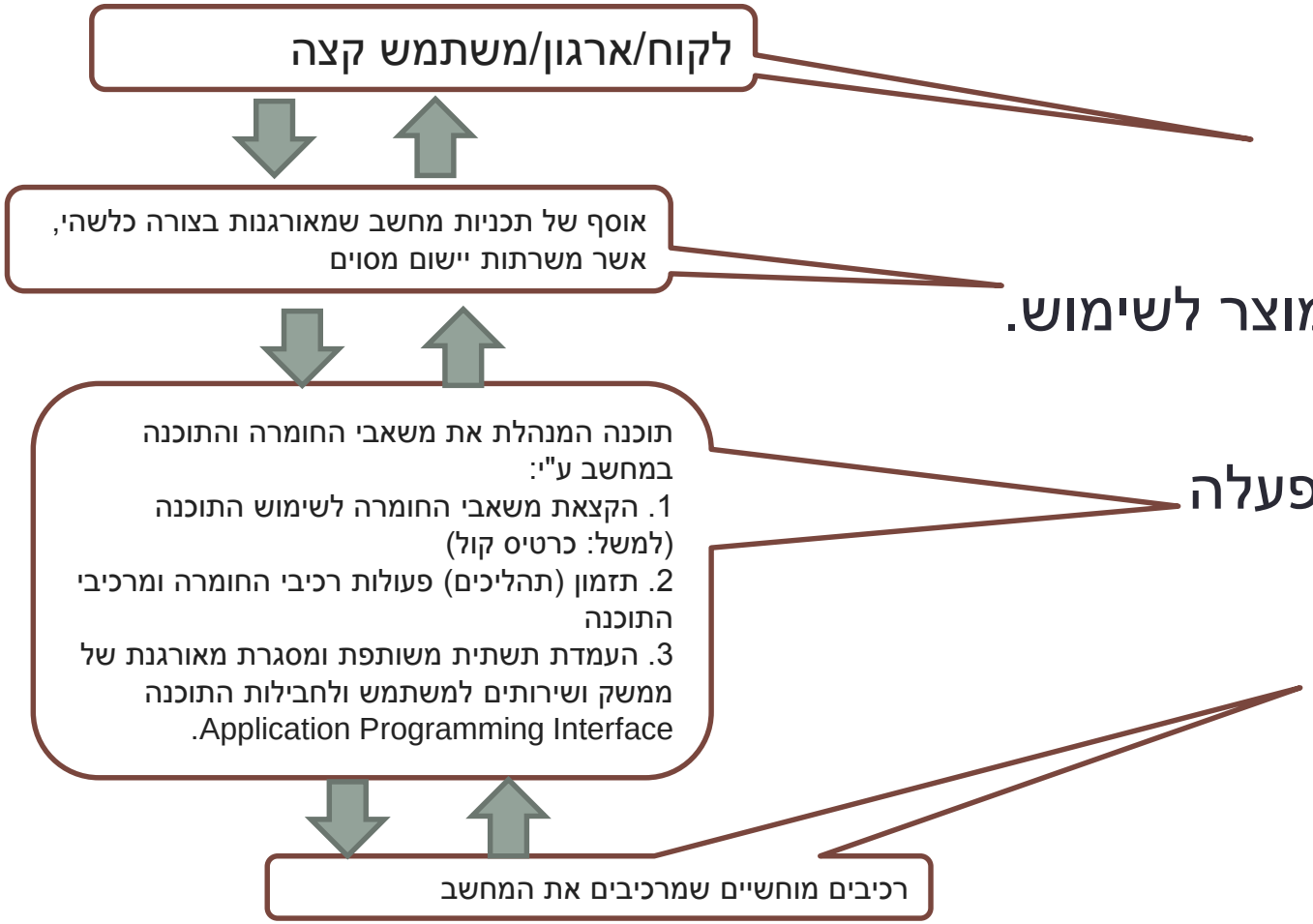
- המייל שלי: as.shadi@campus.technion.ac.il (זמינות כל היום)
- מטלות: הגשת המטלות תהיה במייל. תיבדק אפשרות דרך מערכת ממוחשבת.
- עבודה במעבדה – תרגול.
- העלאת מצגות.

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 המיחשוב.

מבוא – מושגים ששמענו!!!

- בעולם התוכנה יש מושגי יסוד רבים, אבל השחקנים המרכזיים הם:
  - משתמש
  - תוכנה – מוצר לשימוש.
  - מערכת הפעלה
  - חומרה.



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).
- 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 ביחדת 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)
- 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

תכנות מודולרי

- חלוקת מערכות תוכנה גדולות בשיטת "הפרד ומשול"
- חלוקה למשימות קטנות ולתתי משימות אם יש צורך.
- כל תת משימה נקראת מודול – Module
- חלוקה בין צוותי פיתוח
- קלות תחזוקה
- מינימום תקלות
- תיקון שגיאות באזור מסוים.
- שפות למימוש הרעיון – שפות מונחות עצמים.
- הגדרת מבנים כדרך לתכנות – נקראות מחלקות – יורחב בהמשך.
- שפת JAVA כדרך לביצוע המודול.

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.

Comparison with C and C++

- שפת Javaאינה תומכת ב- typedefs, defines או בעיבוד מוקדם - preprocessor
- ב JAVAאין #define macros
- ב- Java, אין פונקציות עצמאיות אלא רק פונקציות שהן איברי מחלקות, ואלה קרויות בדרך כלל מתודות.
- Javaאינה תומכת בפקודת goto
- ב JAVA אין מצביעים – יוסבר בהרחבה בהמשך
- ב- Java יש טיפוס String – יורחב בהמשך
- בניגוד ל-++C שפת JAVA מספקת מערכים אמיתיים כאובייקטים ממחלקה-ראשונה. יורחב בהמשך.
- בדומה ל-++C גם ב- Javaישנם טיפוסים פרימיטיביים כגון, int, float וכו'
- בניגוד ל-++C בשפת JAVA מספקת טיפוס בוליאני אמתי

Comparison with C and C++

- C and C++ Write once, compile anywhere,
  Java Write once, run anywhere/everywhere.
- בכתיבת קוד המחלקה ב ++Cנהוג לפצל כל מחלקה ל- Hול-
  CPP, אבל בשפת JAVAאין הפרדה, כל הקוד נמצא בקובץ אחד
- ב JAVAכל מחלקה בקובץ נפרד.
- הבדלים בבניית מחלקות לרבות בנאי, יורחב בהמשך.
- ב ++C ניתן לרשת מכמה מחלקות, ב JAVAרק ממחלקה אחת.
- ב JAVA אין צורך לשחרר זיכרון.

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.

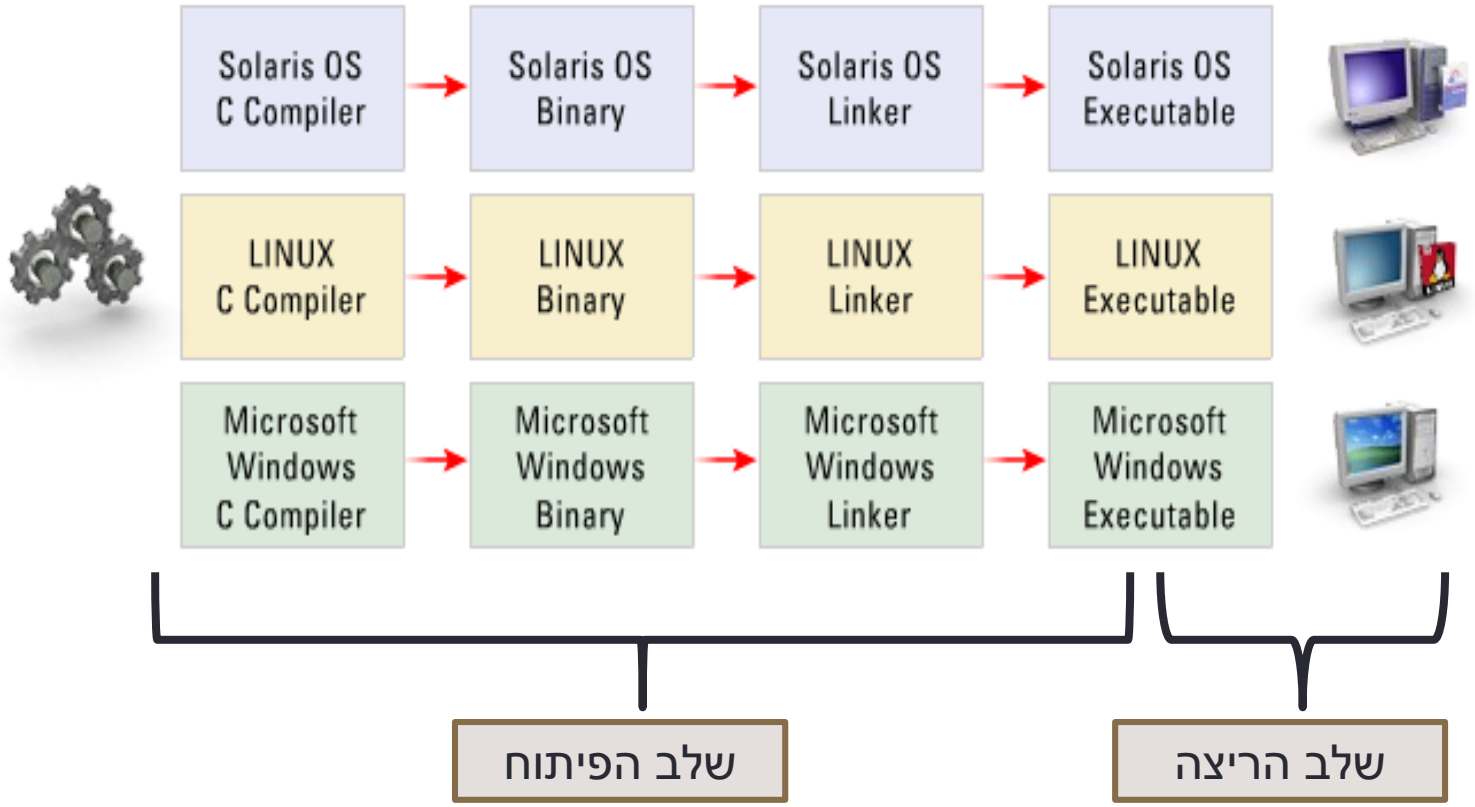
- המפרש נעשה ע"י ספקי תוכנה, שזה תפקידם, עבור רוב סביבות הריצה הנפוצות.

Write Once –Run Anywhere !

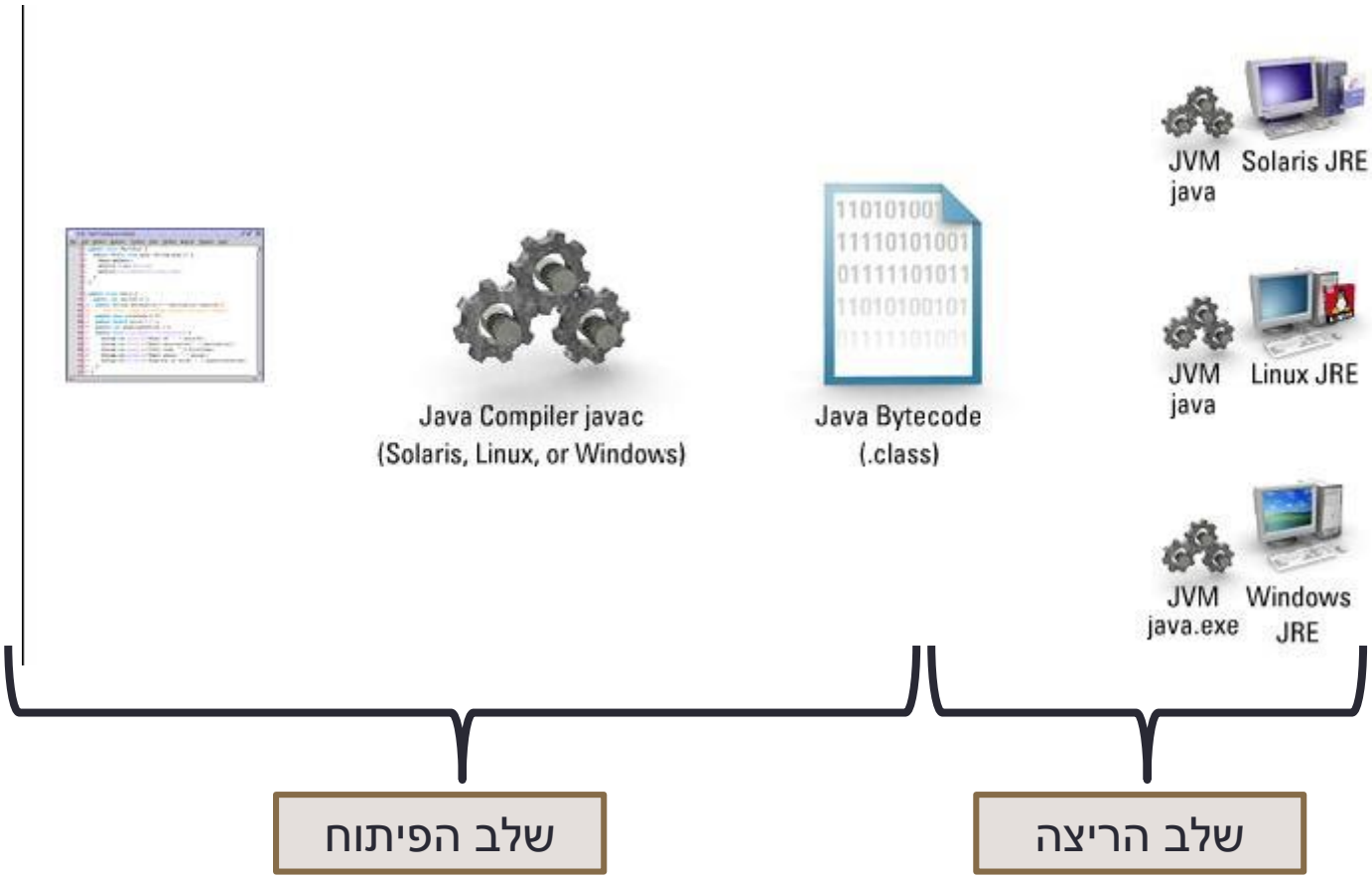
- מדוע אין המהדר יוצר קובץ שתואם בדיוק לחומרת המחשב ? וכך היה נחסך זמן ריצה וגם הקובץ היה מובן וקריא יותר?!
- זאת מכיוון שאיננו יודעים בדיוק על איזה מחשב תרוץ תוכנית ה java שכתבנו.
- תוכניות java חוצות סביבות
  - סביבה= חומרה+ מערכת הפעלה

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

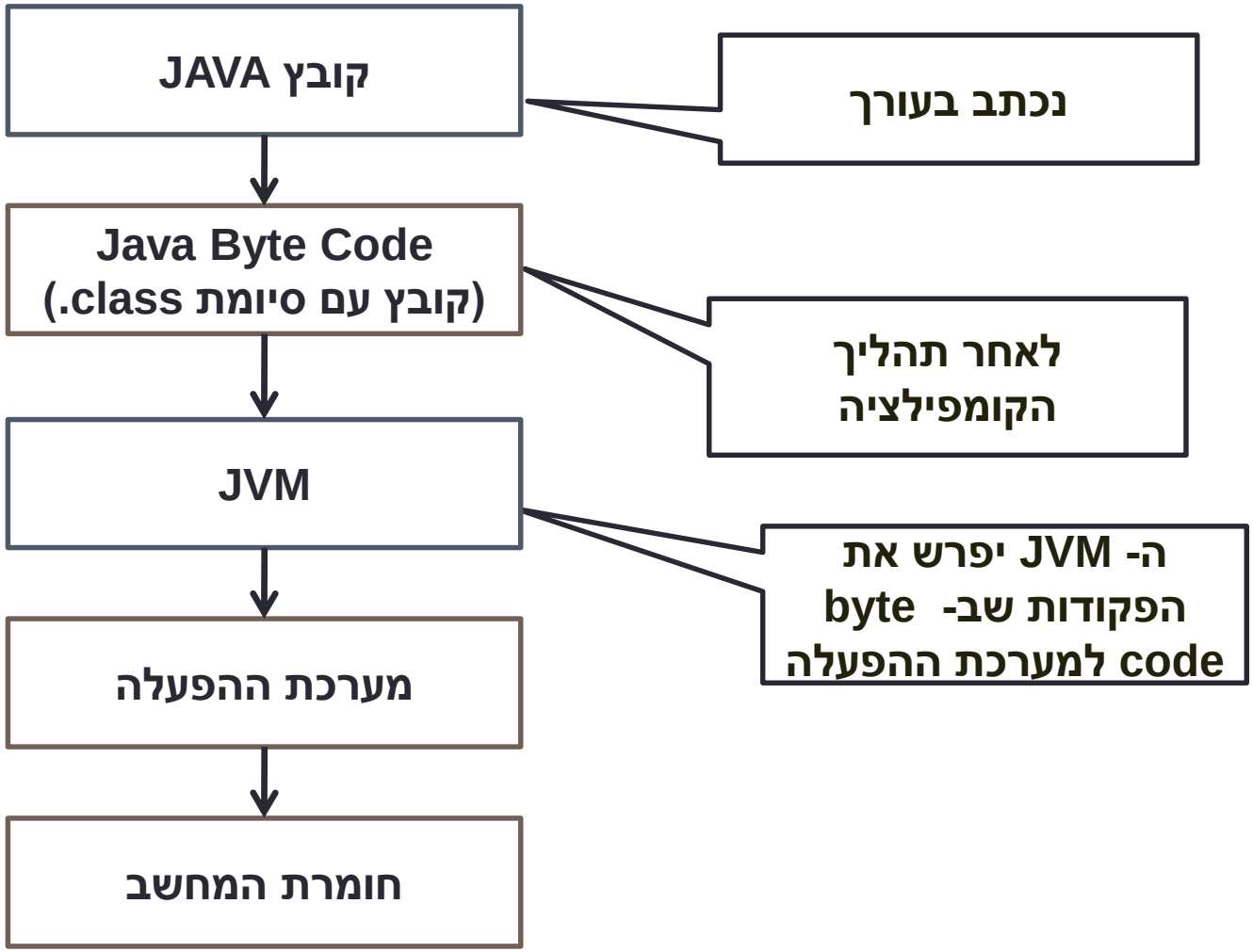
תלות בסביבה


Solaris OS C Compiler
Solaris OS Binary
Solaris OS Linker
Solaris OS Executable
LINUX C Compiler
LINUX Binary
LINUX Linker
LINUX Executable
Microsoft Windows C Compiler
Microsoft Windows Binary
Microsoft Windows Linker
Microsoft Windows Executable
שלב הפיתוח
שלב הריצה

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 JAVAבשביל להדר, לנפות משגיאות ולהריץ תוכניות JAVA. גרסה עדכנית של JDK ניתן להשיג מ- http://www.oracle.com/technetwork/java/javase/downloads.
- 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 ה- 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 ו- 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.

מבוא לתכנות מוכוון-עצמים

- עצם (אובייקט): הוא הבסיס בגישה Object Oriented Programming. בחיי היום יום אנו מוקפים בעצמים ממשיים כגון: מכונית, פלאפון וכו'. עצמים ממשיים אלו חולקים שני מאפיינים: לכולם יש תכונות (attribute) המתארות את מצב העצם ולכולם יש התנהגויות (behavior). לדוגמא: למכונית יש תכונות כמו צבע, דגם, מס' רישוי, נפח מנוע ויש לה התנהגות כמו האצה, האטה, עצירה.
- גם לעצם בתכנות מוכוון עצמים יש תכונות והתנהגויות. תכונות העצם נשמרות במשתנים (variable) והתנהגות מיושמת על ידי פעולות (method).
- <u>לסיכום</u>
- עצם הוא יחידה סגורה ועצמאית המוגדר על ידי תכונות המייצגות את מצבו ופעולות המייצגות את הפעולות שניתן לבקש מהעצם לבצע.

מבוא לתכנות מוכוון-עצמים

- מחלקה (class): מתארת עצם. תבנית שלפיה ניתן ליצור עצמים מסוג מסוים. המחלקה תכיל את תכונות העצם ללא ערכים ואת הפעולות.
- כאשר יוצרים עצם חדש מסוג המוגדר במחלקה יוצרים אובייקט של המחלקה המהווה מופע (instance) של עצם. לדוגמא הנייד של אבי, הנייד של רונן והנייד של נועם הם שלושה מופעים של המחלקה נייד. לכל מופע יש אותם מאפיינים ואותם פעולות אך ערכי המאפיינים לא בהכרח זהים. למשל: עבור המאפיין מס' הנייד לשלושת המופעים יש ערכים שונים אך למאפיין גודל זיכרון למופע של אבי ולמופע של רונן יש אותו ערך.

מבוא לתכנות מוכוון-עצמים

- כימוס: מחלקה הינה "קופסא שחורה" המספקת למשתמשים בה התנהגויות אך מסתירה מהם את המבנה הפנימי שלה. המשמעות של "קופסה שחורה" היא שאין צורך להכיר את המבנה הפנימי של המחלקה על מנת להשתמש בה. לדוגמא בשפת C איננו יודעים כיצד בנויה הספריה stdio.h.
- מאפייני העצם מהווים את המבנה הפנימי של העצם. הפעולות של העצם מקיפות מאפיינים אלו ומסתירות אותם כך שניתן להגיע למאפיינים רק באמצעות פעולות של העצם. אריזת מאפייני העצם נקראת כימוס (encapsulation).

מבוא לתכנות מוכוון-עצמים

- העטיפה של המשתנים בצורה זו מספקת שני יתרונות עיקריים למתכנתים:
- מודולריות
- ניתן לבצע שינויים במבנה הפנימי של העצם מבלי להשפיע על עצמים אחרים ותוכניות המשתמשות בו.
- הסתרת מידע - לאובייקט יש ממשק (interface) שבאמצעותו אובייקטים יכולים לתקשר עמו.

עקרונות הפיתוח בשפת JAVA

- חלוקת הקוד לקבוצות של מחלקות, כל קבוצה נקראת חבילה. המחלקות בחבילה בעלות קשר לוגי (ידוע).
- כל מופע של מחלקה נקרא אובייקט, וכל אובייקט הוא הפנייה (מצביע).
- לזכור: כל קובץ עם סיומת java. מכיל מחלקה אחת בלבד.
- כלל: מחלקה תמיד תתחיל באות גדולה למשל: Book, Student
- כלל: תכונה או פונקציה (מתודה) תתחיל באות קטנה למשל: id, color. שם שמורכב משתי מלים או יותר, מילה תחילת מילה (לא כולל ראשונה) מתחילים עם אות גדולה. למשל:getName()

☺ נתחיל להכיר את JAVA

נהוג לקרוא למחלקה ראשית שכוללת את ה main בשם Program

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

```
public class Program{

    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 בתבצעת באמצעות הסימן +. זה כולל ערכי משתנים.

לדוגמא:

```
int x =7;
System.out.println("x=" + x);
```

יודפס x=7

Packages in java

```
package My_First_Package;
public class Calc {
  public int add(int a, int b){
            return a+b;
  }
  public static void main(String args[]){
            Calc x= new Calc ();
            System.out.println(x.add(10, 20));
  }
}
```

מתודה – יוסבר בהמשך

```
import My_First_Package.Calc;
public class Calc2{
  public static void main(String args[]){
            Calc x= new Calc ();
            System.out.println(x.add(55, 45));
  }
}
```

קוד שנמצא בקובץ java אחר, אבל בתוך Package אחר בפרויקט
שימוש ב import

```
package My_First_Package;
public class Calc3 {
public static void main(String args[]){
Calc x= new Calc ();
System.out.println(x.add(55, 45));
  }
}
```

קוד שנמצא בקובץ java אחר, אבל בתוך Package אותו בפרויקט

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:

שם המשתנה	סוג. טיפוס 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);
        }
}
```

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 4ליטר?!

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 בינאריים לפי סדר הקדימויות שלהם

אופרטורים מתמטיים :

+, -, *, /, %, ++, --, <<, >>, >>>, &, |, ~

אופרטורים לוגיים:

!, &&, ||, ^

אופרטורים להשוואה בין ביטויים:

<, <=, ==, !=, >, >=

אופרטורים אונריים

מחזיר את X ומקדם.מוריד אותו ב 1	x++ x--
מקדם.מוריד ב1 ואז מחזיר את הערך החדש	++x x--
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);
```

Scanner - פעולות קלט

- 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 naxtDouble() 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)

משפטי תנאי- IF()

הסינטקס של משפט התנאי if

```
if(boolean expression)
    statement;
```

דוגמא:

```
import java.util.Scanner;
public class Program {
public static void main(String[] args) {
    int x,y;
    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();
    if(x>y)
            System.out.println("x is the biggest:"+ x);
    }
}
```

משפטי תנאי- IF-ELSE

- הסינטקס של המשפט if- else

```
if(boolean expression)
    statement
else
    statement
```

דוגמא

```
import java.util.Scanner;
public class Program {
public static void main(String[] args) {
    int x,y;
    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();
    if(x>y)
            System.out.println("x is the biggest:"+ x);
    else
            System.out.println("y is the biggest:"+ y);
    }
}
```

משפטי תנאי- Switch

הסינטקס של המשפט switch

```
switch( expression 1 )
{
case constant1:
        statements
        break;
case constant1:
        statements
        break;
default:
        statements
}
```

```
int x= 10;
switch(x)
{
  case 10:
        System.out.println("the number is 10");
  case 20:
        System.out.println("the number is 20");
  default:
        System.out.println("the number is 10
and either not 20");
}
```

לולאות - For

- הסינטקס של לולאה for :

```
for(initial; boolean; alter)
      statements
```

דוגמא 1:

```
int counter = 10, ind;
for(ind=0;ind<counter;ind++)
{
   System.out.println(ind);
}
```

לולאות - While

- הסינטקס של הלולאה while:

```
while(boolean expression)
        statement
```

דוגמא:

```
int counter = 10, ind;
ind=0;
while (ind<counter)
{
   System.out.println(ind);
   ind++;
}
```

לשים לב שמאתחלים את האינדקס

לולאות – Do While

- הסינטקס של הלולאה Do While

```
do
    statement
while(boolean exp)
```

דוגמא:

```
 int ind=0, counter=10;
do
{
   System.out.println(ind);
   ind++;
}
while( ind<=counter);
```

תרגול

- כתוב תכנית אשר קולטת שני מספרים ומדפיסה את כל המספרים הזוגיים שביניהם.
- כתוב תכנית אשר קולטת מספרים עד שייקלט הערך 1-. התוכנית תדפיס את ממוצע המספרים שנקלטו.

תרגול

- כתוב תכנית אשר מכריחה את המשתמש לקלוט מספר שלם n בין 7-10 כולל.
- התוכנית תדפיס על המסך את הצורה הבאה:

```
*
*
* * *
* * * *
* * . n כוכביות. * *
* * * *
* * *
* *
*
```