

תוכנה 1

תרגול 1: סביבת העבודה ומבוא ל-Java
אלכסיי זגלסקי וניר אטיאס

פירוק רטיה

אלכסיי זגלסקי

שעת קבלה: שני 13:00-14:00, בתיאום מראש
משרד: בניין הנדסת תוכנה, חדר 209

ניר אטיאס

שעת קבלה: בתיאום מראש
משרד: שנקר פיזיקה 405

אתר הקורס: <http://courses.cs.tau.ac.il/software1/1112a>

סביבת המחשוב באוניברסיטה היא Linux

■ תנאי קדם: פתיחת חשבון אישי במחשבי האוניברסיטה

■ הנחיות לפתיחת חשבון והכרת סביבת העבודה באתר הקורס.

עוד? בירוקרטיה

■ נוהל הגשת תרגילים (פרטים מלאים ב[אתר](#))

■ מועד ההגשה

■ שיטת חישוב הציון

■ הגשה באיחור

■ הגשה דרך ה- VirtaulTAU

■ הגשת תרגיל מספר 1

■ בעקבות השביתה שונה תרגיל מספר 1

■ מועד ההגשה שלו נדחה ליום רביעי 9.11.2011

■ פרטים באתר

write once

```
C:\WINDOWS\system32\cmd.exe  
C:\>  
C:\>javac HelloWorld.java
```

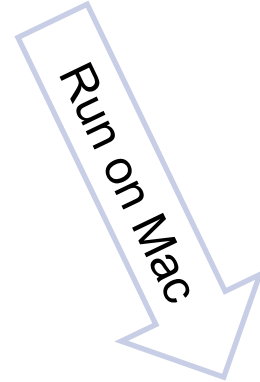
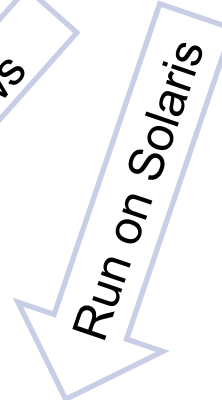
```
HelloWorld.java - Notepad  
File Edit Format View Help  
public class HelloWorld {  
    public static void main(String[] arguments) {  
        System.out.println("Hello world");  
    }  
}
```



HelloWorld.java



HelloWorld.class



```
C:\WINDOWS\system32\cmd.exe  
C:\>  
C:\>java HelloWorld
```



SOLARIS



Write Once – Run Anywhere !

המפרש (interpreter)

- את הקוד שנכתב בשפת Java מריץ מפרש
- בדומה לשפת Scheme
- לריצה בעזרת מפרש יש כמה חסרונות:
 - מאט את מהירות הריצה
 - טעויות מתגלות רק בזמן הריצה
- לצורך כך הוסיפו ב Java שלב נוסף – הידור (compilation)

המהדק (compiler)

- מבצע עיבוד מקדים של קוד התוכנית (שכתובה בקובץ טקסט רגיל) ויוצר קובץ חדש בפורמט נוח יותר
- קובץ זה אינו קריא למתכנת אנושי (אף שניתן לפתוח אותו בעורך טקסט כגון Notepad), אולם המבנה שלו מותאם לקריאה ע"י המפרש של Java
- פורמט זה נקרא byte code והוא נשמר בקובץ עם סיומת .class
- בתהליך העיבוד ("קומפילציה") נבדק התחביר של הקוד – והשגיאות המתגלות מדווחות למתכנת

יציאות (portability)

- מדוע אנו מסתפקים בפורמט "נוח יותר"?
- מדוע אין המהדר יוצר קובץ בפורמט התואם בדיוק לחומרת המחשב, וכך היה נחסך בזמן ריצה גם שלב ה"הבנה" של הקוד?
- זאת מכיוון שאיננו יודעים מראש על איזה מחשב בדיוק תרוץ תוכנית ה-Java שכתבנו
- תוכניות Java חוצות סביבות (cross platform)
- סביבה = חומרה + מערכת הפעלה
- תוכנית שנכתבה והודרה במחשב מסוים, תוכל לרוץ בכל מחשב אשר מותקן בו מפרש ל-Java

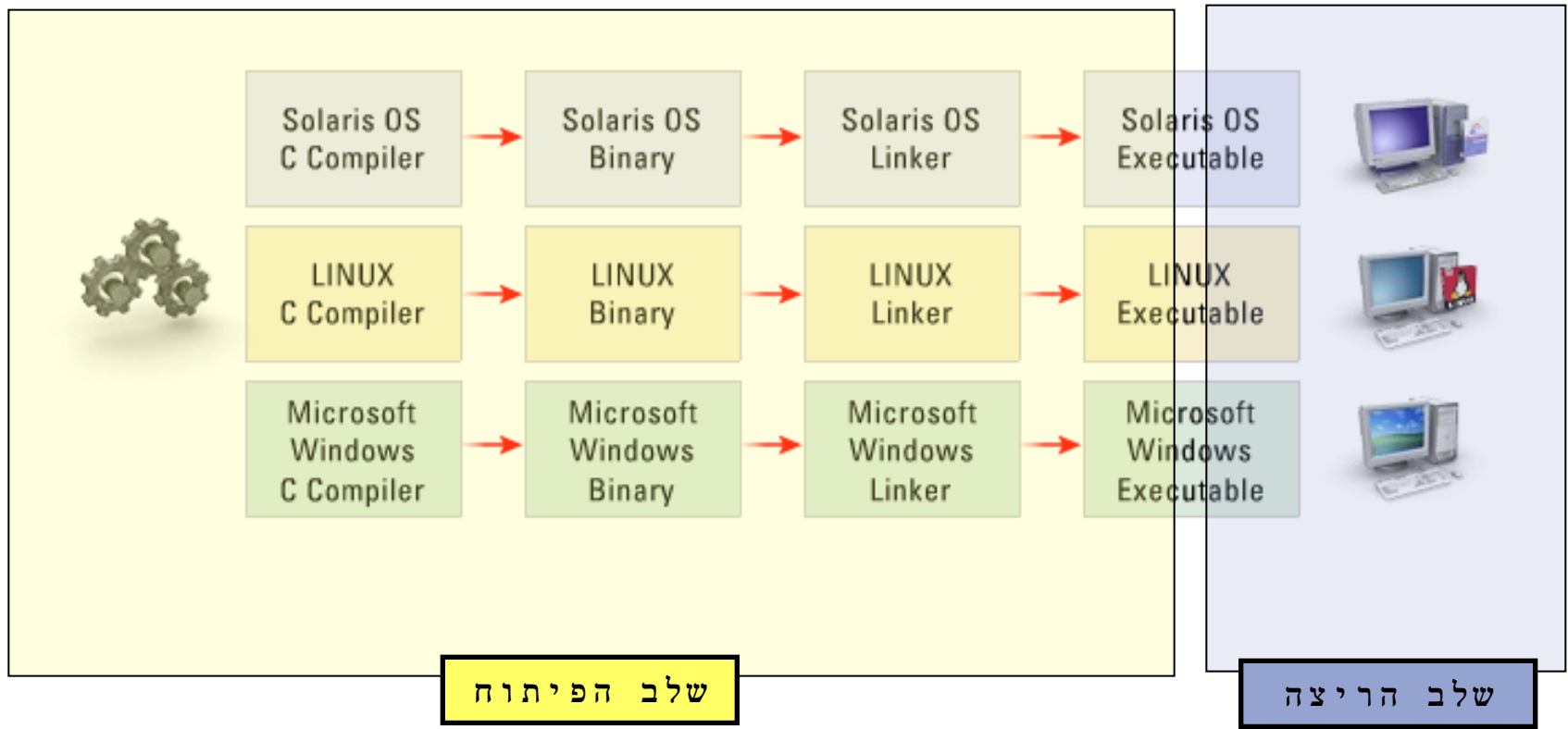
המכונה הוירטואלית

(Java Virtual Machine)

- הקובץ המהודר מכיל הוראות ריצה ב"מחשב כללי" – הוא אינו עושה הנחות על ארכיטקטורת המעבד, מערכת ההפעלה, הזיכרון וכו'...
- עבור כל סביבה (פלטפורמה) נכתב מפרש מיוחד שיודע לבצע את התרגום מהמחשב הכללי, המדומה, למחשב המסוים שעליו מתבצעת הריצה
- את המפרש לא כותב המתכנת!
- דבר זה כבר נעשה ע"י ספקי תוכנה שזה תפקידם, עבור רוב סביבות הריצה הנפוצות

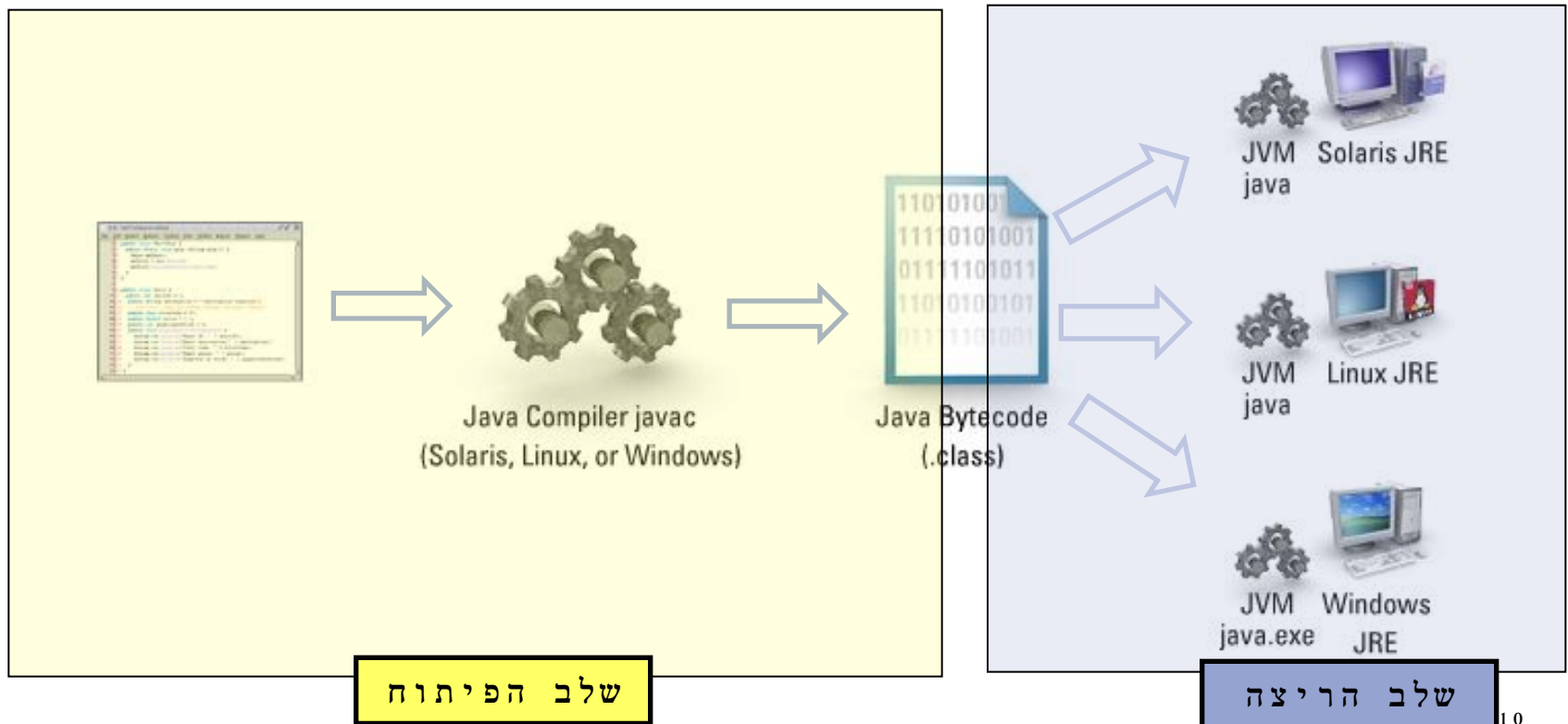
תלות בסביבה (platform specific)

■ בשפות אחרות (C/C++) אין הדבר כך:



עצמאות סביבתית (platform independence)

■ ב Java תכונה זו אפשרית הודות לרעיון "שפת הביניים" וה JVM הנפרד לכל סביבה



life of a



המחלקה ציבורית –
ניתן להשתמש בה ללא
הרשאות מיוחדות

הגדרת מחלקה בשם HelloWorld.
בשלב זה, נזוהה מחלקה עם קובץ
באותו שם

```
public class HelloWorld {
```

יצירת תחום (scope)

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

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

```
}
```

```
}
```

חתימת המתודה

גוף המתודה

הגדרת מתודה
(פונקציה)

המתודה main

```
public static void main(String[] arguments) {  
    System.out.println("Hello World");  
}
```

כאשר אנו מריצים מחלקה ה JVM מחפש מתודה עם

חתימה זו, ומריץ אותה

main – שם המתודה

public - המתודה ציבורית – ניתן להשתמש בה

ללא הרשאות מיוחדות

static – מתודה של המחלקה (יוסבר בהמשך)

void – טיפוס הערך המוחזר. למתודה זו אין ערך

מוחזר (ריק = void)

המתודה main

```
public static void main(String[] arguments) {  
    System.out.println("Hello World");  
}
```

String[] arguments הגדרת פרמטר
למתודה בשם arguments ומטיפוס מערך של
מחרוזות

לכל המשתנים ב Java יש טיפוס המעיד על סוג וטווח
הערכים שיכולים להיות מאוחסנים במשתנה (למשל:
מספר שלם, תו, משפט, מערך ואחרים)
שם המשתנה אינו חלק מהחתימה של המתודה

המתודה main

```
public static void main(String[] arguments) {  
    System.out.println("Hello World");  
}
```

- `System.out.println` קריאה למתודה (method call, זימון מתודה) – אנו משתמשים כאן בשמה המלא של המתודה (qualified name) המכיל את התו '.' (נקודה)
- העברת ארגומנט מטיפוס מחרוזת (String) – משפט עטוף במרכאות הוא מטיפוס מחרוזת
- משפטים ב Java מסתיימים בתו ';' (נקודה-פסיק)

סביבת פיתוח והרצה Java-f

■ גרסת ה-Java שעמה נעבוד:

Java SE (Standard Edition) 6.0

■ חבילת סביבת ההרצה:

JRE (Java Runtime Environment) that includes:

- JVM (Java Virtual Machine)
- Standard Class Library

■ חבילת ערכת הפיתוח:

JDK (Java Development Kit) that includes:

- JRE
- Command line tools: compiler, debugger etc.

■ הורדה ותיעוד ב-

<http://www.oracle.com/technetwork/java/javase/downloads/index.html>

סביבת פיתוח שלוקה

IDE = Integrated Development Environment ■

סביבה המשלבת רכיבי/כלי פיתוח עצמאיים: ■

עורך טקסט (editor) ■

סייר הקבצים (browser) ■

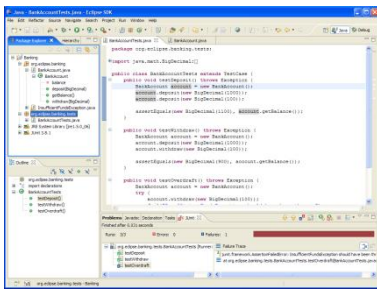
מהדר (compiler) ■

סביבת זמן ריצה (JRE) ■

מנפה השגיאות (debugger) ■

ועוד... ■

Eclipse – ה- IDE בו נשתמש בקורס. ■



Eclipse

- IDE המתאים גם לפיתוח תוכנה ב Java
- ניתן להתקנה ב- Linux, Windows ועוד
- דורש התקנה בנפרד של JRE (או JDK)
- אתר הבית: www.eclipse.org

- הורדת התוכנה כקובץ zip (הוראות התקנה ב-[הכרת סביבת המחשוב](#) באתר הקורס)
- אוסף גדול של מאמרים

■ הכרות: [דפי עבודה ללימוד Eclipse](#) באתר הקורס

■ דוגמא: פיתוח והרצת תכנית "Hello World" ב Eclipse

■ הסבר מפורט לגבי השמשת סביבת העבודה מהבית:

<http://courses.cs.tau.ac.il/software1/1112a/misc/workenv.pdf>

הערות תיפוס

- התוכנית מיועדת להיקרא על ידי המחשב (למעשה על ידי הקומפילר), אבל גם על ידי תוכניתנים
- הערות הן טקסט בתוכנית שמיועד לקוראים אנושיים

```
/**  
 * This is the first 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

- בג'אווה שלושה סוגי הערות:
 - הערה עד סוף השורה //
 - הערה רגילה, יכולה להתפרס על מספר שורות /*
 - הערת תיעוד (יכולה להתפרס על מספר שורות) /**
- הערות לתיעוד שמופיעות לפני הגדרת מחלקה, שדה, או שירות עוברות, בעזרת כלי שנקרא javadoc לתיעוד המקוון של המחלקה
- הערות לתיעוד הן מובנות, ויש להן פורמט מיוחד שמיועד לאפשר לתוכניתן לתעד את הארגומנטים של שירות, את משמעות ערך החזרה, וכדומה
- כתבו הערות על מנת לבאר את הקוד:
 - הערות אודות המובן מאליו רק מכבידות: `i++ // add one to i`
 - אבל הערה טובה יכולה לחסוך הרבה זמן למי שקורא את הקוד

...פיו