

מעבדה מספר 1

Java (+JDK), IntelliJ, Maven

המעבדה מיועדת להכרת Java וכלי הפיתוח המשמשים לפיתוח אפליקציות Java, והיא כוללת שלושה חלקים: חלק א' – IntelliJ, חלק ב' – Maven, חלק ג' – JDK.

חלק א' – IntelliJ IDEA

מטרת חלק זה היא היכרות עם סביבת הפיתוח IntelliJ IDEA. IntelliJ IDEA היא סביבת פיתוח משולבת לפיתוח תוכנה (IDE – Integrated Development Environment), המתאימה במיוחד ל-Java. היא חופשית (בגרסת Community Edition), פתוחה ופופולרית. קיימת לה גם גרסה בתשלום, המכילה תכונות רבות נוספות (אין צורך בהן לקורס, אך הן עשויות לעזור במקרים מסוימים). אתם, כסטודנטים, יכולים לקבל אותה בחינם [כאן](#).

סביבות פיתוח הן כלי חשוב למתכנת:

- עוזרות בניהול מידע בפרויקטים גדולים.
- מספקות ממשק גרפי אחיד ומקלות על העבודה.
- מבצעות תהליכים קבועים בצורה אוטומטית.
- מכילות קיצורים.
- מסובכות יחסית ולכן דורשות זמן ללימוד.

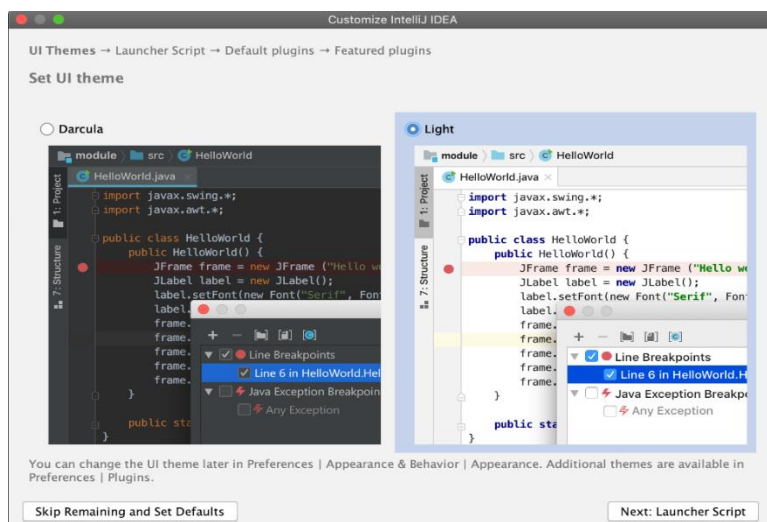
באתר הקורס, במדור Java, תמצאו מדריך להתקנת סביבת העבודה (בו תיעזרו להתקנת סביבת העבודה על מחשבכם האישי).



א. הפעלת IntelliJ

זהו הלוגו של IntelliJ. כאשר מפעילים את סביבת העבודה בפעם הראשונה, ייפתח אשף הגדרות שבו יש לבחור את ההעדפות הרצויות לכם.

בשלב הראשון, עליכם לבחור את ערכת הנושא המועדפת עליכם: כהה (Drakula) או בהירה:

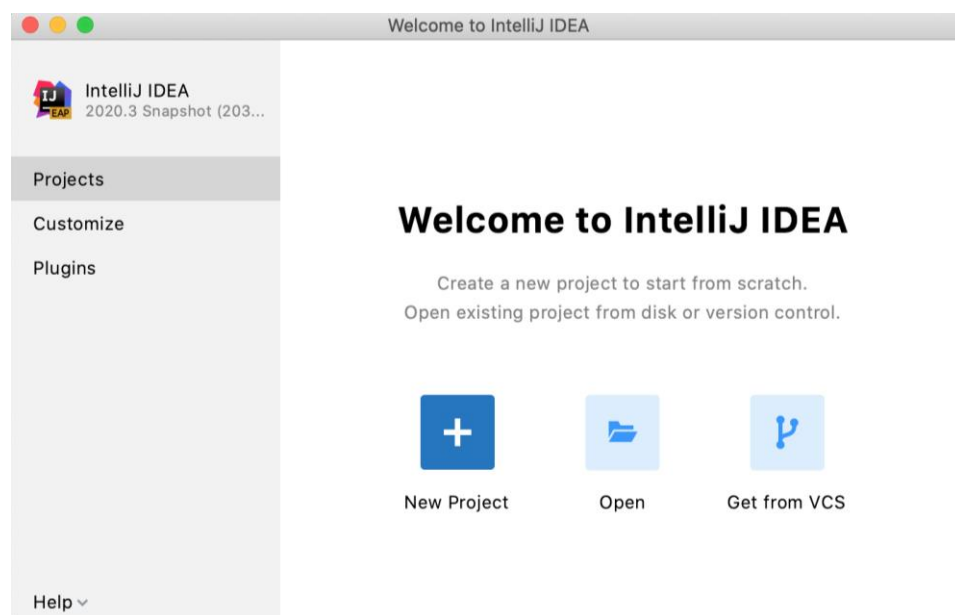


המעבדה נכתבה ע"י ד"ר מלכי גרוסמן, ליאל פרידמן ושיר סנה

לאחר מכן, עבור מערכות macOS ו-Linux, האשף יציע לכם ליצור סקריפט הפעלה. תוכלו לבחור לעשות את זה (מומלץ אם אתם מתכננים להריץ את הסביבה משורת הפקודה), או לדלג על כך.

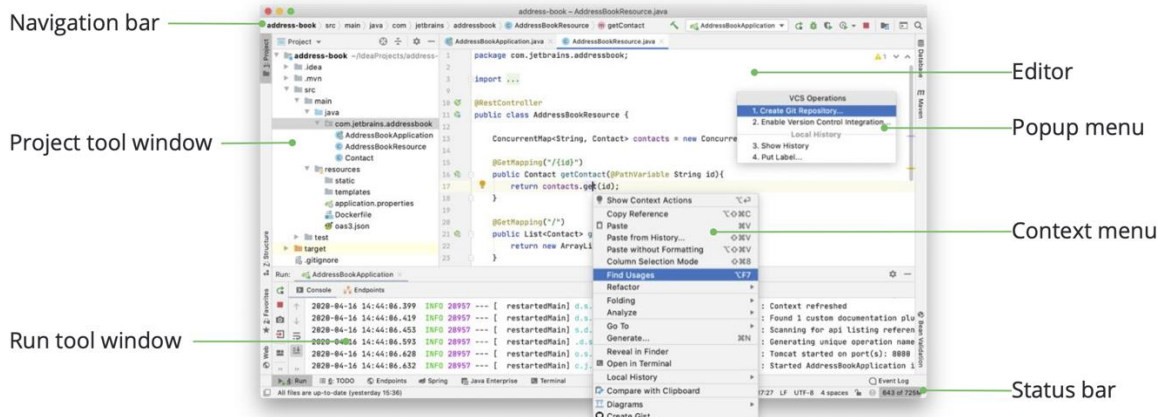
בשלב השלישי והרביעי, תוכלו לבחור תוספים (plugins) שונים לסביבת העבודה. מומלץ להתקין (אם מופיע לכם) את ההרחבות הבאות: Java Frameworks, Build Tools, Version Controls, Test Tools.

לאחר מכן, כאשר תלחצו על Start using IntelliJ IDEA, יופיע מסך הפתיחה של סביבת העבודה:



בחלון זה תוכלו לבחור האם ליצור פרויקט חדש, לפתוח פרויקט קיים או להוריד פרויקט ממערכת ניהול גרסאות (על כך במעבדה 5 – Git). תפגשו אותו בכל פעם שתצאו להחליף בין פרויקטים (בעת הפעלת הסביבה, אם אין פרויקט פתוח, וכן בעת סגירת פרויקט פתוח בסביבה). נרחיב על יצירת פרויקט חדש בהמשך.

ב. חלון סביבת העבודה של IntelliJ



חלון Project Tool: בו מופיעים כל הקבצים (בין היתר מחלקות בג'אווה, אבל לא רק) המקושרים לפרויקט הנוכחי.

חלון העורך: בו מוצג הקובץ הנוכחי (בדרך-כלל מחלקה) עליו עובדים – מעין "מעבד תמלילים". חלונות הכלים (כאן מוצג חלון כלי הריצה – Run Window): חלונות אלו מספקים פונקציונליות נלווית לעריכת קוד. בין היתר, הרצה של תוכניות תופיע כאן. בנוסף, בלשונית Problems יופיעו בעיות שונות בקוד – שגיאות (בדרך כלל יהיו שגיאות קומפילציה) ואזהרות (שגם אליהן חשוב לשים לב ולהימנע ככל הניתן).

סרגל הניווט: בו מופיע מיקום הקובץ הנוכחי, וכן המחלקה והמתודה שבה אנחנו נמצאים כרגע בעורך.

סרגל הסטטוס (Status Bar): בצד שמאל, מופיעות ההתראות האחרונות שהתקבלו ותיאור של פעולות (כאשר נמצאים עם העכבר על כפתור מסוים). ניתן ללחוץ על התראה כדי לפתוח אותה ביומן האירועים וניתן להעתיקה בלחיצה ימנית על העכבר. בצד ימין, ישנם וידג'טים (widgets) שונים המתארים את המצב הנוכחי של סביבת העבודה, בהם: מספר השורה והעמודה, אופן המעבר בין שורות בקובץ, קידוד הקובץ, הזחה (indentation), האם הקובץ לקריאה בלבד והשימוש הנוכחי בזיכרון המחשב.

תפריט הקשר (Context Menu): מופיע בלחיצה ימנית על העכבר ומכיל, בנוסף לפעולות הסטנדרטיות של העתקה והדבקה, גם פעולות נוספות שיוכלו לעזור לכם. מומלץ להשקיע דקה ולחקור את האופציות השונות, במיוחד בתפריט Refactor – הן יכולות לחסוך לכם זמן רב במהלך העבודה.

תפריט פופאפ (Popup Menus): מופיעים בעת לחיצה על קיצורי מקשים מסוימים ומציעים פעולות מקבוצה מסוימת, למשל: כאלה שמתאימות לניהול גרסאות (Alt+` בווינדוס ובלינוקס, Ctrl+V ב-macOS).

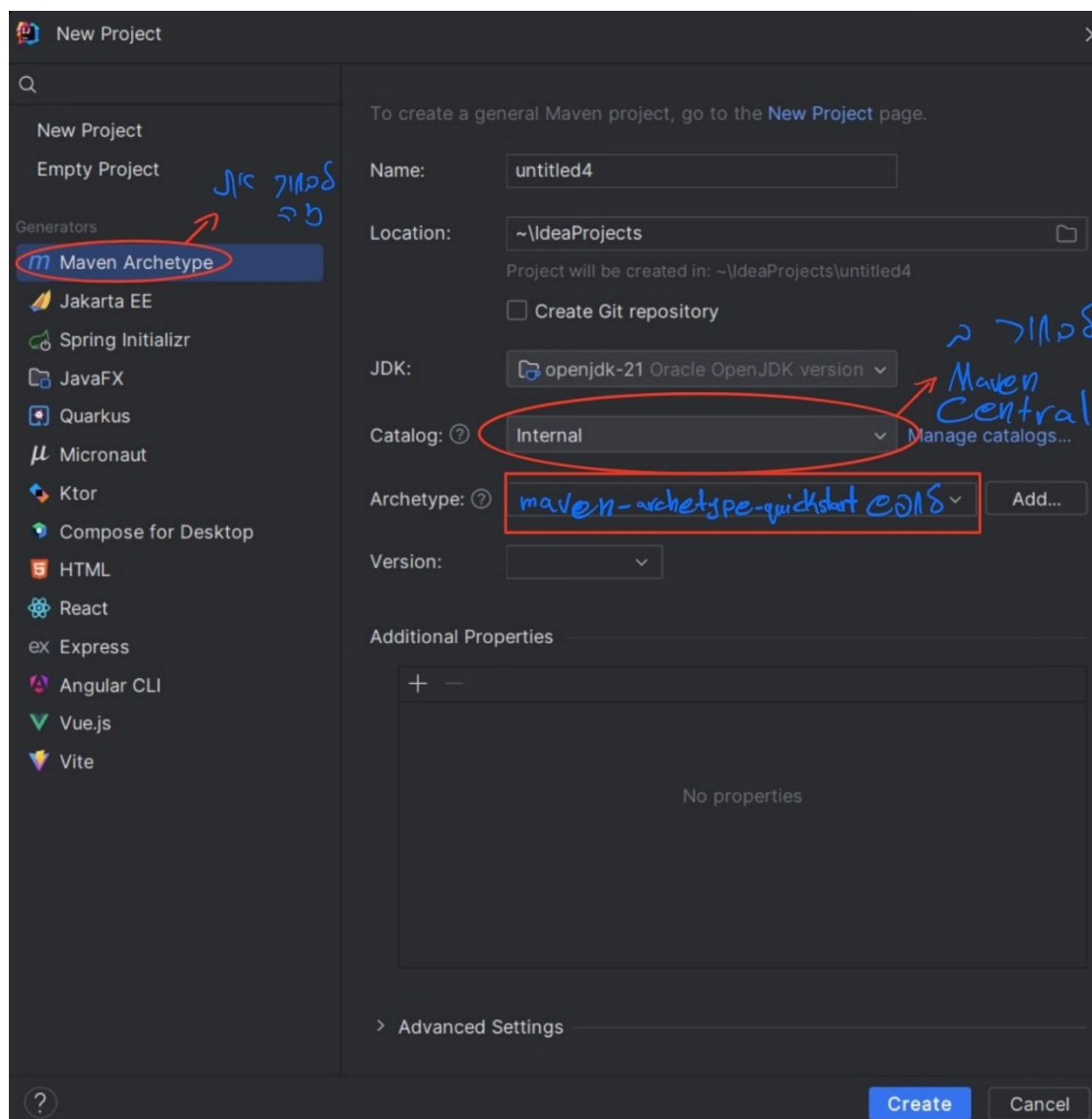
ג. **Maven** – Maven היא כלי Build Dependency Management לסביבת ה-JVM (Java Virtual Machine) ובפרט ל-Java. זהו כלי נוח לניהול של פרויקטים, שנכתב ומתוחזק ע"י קרן Apache. הסבר נרחב יותר על Maven נראה בהמשך המעבדה הנוכחית.

ד. יצירת פרויקט

על מנת לכתוב קוד ולהריץ אותו, עלינו ליצור תחילה פרויקט. פרויקט מכיל את הקבצים הבאים:

- קובצי המקור של הקוד.
 - הגדרות עבור הפרויקט (למשל דגלי הקומפילציה הדרושים).
 - קבצים נוספים נדרשים (כמו קובצי JAR או קבצים גרפיים נדרשים).
- יצירת פרויקט נעשית ממסך הפתיחה, או בתפריט File->New->Project.

בחלון שנפתח נבחר בצד שמאל ב-Maven. Project SDK נבחר את גרסה 17 או 21 (אם לא נבחרה עבורנו באופן אוטומטי). נסמן את התיבה של Create from archetype, על מנת שנוכל ליצור פרויקט מתבנית. נלחץ על אחת האפשרויות (לא רלוונטי איזו בשלב זה) ונתחיל להקליד: org.apache.maven.archetypes:maven-archetype-quickstart:archetype-quickstart



ונלחץ Create.

ולמעלה אפשר לשנות את השם של הפרויקט ואת המיקום שלו וב advanced settings ניתן לשנות את ה-Group ID, Artifact ID, ואת הגרסה בתוך Artifact Coordinates, אך ניתן להשאירם כמו שהם. פרמטרים אלו קובעים את מבנה החבילות בפרויקט כעת, עלינו להמתין שהסביבה תסיים לייצר לנו את הפרויקט ההתחלתי (שימו לב להתקדמות בסרגל הסטטוס מצד ימין). אם תופיע לכם התראה עם הכותרת Experimental Feature Alert – מומלץ לגלול למטה וללחוץ על Accept.

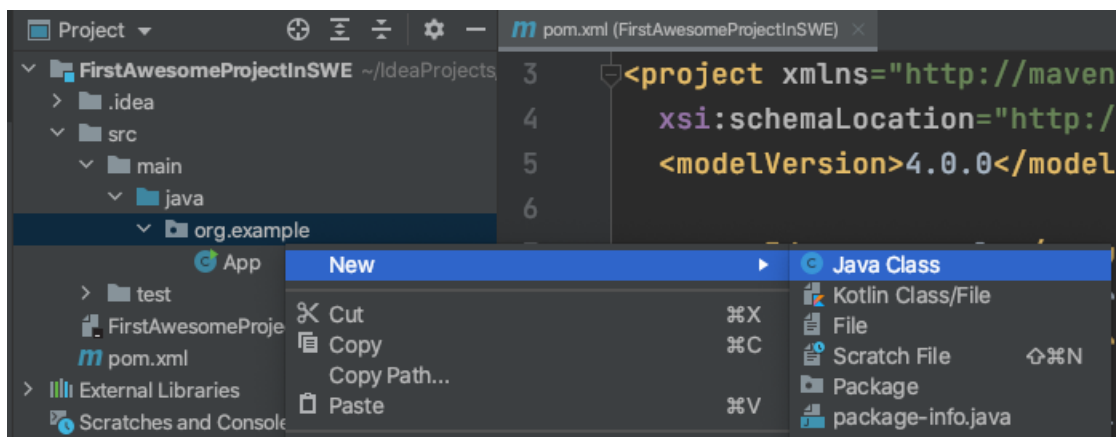
נפתח את ה-POM של הפרויקט ונשנה שתי הגדרות על מנת שנוכל להריץ בג'אווה 17 או 21 לפי ה-JDK שהורדנו:

```
<properties>
<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
<maven.compiler.source>17</maven.compiler.source>
<maven.compiler.target>17</maven.compiler.target>
```

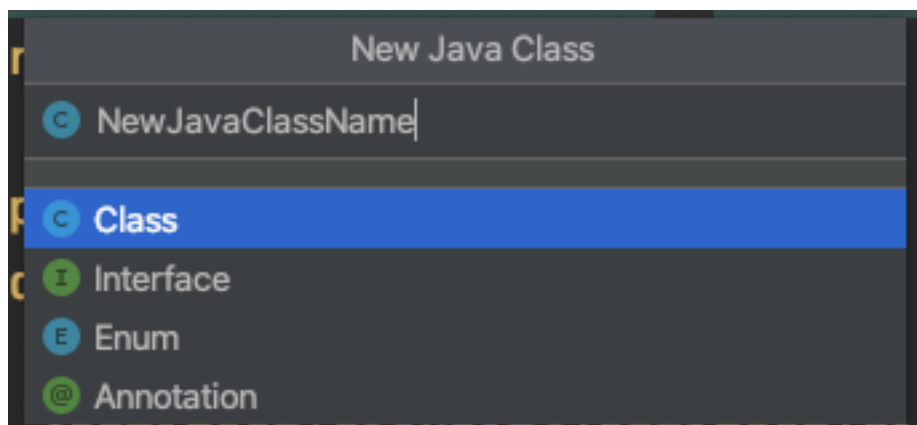
יצירת פרויקט חדש תגרום לפתיחת ספרייה חדשה בתיקייה שהוגדרה מקודם, בה ישמרו כל המחלקות שתיצרו או תוסיפו לפרויקט. נהוג להוסיף את המחלקות בתיקייה src/main/java, בחבילה ששמה זהה לדומיין, אחריו נקודה, ולבסוף שם הפרויקט (היכן שנמצאת המחלקה App בתבנית זו). בדיקות יחידה (unit tests – נרחיב עליהן במעבדה אחרת) תישמרנה, על פי רוב, בתיקייה src/test/java.

ה. יצירת מחלקה חדשה בפרויקט קיים

בחלון ה-Project Tool ננווט לחבילה שנוצרה באופן אוטומטי ונלחץ על הכפתור הימני של העכבר. נבחר באפשרות New->Java Class:



נכתוב את שם המחלקה (במקרה הזה נרצה מחלקה ולכן יש להשאיר את הבחירה על :Class



ונלחץ Enter. שימו לב ששם המחלקה הוא כשם הקובץ (ראו הרחבה למטה). בעת בחירת השמות, יש לשמור על המוסכמות: שם מחלקה מתחיל באות גדולה. הקובץ החדש יפתח אוטומטית בעורך הטקסט במרכז המסך.

מחלקות ב-Java: מחלקה יכולה להיות ציבורית (public) ולא ציבורית (ללא הרשאת גישה). בקובץ יכולה להיות מחלקה ציבורית אחת בלבד ושמה חייב להיות כשם הקובץ (למשל בדוגמה, יוצר קובץ ששמו NewJavaClassName.java ושם המחלקה NewJavaClassName). אם אין זהות בין שם הקובץ לשם המחלקה הציבורית זו תהיה שגיאת קומפילציה.

ו. עריכת מחלקה

להלן קוד המחלקה `NewJavaClassName`:

```
package org.example;  
  
public class NewJavaClassName {  
  
}
```

נוכל לייצר קוד ראשוני (boilerplate) ע"י פתיחת התפריט המתאים (בווינדוס ובלינוקס `Alt+Insert`, במק `Cmd+N`) תוך כדי שהסמן נמצא בגוף המחלקה:



בצורה כזו, נוכל לייצר בנאי. אם נרצה ליצור מתודת `main`, נוכל לכתוב `main` ולבחור באופציה המוצעת לנו, ע"י לחיצה עם העכבר או `Tab`.

נוסיף לה הדפסה של המחרוזת "Hello World!":

```
1 package org.example;  
2  
3 public class NewJavaClassName {  
4     public static void main(String[] args) {  
5         System.out.println("Hello World!");  
6     }  
7 }  
8
```

שימו לב ללחצני ה-Play בצד שמאל – נוכל להריץ ישירות את המתודה משם (אך מומלץ להריץ דרך Maven – על כך בהמשך).

לאחר הכרזה על מחלקה חדשה בפרויקט, ניתן ללחוץ בחלון הנווט על החבילה (Package) בה הוגדרה המחלקה, על מנת לאתרה ולפותח (אם לא הוגדרה חבילה היא נמצאת ב-default package).

לאחר מכן, כדי להתחיל לערוך אותה, יש להקליק פעמיים על שם המחלקה במקש השמאלי של העכבר.

שמרו את הקובץ ע"י בחירה ב-File->Save או לחיצה על הקיצור Ctrl+S.

ז. פתיחה/סגירה של פרויקט

יש לסגור את החלון הנוכחי. אם לא פתוח פרויקט אחר במקביל, סביבת העבודה תחזור למסך הראשי.

ח. מחיקת פרויקט

במסך הראשי, יש ללחוץ לחיצה ימנית על שם הפרויקט ולבחור באופציה שתציג אותו בסייר הקבצים (במק – Reveal in Finder), שם ניתן למחוק את התיקייה של הפרויקט. לאחר מכן, ניתן למחוק את הפרויקט מרשימת הפרויקטים האחרונים ע"י חזרה למסך הראשי של סביבת העבודה ובחירה באפשרות Remove from Recent Projects.

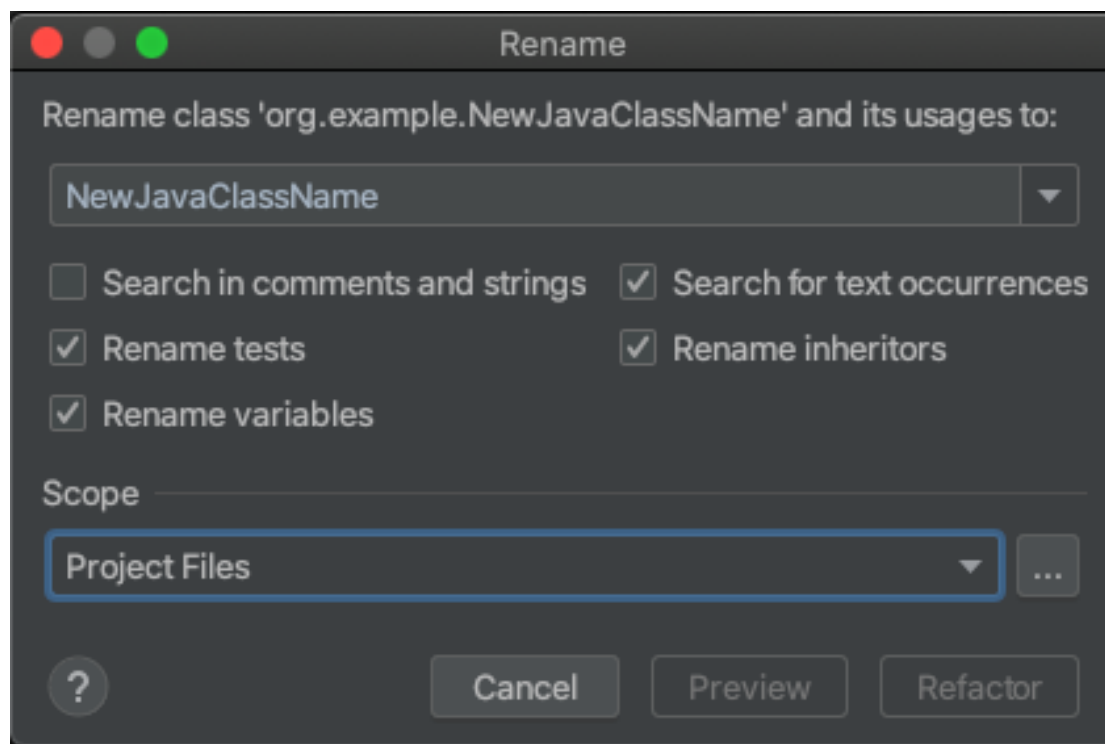
ט. פתיחת פרויקט קיים על הדיסק

במסך הראשי, נלחץ על Open. נווט לתיקייה של הפרויקט ונלחץ על Open בחלון שנפתח. שימו לב שהפעולה הנ"ל אינה מעתיקה את הפרויקט לתיקייה אחרת, כלומר כל השינויים נשמרים בתיקייה המקורית.

י. שינוי שמות מהיר (למחלקה, פעולה, פרויקט וכו')

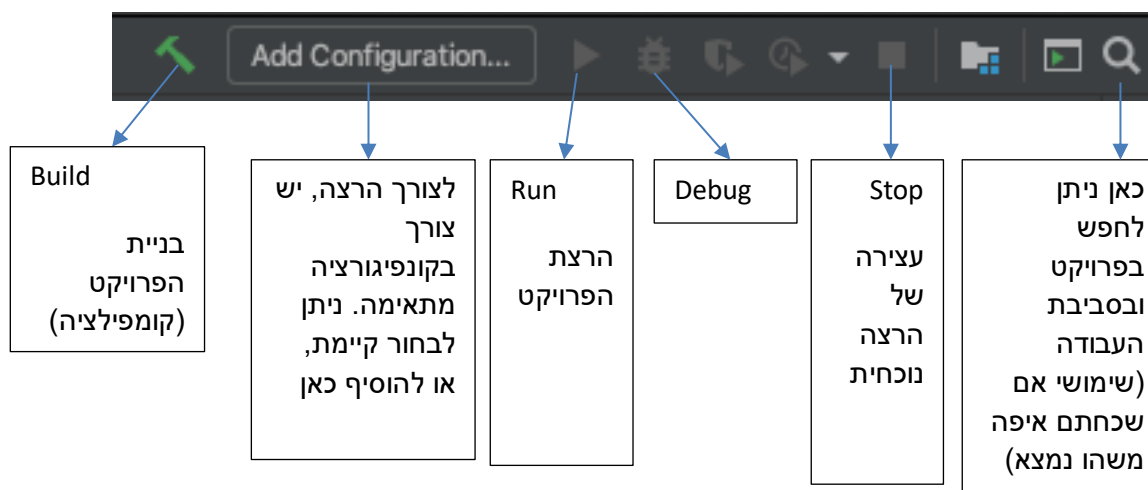
יש למקם את סמן העכבר על המשתנה/שם המחלקה/שם הקובץ בעץ של הפרויקט,

ללחוץ על הלחצן הימני של העכבר ולבחור באופציה Refactor->Rename.



מומלץ להשאיר את כל האפשרויות מסומנות, פרט לאפשרות `Search in comments and strings`. ניתן לקבל תצוגה מקדימה של השינוי בלחיצה על `Preview` ולבצעו בלחיצה על `Refactor`.

יא. סרגל הכלים הבסיסי של IntelliJ



נמצא בפינה הימנית עליונה של המסך.

יב. הידור והרצת תוכנית

ככלל, IntelliJ מספקת סביבת עבודה נוחה מאד. במהלך העריכה הקוד מקופל ושגיאות מאותרות. לדוגמה, במחלקה `NewJavaClassName` נפלה טעות בזמן העריכה ונוסף סוגר

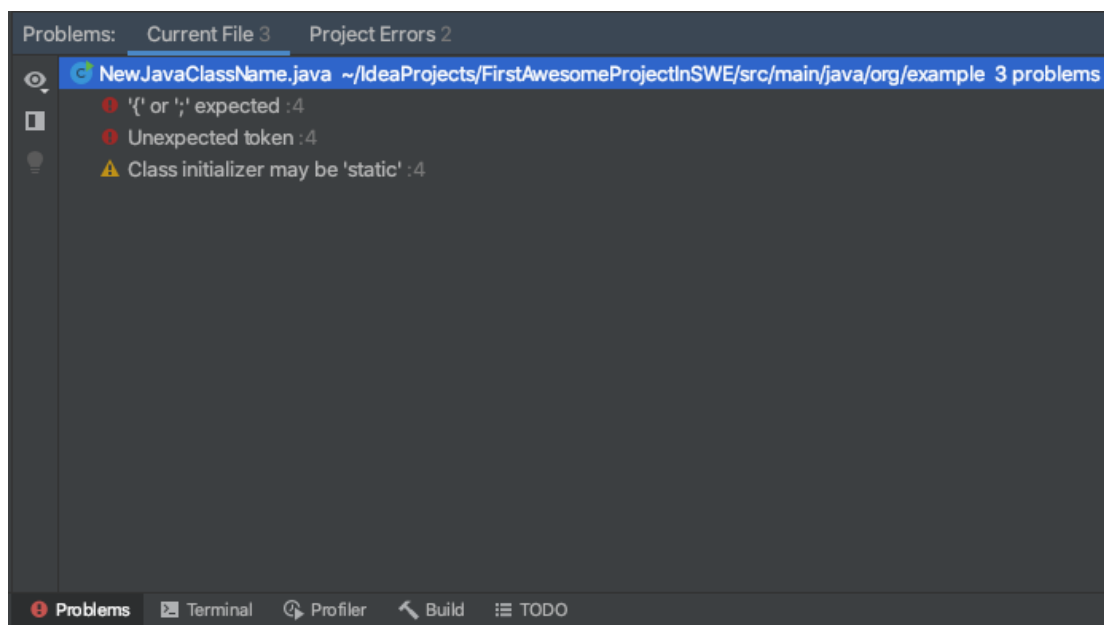
שמאלי מיותר בשורה 4. שורות 4 סומנה כשגויה. הצבעה על סימן השגיאה מציגה הסבר והמלצה לתיקונה.

```

1 package org.example;
2
3 public class NewJavaClassName {
4     public static void main(String[] args) {
5         System.out.println("Hello World!");
6     }
7 }
8

```

שימו לב שהשגיאה מופיעה גם בחלק של Problems:



על מנת להריץ, נשתמש גם כן ב־Maven. ניכנס ל־pom.xml. נמחק את האלמנט של pluginManagement בתוך build אבל נשאיר את התוכן שהיה בו. בתוך plugins -> build נוסיף plugin חדש עם המאפיינים הבאים (ניתן לעשות זאת בקיצור ע"י כתיבה של "אק" (ללא הגרשיים ולא כאלמנט) בתוך plugins ובחירה באפשרות ההשלמה האוטומטית):

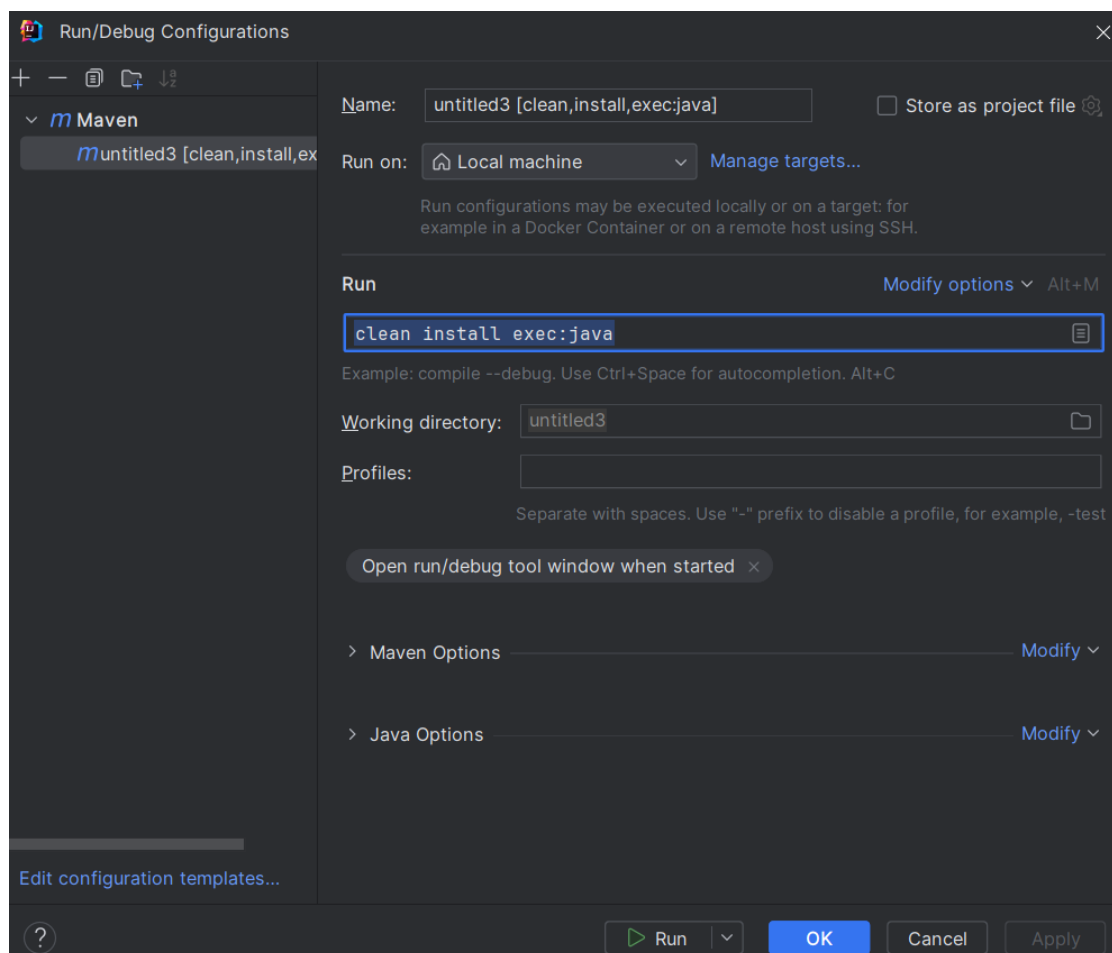
- Group Id: org.codehaus.mojo
- Artifact Id: exec-maven-plugin
- Version: 3.0.0
- נוסיף אלמנט של configuration ובתוכו אלמנט של mainClass, שיכיל את השם של המחלקה שבה ה־Main שנרצה להריץ, כולל הדומיין.

וב־XML:

```
<plugin>
  <groupId>org.codehaus.mojo</groupId>
  <artifactId>exec-maven-plugin</artifactId>
  <version>3.0.0</version>
  <configuration>
    <mainClass>org.example.NewJavaClassName</mainClass>
  </configuration>
</plugin>
```



כעת, עלינו לסנכרן את הפרויקט נכל לעשות זאת ע"י לחיצה על הסימן (מצד שמאל) שיופיע לנו אוטומטית בתוך חלון העורך. בפינה הימנית עליונה, נלחץ על Add Configuration. נלחץ על סימן הפלוס בצד שמאל למעלה של החלון שייפתח ובבחר ב־Maven. בשדה Command line נכתוב: clean install exec:java. החלון אמור להיראות דומה לזה:




ניתן גם לסמן את האפשרות של Store as project file, שתוכל להיות שימושית במקרה של עבודה של מספר אנשים על אותו הפרויקט. נלחץ על OK. החלון ייסגר, וכעת נוכל ללחוץ על כפתור ההרצה בפינה הימנית עליונה של סביבת העבודה.

```
[INFO] --- exec-maven-plugin:3.0.0:java (default-cli) @ FirstAwesomeProjectInSWE ---
Hello World!
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 4.365 s
[INFO] Finished at: 2021-10-06T12:18:08+03:00
[INFO] -----

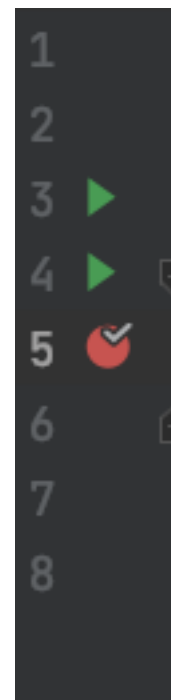
Process finished with exit code 0
```

הפלט מוצג. תוכלו להתעלם משאר ההערות שקשורות לתהליך הבנייה ול-Maven, או לחקור עליהן – מומלץ ויוכל לעזור לכם בהמשך ב-debugging.

יג. עבודה עם ה-Debugger

כדי לדבג את התוכנית, עלינו ללחוץ על הכפתור  בפניה הימנית עליונה.

כדי להריץ את התוכנית ב-Debug Mode יש לקבוע Break Points (נקודות עצירה של התוכנית) על ידי הקלקה כפולה על השטח האפור שמימין למספרי השורות בעורך.



הערה: חייבים לקבוע נקודת עצירה אחת לפחות! אחרת אין טעם לבצע debugging. כך נראית סביבת העבודה במצב debug:

שורה debug שורה אחר
(דיבוג "שטוח")
קיצור מקשים: F8

שורה debug שורה אחר
(דיבוג "עמוק")
קיצור מקשים: F7

חלון Variables מכיל את
המשתנים המקומיים הנמצאים
בהקשר הריצה הנוכחי וערכיהם.

לחצו על כפתור "Resume"
כדי להריץ את התכנית.
לעצירת הדיבוג יש ללחוץ על
הכפתור האדום.

הסבר על סביבת העבודה בפרספקטיבת Debug

התחלת הדיבוג של התוכנית

- לחיצה על כפתור החיפושית.
- debug שורה אחר שורה (debug "שטוח")
- לחיצה על מקש F8. אם השורה מכילה זימון של מתודה כלשהיא, לא תתבצע כניסה למתודה ומעקב אחר הביצוע שלה, אלא יתבצע debug של המתודה הנוכחית.
- debug שורה אחר שורה (debug "עמוק")
- לחיצה על מקש F7. אם השורה מכילה זימון של מתודה כלשהיא, ה־debugger יכנס לפעולה ויידבג גם אותה, במידה ויש גישה אליה.

מטלה 1

1. צרו פרויקט חדש בשם my-first-project.
 2. בתוך הפרויקט צרו מחלקה חדשה בשם MyFirstClass.
 3. העתיקו את הקוד הבא לתוך המחלקה שיצרתם, ושמו: טיפ: לאחר העתקה מהקובץ, כנראה שחלק מהשורות ישתבשו. מומלץ להשתמש בכלי Reformat code של סביבת העבודה (תפריט code) על מנת שהקוד יסודר לכם באופן אוטומטי.
- ```

public class MyFirstClass {

 public static void main(String[] args) {

 int a=3;
 int b=0;
 int c;
 int d=0;
 int e;

 for(int i=0; i<=5; i++){
 b+=i*2;
 System.out.println(a + " " + b + " " + i);
 }

 a=9.0;
 if(a<b)
 {
 int f=5;
 a=a+f;
 }

 else{
 a=a-f;
 }

 System.out.println(e);
 System.out.println(b);
 System.out.println(i);
 System.out.println(a/d);

 }
}

```
4. הריצו את הקוד. האם הקוד מתקמפל? אילו שורות קוד נותנות אזהרות (warnings), אילו שגיאות קומפילציה, ואילו שגיאות זמן ריצה (run time)? ממה נובעת כל אחת מהן?
  5. סמנו נקודות ביקורת על השורות הבעייתיות לדעתכם, עברו לתצוגת Debug, הפעילו את ה-debugger ודבגו את הקוד שורה אחר שורה (debug עמוק ע"י F5).
  6. בהתאם למסקנותיכם מסעיפים 4-5, תקנו את הקוד כך שהקוד יתבצע ללא שגיאות.
  7. הוסיפו לתוכנית קוד כך שהשורה הראשונה שתודפס תהיה שמכם ומספר תעודת הזהות שלכם.
- הוראות הגשה: יש לכלול בקובץ ההגשה תשובה לשאלה 4 ותמונת מסך של הפלט המתקבל מהרצת התוכנית (בגרסה האחרונה בהתאם לסעיף 7).

## חלק ב' - Maven

Maven היא כלי Build | Dependency Management לסביבת ה-JVM (Java Virtual Machine) ובפרט ל-Java. הכלי מבוסס על תקן XML.

### א. XML – eXtensible Markup Language

זהו תקן לייצוג ולקידוד נתונים במחשבים. מעבר לשימוש שנעשה בו בקורס, סביר להניח שתיתקלו בו לא פעם, ולכן כדאי להקדיש זמן להכרתו.

היחידה הבסיסית ב-XML נקראת **תגית (Tag)**. תגיות נראות כמחרוזת עם אותיות, מספרים וסימנים (אך לא חובה שיהיו כולם בכל אחת) בין הסימנים < >. למשל: <element>. כל תגית מגדירה רכיב כלשהו של המידע – למשל, תגית יכולה לייצג ספר. עלינו להקפיד לסגור תגיות – יש שתי דרכים מקובלות לעשות זאת:

- באמצעות תגית סגירה - </element>. נשים לב כי שם תגית הסגירה צריך להיות תואם לשם תגית הפתיחה. דרך זו נוחה במיוחד אם בתוך התגית יש לנו עוד אלמנטים (טקסט ו/או תגיות נוספות).
- באמצעות הוספת / לפני > - למשל: <element />. דרך זו נוחה אם אין לנו עוד אלמנטים בתוך התגית.

נראה דוגמה מעט יותר מורכבת:

```
<?xml version="1.0" encoding="utf-8"?>
<courses>
 <course>Software Engineering</course>
 <course>Discrete Math</course>
 <course>Operating Systems</course>
 <course>Making Biscuits 101</course>
</courses>
```

שימו לב לתגית הראשונה – זוהי תגית מיוחדת שנמצאת (לרוב) בתחילת המסמך, ויכולה להופיע אך ורק בתחילתו. במקרה הזה אנחנו מגדירים שני פרמטרים: גרסת ה-XML שבשימוש (ברוב המוחלט של המקרים תהיה 1.0) וקידוד המסמך (יהיה לרוב UTF-8). לאחר מכן, יש לנו רשימת קורסים ובה ארבעה קורסים שונים: הנדסת תוכנה, מתמטיקה דיסקרטית, מערכות הפעלה ו**מבוא להכנת ביסקוויטים**.

יש לנו אפשרות נוספת להגדיר מידע – הוספתו כתכונות (Attributes) לתגיות. נראה דוגמה:

```
<?xml version="1.0" encoding="utf-8"?>
<courses>
 <course lecturer="Malki Grossman">Software Engineering</course>
 <course lecturer="Or Meir">Discrete Math</course>
 <course lecturer="Rachel Kolodany">Operating Systems</course>
 <course lecturer="Random cats on YouTube">Making Cookies 101</course>
</courses>
```

במקרה הזה נוסף שם המרצה לכל קורס.

הפורמט של מסמך XML כשלעצמו אינו מוגדר, כלומר כל עוד שומרים על כללי התחביר הבסיסיים יהיה תקין. עם זאת, ישנם מספר פורמטים שבנויים בצורת XML שבהם ישנה מוסכמה על סוגי התגיות וצורת ארגון המידע.

**ב. Maven**

זהו כלי נוח לניהול של פרויקטים, שנכתב ומתוחזק ע"י קרן Apache. מייבן מאפשר, בין היתר:

- ניהול תלויות לפרויקט – פשוט מציינים שלושה פרמטרים והוא מוריד את JARs הנדרשים.
- פתיחת פרויקט מתבניות (Archetypes) שונות – נראה דוגמה לכך במעבדה 3.
- אריזת הפרויקט לקובץ JAR – ניתן לגרום לכך שיכיל את כל התלויות של הפרויקט, וכך בעצם ירוץ בכל מחשב שעליו ישנו JVM מתאים, ללא תלות בספריות חיצוניות, שייתכן שאינן מותקנות במחשב.
- יכולת שיתוף נוחה של הפרויקט, גם אם לא משתמשים באותה סביבת פיתוח

על מנת שנוכל להשתמש במייבן, עלינו להגדיר לו כיצד לעבוד – את זה עושים בעזרת קובץ POM (Project Object Model) – זהו קובץ XML שנמצא בתיקיית הפרויקט. לרוב, נתחיל לעבוד מתבנית כלשהי, ואז נצטרך רק לערוך את הקובץ. בכל זאת, נראה דוגמה:



```
<project xmlns="http://maven.apache.org/POM/4.0.0"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
 http://maven.apache.org/xsd/maven-4.0.0.xsd">
 <modelVersion>4.0.0</modelVersion>

 <groupId>com.mycompany.app</groupId>
 <artifactId>my-app</artifactId>
 <version>1.0-SNAPSHOT</version>

 <properties>
 <maven.compiler.source>13</maven.compiler.source>
 <maven.compiler.target>13</maven.compiler.target>
 <maven.compiler.release>13</maven.compiler.release>
 </properties>

 <build>
 <pluginManagement>
 <plugins>
 <plugin>
 <groupId>org.apache.maven.plugins</groupId>
 <artifactId>maven-compiler-plugin</artifactId>
 <version>3.8.1</version>
 </plugin>
 </plugins>
 </pluginManagement>
 </build>

 <dependencies>
 <dependency>
 <groupId>junit</groupId>
 <artifactId>junit</artifactId>
 <version>4.12</version>
 <scope>test</scope>
 </dependency>
 </dependencies>
</project>
```

באיזו גרסה של מייבן אנחנו משתמשים?

הגדרת הדומיין, השם והגרסה של הפרויקט (לפי הסדר)

תכונות (properties) הן מעין משתנים בקובץ ה-POM. יש כמה שמות מיוחדים – למשל, כאן אנחנו מגדירים באיזו גרסה של ג'אווה אנחנו רוצים להשתמש.

הגדרות שקשורות לבנייה של הפרויקט – ניתן לשנות ו/או להרחיב את הפעולות המתבצעות באמצעות תוספים שונים.

כאן ניתן להגדיר תלויות, כלומר ספריות תוכנה שהפרויקט משתמש בהן. בדוגמה פה הוספנו את ספריית Junit, עליה נדבר בהרחבה במעבדה אחרת.

## חלק ג' - סביבת הפיתוח JDK

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. גם IntelliJ משתמשת בכלים אלה לביצוע הידור והרצה ולכן יש להתקין את ה-JDK לפני שמתקינים את סביבת העבודה.

סביבת פיתוח זו אינה מבוססת חלונות והיא פחות נוחה אולם יש להכירה. ניתן להפעילה משורת הפקודה.

אנו נשתמש בה בעיקר לצורך הכוונת קלט פלט של תוכניות ב-Java.

### א. הגדרות נדרשות

יש להגדיר למערכת את ה-directory שבה נמצא הקומפילר javac ע"י הפקודה הבאה:

```
set path=%path%;C:\ProgramFiles\Java\jdkxxxx\bin\
```

xxxx - תלוי בגרסה שנבחרה.

ניתן לקבוע את ה-path גם ב-Windows באופן הבא:

**Control Panel → System → "Advanced System Settings"**

יש ללחוץ על כפתור "Environment Variables".

ברשימה למטה יש לבחור "System Variables" וללחוץ על Path.

שם יש ללחוץ על כפתור "Edit" ולשנות ל-

```
C:\Program Files\Java\jdkxxxx\bin\
```

ואח"כ OK (xxxx – תלוי בגרסה).

### ב. הידור התוכנית

בחלון ה-cmd יש להחליף את ה-directory הנוכחי לזה בו נמצאת תיקיית הפרוייקט ב-workspace ע"י הפקודה הבאה:

```
C:\>cd Users\Lena\workspace\projectdirectory\src\main\java
```

נניח שהדומיין שלנו הוא com.example. במקרה כזה, יש לקמפל את התוכנית על ידי

```
javac com\example\JavaWelcome.java
```

javac יוצר את הקובץ JavaWelcome.class

### ג. הרצת התוכנית

הרצת התוכנית נעשית על ידי הפקודה הבאה:

```
java com.example.JavaWelcome
```

כלומר טוענים את המחלקה JavaWelcome.class ל-JVM. אם התוכנית כוללת מספר מחלקות יש לכתוב את המחלקה בה נמצאת מתודת main ובאופן אוטומטי תיטענה כל המחלקות האחרות של התוכנית.

אם אין שגיאות בתוכנית היא תתבצע והפלט יתקבל בחלון ה-cmd.

## דוגמה:

המחלקה Mul קולטת שני מספרים שלמים, מחשבת ומדפיסה את מכפלתם.

```
import java.util.Scanner;
public class Mul {
 public static void main(String[] Args) {
 // Allow user input
 Scanner input = new Scanner(System.in);
 System.out.println("Please enter 2 integers that you want to multiply
up");
 System.out.println("First ");
 int first = input.nextInt();
 System.out.println("Second ");
 int second = input.nextInt();
 int product = first * second;
 System.out.println("The product of " + first + " and " + second + " is
" + product);
 input.close();
 }
}
```

הערה: כפי שהוסבר לעיל, שם הקובץ שמכיל את המחלקה Mul הוא Mul.java. לאחר הידור נוצר קובץ בשם Mul.class

## מטלה 2

קמפלו והריצו את תוכנית Mul משורת הפקודה.  
צרכו לדוח המעבדה תמונת מסך הכוללת את פקודת ההידור, ההרצה ותוצאותיה.

### א. הכוונת קלט/פלט

לעיתים מעוניינים להריץ תוכנית ולקבל נתונים מקובץ ו/או להדפיס תוצאות לקובץ מבלי לשנות את התוכנית. ניתן לעשות זאת על ידי הכוונת קלט/פלט.  
ב-IntelliJ ניתן לבצע הכוונה לפלט, אך לא ניכנס לכך. לתוכניות פשוטות, נשתמש בשורת הפקודה.

ניתן לבצע הכוונת קלט/פלט בשורת הפקודה באופן הבא:

ננווט לתיקייה src/main/java. נניח ששם החבילה (או הדומיין בפרויקטים של Maven) הוא com.example. נריץ כך:

**java com.example.Mul < infilename > outfilename**

שם הקובץ עם הנתונים

שם הקובץ בו ישמר  
פלט התוכנית. אם  
הקובץ קיים, תוכנו  
ימחק.

**java com.example.Mul >> outfilename**

אם הקובץ קיים, תוכנו ישמר והפלט של  
ההרצה ישורשר לסופו.

דוגמה להכוונת קלט/פלט בתוכנית Mul:

```

C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19045.2604]
(c) Microsoft Corporation. All rights reserved.

C:\Users\User\Desktop>javac Mul.java
C:\Users\User\Desktop>java Mul < input.txt > output.txt
C:\Users\User\Desktop>type output.txt
Please enter 2 integers that you want to multiply up
First
Second
The product of 3 and 4 is 12
C:\Users\User\Desktop>

```

**הידור התוכנית**

**הרצה עם הכוונת קלט ופלט**

**תוכן הקובץ output.txt לאחר הרצת התוכנית. כל מה שמודפס במהלך התוכנית נכלל בקובץ.**

**הערה:** בפרויקטים של Maven עליכם לכלול את שם החבילה המלא שאינו מופיע בצילום המסך.

## מטלה 3

יש לפתור את תרגיל מעבדה 1 המפורט בקובץ מטלת המעבדה.

## הוראות הגשה לדוח מעבדה 1:

1. ההגשה בזוגות בלבד באמצעות הגשה אלקטרונית. ניתן להגיש מספר פעמים עד לשעת הסיום.
2. יש להגיש קובץ zip אחד. שם קובץ ה-zip הוא מספרי תעודות הזהות של הסטודנטים מופרדים בגרש תחתון. ה-zip כולל שני קבצים:  
א. קובץ word ובו תשובה למטלות 1 ו-2 כמפורט להלן.  
ב. קובץ מקור (**ArithmeticApp.java**) ע"פ דרישות תרגיל מעבדה 1 כמפורט בקובץ מטלת הבית.
3. אנא הקפידו על הוראות ההגשה.

**עבודה נעימה!**