

# תוכנה 1 בשפת Java

## שיעור מספר 1: "שלום עולם"

### פרופ' ליאור וולף

בית הספר למדעי המחשב  
אוניברסיטת תל אביב



presented by kenichi-naitou.

# מה בתכנית?

- טעימה משפת Java
- פונקציית `main`
- 8 הטיפוסים היסודיים
- ביטויים ואופרטורים
- טיפוס המחרוזת וטיפוס המערך



# שלום עולם

```
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

compile



HelloWorld.class

Run on Windows

Run on Solaris

Run on Mac

```
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 Python

- לריצה בעזרת מפרש יש כמה חסרונות:

- מאט את מהירות הריצה

- טעויות מתגלות רק בזמן הריצה

- לצורך כך הוסיפו ב Java שלב נוסף – *הידור* (compilation)

# המהדר (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

# יבילות (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

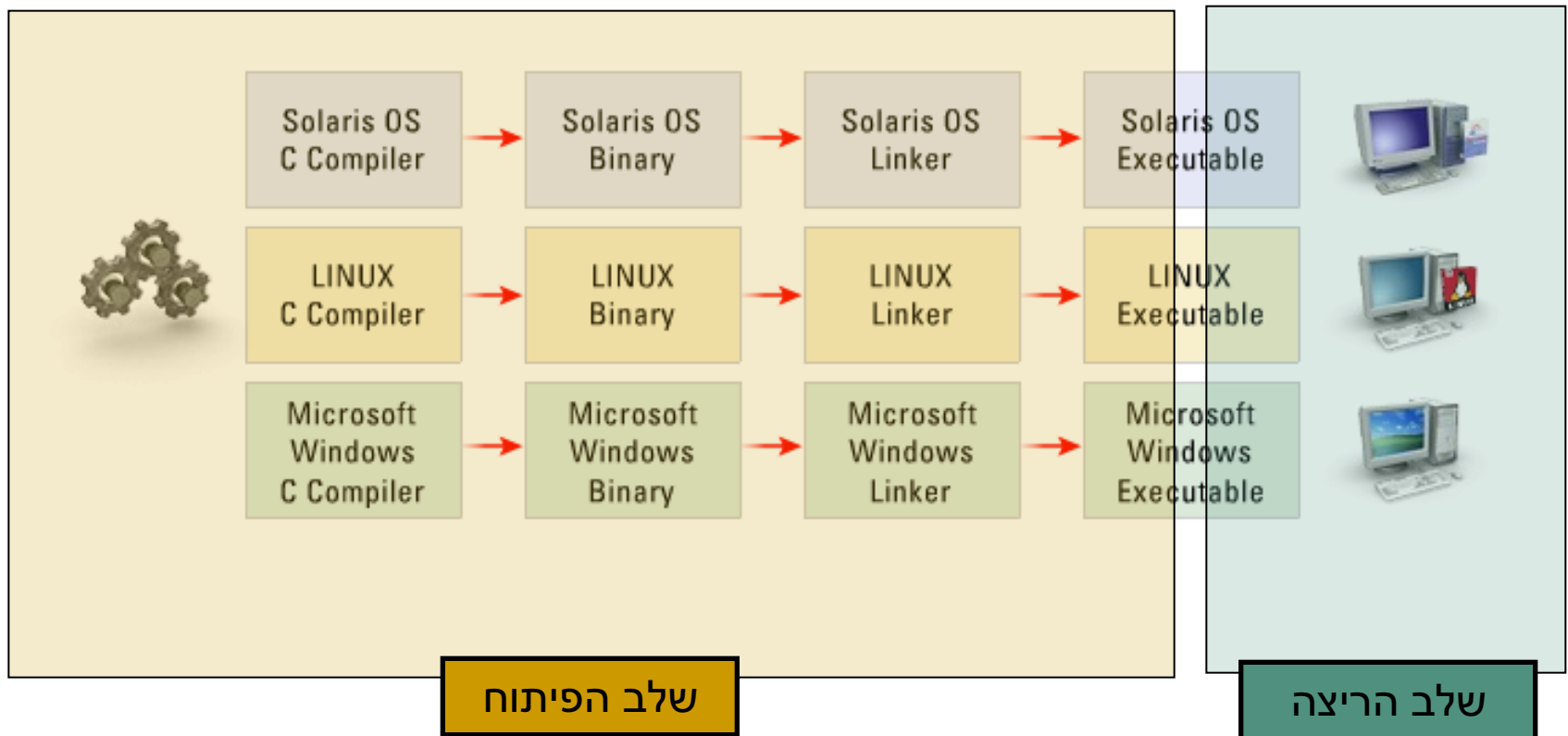
# המכונה המדומה

## (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)

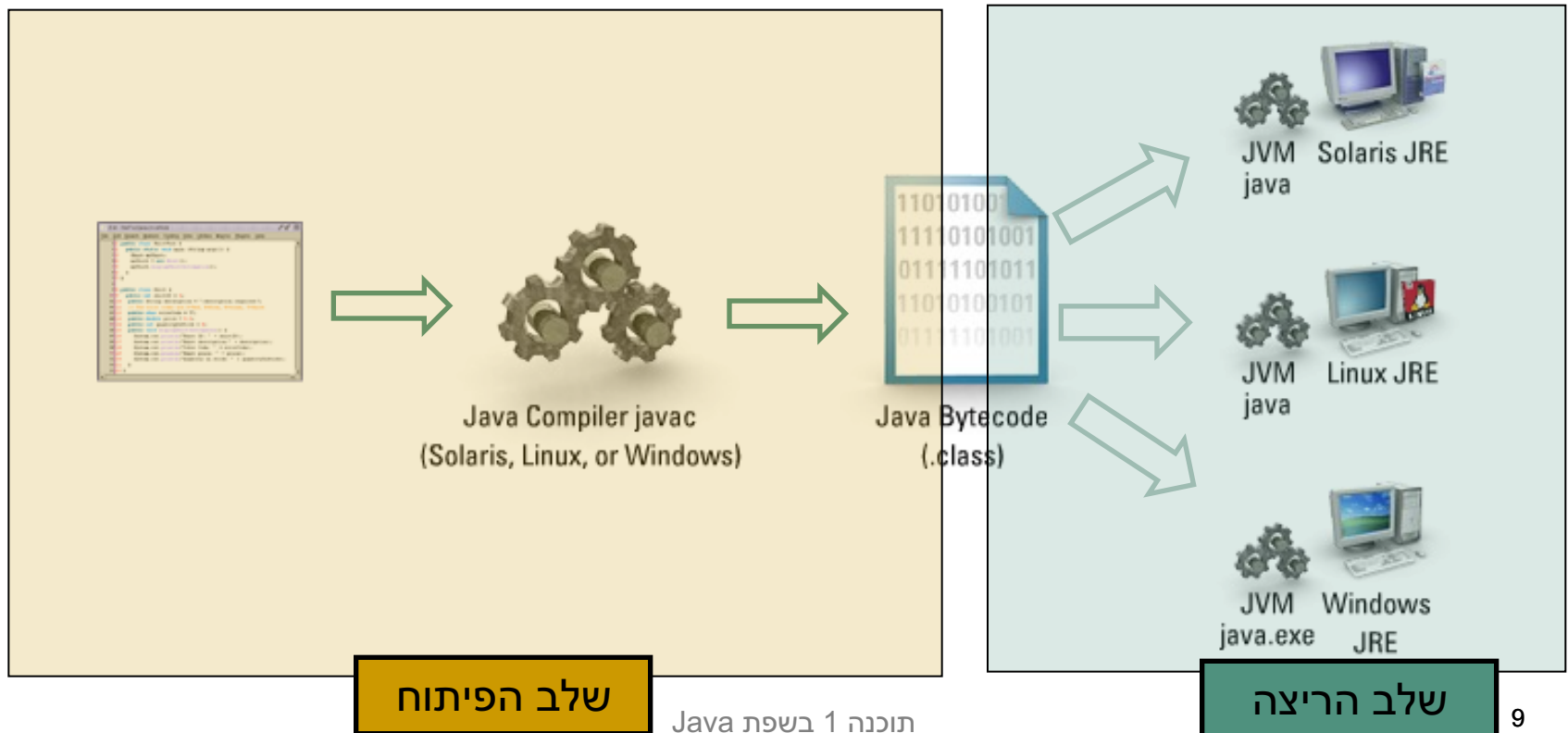
■ בשפות אחרות (C/C++) אין הדבר כך:





# עצמאות סביבתית (platform independence)

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



תוכנה 1 בשפת Java  
אוניברסיטת תל אביב



# שלום עולם

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

המחלקה ציבורית –  
ניתן להשתמש בה ללא  
הרשאות מיוחדות

חתימת המתודה

```
public class HelloWorld {
```

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

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

```
}
```

גוף המתודה

הגדרת מתודה (פונקציה)

יצירת תחום (scope)

# 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)