מִפְלָט לַעֲצָמִים ב-Java

מטעם קבוצת הרובוטיקה Excalibur FRC

שנת הלימודים התשפ"ה (2024 - 2025)

עורכים:
יהודה רוטשטיין
אורי קורנגוט

הוצאה לאור:
ישיבת בני עקיבא לפיד - מודיעין

פרטים:
תאריך פרסום - ספטמבר 2024
מספר גרסה - 1.0.0
מתאים לגרסת Java 8 +

תוכן עניינים

פרק ד: תכנות מונחה עצמים:

מבוא לJava ולהנדסת תוכנה

Java היא אחת משפות התכנות הפופולריות והמרתקות ביותר בעולם. היא נמצאת בכל מקום - מאפליקציות בסמארטפון שלכם, דרך מערכות מורכבות במחשבי ענק ועד לתוכנות ששולטות במכשירי חשמל ביתיים חכמים. נראה כיצד אפשר להפוך רעיונות למציאות באמצעות כתיבת קוד.

הבחירה ב-Java היא בחירה חכמה מסיבות רבות:

1. **מונחית עצמים (Object-Oriented Programming - OOP)**: שפה זו מאפשרת לכם לחשוב בתבניות של עצמים ופעולות, מה שמקל על בניית תוכנות מורכבות. תארו לעצמכם שאתם יוצרים עולם מלא באובייקטים שמתנהגים בצורה מסוימת – זה בדיוק מה שעושים ב-Java.
2. **ניידות (Portability)**: "כתוב פעם אחת, הרץ בכל מקום" - זהו אחד העקרונות המרכזיים של Java. קוד שנכתב בשפה זו יכול לרוץ על כל מחשב או מכשיר שיש לו את סביבת ההרצה של Java.
3. **ביצועים**: למרות שמדובר בשפה שמתורגמת בזמן ריצה, Java מספקת ביצועים גבוהים במיוחד בעזרת ה- Java Virtual Machine (JVM) שמתפקד כמו תרגום מיידי בין הקוד שלכם לבין המחשב.
4. **קהילה ומשאבים**: עם קהילה ענקית של מפתחים מכל העולם, תמיד תוכלו למצוא תמיכה, דוגמאות, ספריות קוד פתוח ופתרונות לבעיות שאתם נתקלים בהן.

הנדסת תוכנה היא מדע ואמנות התכנון, הפיתוח והתחזוקה של תוכנות. תחשבו על זה כמו על בניית בניין – מהרעיון הראשוני ועד לתחזוקה היומיומית. הנדסת תוכנה כוללת שלבים שונים שמבטיחים שהתוכנה תהיה אמינה, יעילה ונוחה לשימוש.

במהלך הספר, נלמד יחד את היסודות של Java ושל הנדסת תוכנה. נעבור דרך נושאים מרתקים כמו תכנות מונחה עצמים, טיפול בקלט ופלט, עבודה עם לולאות ותנאים, בניית ממשקי משתמש ועוד. בכל פרק, תמצאו דוגמאות, תרגילים ואתגרים שיעזרו לכם להפוך למתכנתים מיומנים.

בהצלחה במסע הלימוד שלכם!

מושגים בסיסיים בכתיבת תוכנה

הJDK:

- הJDK, או בשמו המורחב ה"Java Development Kit" הוא התוכנה המרכזית למפתחים בשפת JAVA.
- ה-JDK מכיל את כל הכלים שדרושים למפתחי JAVA בכתיבה של התוכנה.

הIDE:

- הIDE, הוא כלי שעוזר לנו, כמתכנתים, לתכנת בצורה יעילה יותר, בלי קשר לשפת הקוד שאנחנו עובדים איתה.
- במסגרת השימוש ב-IDE נקבל קיצורי דרך, דרכים נוחות יותר לעבוד עם קבצים, טיפים למציאת בעיות בקוד ועוד יתרונות נוספים.
- במידה וננסה לתכנת ישירות דרך JAVA, ניתקל בהרבה מאוד חסרונות משמעותיים והתכנות יהפוך לחוויה מתישה.
- כיום, מקובל אצל כל המתכנתים להשתמש ב-IDE לכתיבת קוד.
- במסגרת השיעורים שלנו, נשתמש ב-IDE שנקרא Intellij IDEA של חברת JetBrains.

Console:

- לאחר שכתבנו קוד והרצנו אותו, יפתח לנו חלון בתחתית המסך.
- החלון נקרא הConsole, ובו נוכל לראות את התוצאה של הקוד שכתבנו.

פלט:

- כאשר המחשב מוציא מידע, טקסט, או כל דבר אחר, הוא נקרא פלט.

קלט:

- כאשר המחשב מקבל מידע, טקסט, או כל דבר אחר, הוא נקרא קלט.

מילות מפתח:

- מילים אשר "מבצעות" פעולות כלשהן, לדוגמה המילה public, הגורמת לאיבר כלשהו להיות נגיש בכל הקוד שלנו ועליו נלמד בהמשך, רשימת כל מילות מפתח מופיעים בהערות (ראו עמודים 56 - 57)

פרק א':

קלט, פלט ומידע בשפת Java

הדפסת פלט
פעולות מתמטיות
הדפסת פעולות מתמטיות בשילוב עם טקסט
טיפוסי מידע
משתנים
ייבוא ספריות - Scanner
קבלת קלט
ספריית Math
ספריית Random

פלט בשפת JAVA:

נריץ את הקוד הבא:

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

ننתח את הקוד:

- שורות מספר 1,2 הן שורות שמגדירות את ה**מחלקות** ה**שיטות** שלנו. שורות 1 ו- 2, יחד עם הסוגריים המסולסלות בסיום הקוד, צריכות להופיע בכל קוד בשפת JAVA. הן מהוות מעין "מעטפת" לקוד שאנחנו כותבים. למשמעות שלהן נגיע בפרק ג', בהמשך הספר.
- נסתכל על משמעות של שורה 3:

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

בשורה הזו, אנו פוגשים את הפקודה הבסיסית ביותר בשפת JAVA. תפקידה הוא להציג פלט לתוך הConsole. כל דבר שנמצא בתוך המרכאות שבסוגריים (המרכאות (מה שבתוך ה"") לא יוצגו - ערך שמוקף במירכאות נחשב בתור טקסט. *ראה פרק סוגי מידע..*), יודפס לנו ב-Console, לדוגמא, במקרה שנריץ את הקוד נקבל את הפלט הבא:

```
Hello World!
```

שימו לב!

<u>בJava, כל שורה שמבצעת פעולה חייבת להסתיים בנקודה פסיק</u> , במידה ונכתוב קוד ללא שימוש ב ";", נקבל שגיאה והקוד לא יעבוד. במקלדת נוכל למצוא את ה";" משמאל למספר 1 ועל האות "ף".

שאלות:

1. נסו לשחק מעט עם הטקסט שבתוך המירכאות. האם הפלט השתנה כל פעם?

_____

2. השמיטו את ה; מסיום השורה, מה מופיע על המסך?

_____

3. כתבו קוד יחיד אשר ידפיס את הפלטים הבאים (שימו לב להורדת שורות), הניחו שהמחלקות והשיטות כתובות:

I am an eighth grade student

Lives in Modi'in

Studying in a yeshiva

_____

_____

_____

_____

4. מצאו את השגיאות בתוכנית הבאה:

_____

```
public class Main {
    public static void main(String[] args) {
        System.in.println(How are you?")
    }
}
```

הערה - שימושים בפקודת "Backslash" :

הפקודות, מאפשרות לנו להכניס דברים נוספים לתוך ההדפסה, מבלי להשתמש בפקודות נוספות:

\\	מכניס "\" בלי לעשות פקודה שלא רצויה
\"	מכניס ציטוט יחיד בלי לשבור את המחרוזת
\n	יורד שורה בלי פקודה חדשה
\t	מכניס לנו רווח של TAB (ברירת מחדל - 4 רווחים)

דוגמה לשימושים של פקודות Backslash:

```
public class Main {
    public static void main(String[] args) {
            System.out.println("This is a qutoe \" ");
            System.out.println("This is a tab \t ");
            System.out.println("This is a backslash \\ ");
            System.out.println("first line \n second line ");
    }
}
```

סיכום:

- הפקודה `System.out.println` היא האמצעי שלנו להדפיס פלט.
- במידה ולא נשים ; בסיום המשפט, נקבל שגיאה והקוד לא יעבוד.

פעולות מתמטיות:

אחד מהדברים הבסיסיים ביותר בשפות תכנות, ובמחשבים בכלל, הוא לבצע פעולות מתמטיות.
שפת JAVA יודעת לבצע פעולות מתמטיות על פי הסימונים הבאים:

סימון הפעולה:	שם הפעולה:	דוגמא:
+	חיבור	5+3=8
-	חיסור	5-3=2
*	כפל	3*5=15
/	חילוק	10/5=2
%	שארית ("מודולו" בשפה מקצועית)	20%3=2

רוב הפעולות המתמטיות כאן מוכרות לנו מבית הספר היסודי, אך פעולה שלא יצא לנו להשתמש בה היא שארית, (%)או מודלו בשפה המקצועית.
כאשר אנחנו מפעילים מודלו על שני המספרים, נקבל את שארית החלוקה שלהם אחד בשני.

ניתן להדפיס פלט בשילוב עם פעולות מתמטיות, בתנאי שנשמיט את המרכאות, לדוגמא, הקוד:

```
System.out.println(5+4);
```

ידפיס לנו את:

```
9
```

שימו לב, במידה ונריץ את הקוד:

```
System.out.println("5 + 4");
```

נקבל את הפלט:

```
5+4
```

- כל טקסט שיוזן בתוך המרכאות, תמיד יודפס בתור טקסט, גם אם זו פעולה מתמטית!
- כאשר נשמיט את המרכאות, נוכל להדפיס תוצאות של ביטויים מתמטיים. בהמשך נראה ביטויים ואלמנטים נוספים שיודפסו אך ורק כאשר אין מרכאות.

- לא ניתן להדפיס דברים "חסרי משמעות" (כמו טקסט) ללא השימוש במרכאות.

שאלות:

1. מה תהיה התוצאה כאשר תדפיסו כל אחד מהביטויים המתמטיים האלה? (ניתן להשתמש במחשבון)

ביטוי:	תוצאה:
21+3	
21-3+2	
92/2	
3*81	
10%3	

סיכום:

- ניתן להדפיס פעולות מתמטיות באמצעות הפקודה לפלט.
- מודולו, היא פעולה שתיתן לנו שארית.

הדפסת פעולות מתמטיות בשילוב עם טקסט:

בשפת JAVA, ניתן לשלב בשורה אחת (וכך לשמור על הקוד יעיל יותר) הדפסה של טקסט ושל פעולות מתמטיות, בעזרת הפרדה שלהן עם סימן +, לדוגמא, הקוד הבא:

```
public class Main {
     public static void main(String[] args) {
         System.out.println("5 + 5 is " + (5+5) + " Thanks!");
         }
     }
```

יחזיר לנו את הפלט:

```
5 + 5 is 10 Thanks!
```

על הפעולות המתמטיות להיות בתוך סוגריים, אחרת המחשב יתייחס אל כל מספר בפני עצמו, לדוגמא אם נסתכל על השורה:

```
System.out.println("5 + 5 is " + 5 + 5 + " Thanks!");
```

נקבל את הפלט:

```
5 + 5 is 5 + 5 Thanks!
```

- כתיבה שמשלבת בתוכה הדפסות מתמטיות וטקסט, יכולה לעתים קרובות לגרום לשגיאות. כאשר נקבל שגיאה שקשורה לבעיות בכתיבה, נקבל הודעת שגיאה שתצביע לנו על השורה בה הייתה הטעות, ונוכל לראות האם הטעות הייתה בהדפסה.

שאלות:

1. רשמו קוד אשר ידפיס את השורות הבאות. מספרים שיסומנו בירוק צריכים להופיע כטקסט ומספרים שיסומנו באדום צריכים להופיע כביטוי מתמטי. שימו לב, הסעיפים מסומנים ע"י אותיות ה-abc. הניחו שהשיטות והמחלקות כתובות.

```
a. Calculate the result of the expression 5+5
b. 10 is the subtraction of 15-5
c. 5+5 Divide into 3 with this rest 1
```

a. _____

b. _____

c. _____

סיכום:

- במידה ונרצה לשלב טקסט עם ביטויים מתמטיים ב-Java, נוכל לעשות זאת בקלות.
- שגיאות רבות יכולות להיגרם כתוצאה מהשילוב הזה, ולכן צריך לשים לב שכותבים נכון.

טיפוסי מידע:

במדעי המחשב, ישנם סוגי מידע שונים שמייצגים דברים שונים המחשב יודע לעבוד. כעת נכיר את סוגי טיפוסי המידע, ובקרוב נעבוד איתם.

שם הטיפוס:	מה מאחסן:	דוגמא:
String	טקסט	"Hello”
int	מספרים שלמים	8591 ,1 ,-1
Double	מספרים לא שלמים	3.14 ,15.53031 ,8.0
Char	יחידה בטבלת ה - ASCII	'A’, ‘b’
Boolean	אמת או שקר	True, False

על טיפוסי המידע נרחיב כאשר נלמד את המשתנים.

טבלת ה - ASCII מופיעה בהערות (ראו עמוד 58).

משתנים:

משתנה הוא כלי בתכנות המשמש לאחסון מידע שונה בזיכרון המחשב. בקוד הבא, ישנו משתנה מסוג int (מספר שלם) בשם Num שמכיל את הערך 5.

```
public class Main {
    public static void main(String[] args) {
        int Num = 5;
    }
}
```

ננתח את המבנה שלו:

- בתחילת הקוד, מופיעות המחלקות והשיטות, שלא רלוונטיות לנו.
- נגדיר את המשתנה: בשורה מספר 3, המילה int מציינת את סוג המשתנה, שהוא מספר שלם. השם אותו נתנו למשתנה הוא Num. זהו שם שנשתמש בו כדי להתייחס למשתנה הזה במהלך התוכנית. (שם זה יכול להתחלף בכל שם שתחפצו)
- הסימן = משמש להשמה. בקוד הזה, המשתנה Num מקבל את הערך 5. כלומר, נגיד שהזיכרון מוקצה עבור Num והערך שלו הוא 5.
- נשנה מעט את הקוד על מנת להבין אותו טוב יותר:

```
int Num = 5;
System.out.println(Num);
```

לאחר שנריץ את הקוד הזה נקבל את הפלט הבא:

```
5
```

כלומר, מה שעשינו, היה להגדיר מעין "מקום בזיכרון", לו קראנו Num, ולאחסן בו את המספר חמש כך שנוכל לגשת אליו כמה שרק נרצה.

- בדוגמא שהבאנו, הגדרנו משתנה מסוג int. ניתן להגדיר משתנים מכל סוגי טיפוסי המידע באופן הבא:

```
String Text = "Hey!";
int Num = 18-3;
double Onum = 19.81-0.3;
char character = 'A';
boolean Real = false;
```

מוזמנים להביט שוב בעמוד "טיפוסי המידע" ולהיזכר בסוג טיפוסי המידע.
כמו שאנחנו רואים בדוגמא, הצהרה על משתנה תמיד מורכבת בצורה הבאה:
סוג המשתנה + שם המשתנה = ערך המשתנה.

חוקי משתנים:

- בעת הגדרת ערך בוליאני (Boolean), ה-true או ה-false חייבים להתחיל באות קטנה, אחרת נקבל שגיאה.
- כאשר מצהירים על סוג המשתנה, כל סוגי המשתנים, למעט String חייבים להתחיל באות קטנה, ו-String חייב להתחיל באות גדולה.
- Java היא שפה שרגישה לאותיות גדולות, כך שמשתנה שנקרא לו Hello ומשתנה שנקרא לו hello הם שני משתנים שונים.
- כאשר נותנים למשתנה שם, השם חייב להיות מורכב מאותיות בלבד, ללא רווחים. במידה ורוצים לתת למשתנה שם בעל יותר ממילה אחת, נגדיר אותו בעזרת הפרדה בין מילים באותיות גדולות, למשל:
המשתנה real number יהיה בעת הכתיבה realNumber.
- מקובל להתחיל משתנים באות קטנה.
- שם המשתנה יכול להתחיל מאותיות ולאחר מכן לשלב מספרים, למשל: num1.
- בעת הגדרת משתנה מסוג int או double נוכל לעשות פעולות מתמטיות בהגדרת המשתנה עצמו!
- בString נוכל לכתוב כל ערך שאנחנו רק רוצים, אך במידה ונכתוב מספרים בתוך String נקבל מצב בו java לא תתייחס אליהם כמספרים אלא כטקסט ולא נוכל להפעיל עליהם פעולות מתמטיות.

העשרה:

לטיפוס מידע מסוג Double קוראים כך משום שאליו מיוחסים 2 בייטים (יחידת מידע בסיסית) במקום מספר שלם שאליו מיוחס רק אחד. לכן מספר שלם יכול להגיע רק עד 2,147,483,647 ואילו Double יכול עד המספר 1.797E+308, נסו להוסיף 1 למספר המרבי של INT ותקבלו שגיאת מספר גדול מדי.

לאחר שהצהרנו על המשתנה, נוכל לערוך אותו בכל שלב בקוד, על ידי קריאה אליו ועדכון שלו בצורה הבאה:

```
String Text = "Hey!";
Text = "Hello";
```

במידה ונדפיס את הקוד הזה, נקבל את הפלט Hello, מכיוון שערכנו את המשתנה.

- ניתן להצהיר על משתנה בלי לתת לו שום ערך, במטרה לתת לו ערך מאוחר יותר, באופן הבא:

```
int num;
num = 5;
```

- ניתן להצהיר על יותר ממשתנה אחד בפעם אחת, בהנחה שנותנים להם אותו ערך או שלא נותנים לשניהם ערך, באופן הבא:

```
int num, num1 = 2;
int num2, num3;
```

שאלות:

1. הקיפו את האפשרויות הנכונות:

num18%	שם תקין למשתנה / שם לא תקין למשתנה
String	שם תקין למשתנה / שם לא תקין למשתנה
11num	שם תקין למשתנה / שם לא תקין למשתנה
Is it	שם תקין למשתנה / שם לא תקין למשתנה
?IsIt	שם תקין למשתנה / שם לא תקין למשתנה

2. רשמו בכתיבה נכונה את השמות הבאים למשתנים האלו, כפי שמקובל לכתוב אותן:

Big Number	_____
Small Number	_____
Big Number	_____
Very good	_____

3. התאימו כל ערך לטיפוס מידע שלו:

8	boolean
8.5	double
Hey	char
A	int
true	String

4. תן דוגמא למקרה בו צריך להשתמש בטיפוס int ומקרה בו צריך להשתמש בטיפוס double:

_____

_____

סיכום:

- משתנים הם הדרך שלנו לאחסן מידע בזיכרון של המחשב ולגשת אליו.
- ישנן הוראות ברורות כיצד נכון לקרוא למשתנים, לפי מה שמקובל היום בעולם התכנות.
- ניתן להגדיר כמה משתנים בו זמנית.
- ישנם 5 טיפוסי מידע שונים שנלמד עליהם.

ייבוא ספריות:

בשפת Java, ייבוא ספריות הוא תהליך בו מבוצעת הכנסת ספריה חיצונית (external library) לתוך קובץ הקוד שלנו, ספרייה זאת, היא קוד שמישהו אחר כתב בשביל שאנחנו נשתמש בו כדי להשתמש בפקודות נוספות שהספריה הזאת מציעה לנו.

על מנת לייבא ספריה נשתמש בפקודה Import, ולאחר מכן נכתוב את שם הספריה, באופן הבא:

```
import java.util.Scanner;
```

בקוד הזה, ייבאנו את הספריה Scanner, ונוכל להשתמש בפקודות נוספות שהיא מציעה לנו.

בעמודים הבאים נשתמש בספריה Scanner, ובהמשך הספר נכיר ספריות נוספות.

הערה -

בJAVA תמיד נתעדף לכתוב את פקודות ה - Import לפני כל קוד שכתבנו, לכן נכתוב את השורה הזאת בשורה 1, אפילו לפני השיטות והמחלקות.

קבלת קלט:

עד כה, למדנו איך ניתן לבצע פעולת שיתבצעו בלי השפעה של המשתמש על התוכנה. כעת נלמד איך ניתן לקבל קלט מהמשתמש.

```
import java.util.Scanner;
public class Main {
    public static void main(String[] args) {
      Scanner in = new Scanner(System.in);
      int age = in.nextInt();
      System.out.println("You are " + age "Years Old")
    }
}
```

ننתח את התוכנית:

- שורה ראשונה - ייבאנו את הספרייה לתוכנית שלנו.
- שורה שנייה ושלישית - הגדרת השיטות במחלקה.
- שורה רביעית - בשורה זו אנחנו מגדירים אובייקט מסוג Scanner ששמו של האובייקט הוא in, לא נכנס לעניין של אובייקטים כאן (הרחבה בפרק ד'), הניחו ששורה זאת היא "אבן הבניין" של הקלט. נרחיב על כך מאוחר יותר.
- שורה חמישית - ניצור משתנה שאליה נרצה לייחס את הערך שהמשתמש נותן לנו, המשתמש ישתמש ב CONSOLE בשביל להזין ערכים. המשתנה שלנו הוא age והוא מסוג INT (מספר שלם), הערך של משתנה זה הוא הקלט שהמשתמש נתן לנו.

לScanner יש גם את הדרך לקבל סוגי מידע שונים ולא רק מספרים שלמים, בשביל סוגי מידע שונים צריך פעולה קצת אחרת -

String Name = in.nextLine();	String (מחרוזת)
Boolean isTrue = in.nextBoolean();	Boolean (נכון או לא נכון)
Double Time = in.nextDouble();	Double (מספר עשרוני)

שאלות:

1. קבלו מהמשתמש שני מספרים שלמים והדפיסו את סכומם. הניחו שהמחלקות והטיפוסים כתובים.

_____

_____

_____

_____

2. בקשו מהמשתמש להזין שם וגיל והדפיסו את ההודעה "שלום [שם], בן [גיל]". הניחו שהמחלקות והטיפוסים כתובים.

_____

_____

_____

_____

3. בקשו מהמשתמש להזין 5 ציונים והדפיסו את הממוצע שלהם. הניחו שהמחלקות והטיפוסים כתובים.

_____

_____

_____

_____

4. בקשו מהמשתמש להזין אורך ורוחב של מלבן והדפיסו את ההיקף והשטח שלו. הניחו שהמחלקות והטיפוסים כתובים.

_____

_____

_____

_____

5. בקשו מהמשתמש להזין שם עובד, מספר שעות עבודה ושכר שעתי, והדפיסו את השכר הכולל של העובד לפי הנוסחה: מספר שעות עבודה * שכר שעתי. הניחו שהמחלקות והטיפוסים כתובים.

_____

_____

_____

_____

ספריית Math

ספריית Math - ספרייה שבה אפשר להשתמש בהרבה פעולות מתמטיות פשוטות וגם מתקדמות מאוד, בספרייה זה יש לנו הרבה פעולות שנוכל להשתמש.

- ספריית Math מיובאת כבר בJDK שלנו (הרחבה במושגים בסיסיים למתכנת) בחבילת java.lang אז לא נצטרך לייבא אותה בעצמו בעזרת Import אלא נוכל ישר לכתוב קוד.

פעולה	שימוש
`Math.max(x,y)`	מחזירה את הערך הגבוה מבין שני מספרים
`Math.sqrt(x)`	מחזירה את השורש של אותו מספר
`Math.random()`	מחזירה מספר רנדומלי מבין 0.0 ל1.0 כולל
`Math.pow(x, y)`	מחזירה את המספר הראשון בחזקת השני
`Math.round()`	מחזירה את הערך המלא הקרוב ביותר
`Math.abs()`	מחזירה את הערך המוחלט של אותו משתנה

- פעולות שלא נצטרך השנה:

פעולה	שימוש
`Math.asin(a)`	מחזירה את הערך ההפוך של פונקציית הסינוס (arcsine)
`Math.cosh(k)`	מחזירה את הערך של הפונקציה ההיפרבולית של קוסינוס
`Math.atan(x)`	מחזירה את הערך ההפוך של פונקציית הטנגנס

כל הפעולות האלו מחזירות ערכים כלומר בשביל שנראה תוצאה נצטרך להדפיס את הפעולות.

```
import java.util.Math;
public class Main {
    public static void main(String[] args) {
        int biggerNum = Math(3,65);
        System.out.println(biggerNum); // 65 ידפיס
        }
    }
```

ספריית Random

ספריית Random, מאפשרת לנו לקבל מספרים רנדומליים לחלוטין בתוך הקוד שלנו.

נביט בקוד הבא:

```
import java.util.Random;
public class Main {
    public static void main(String[] args) {
        Random rand = new Random();
        int num = rand.nextInt(100);
        System.out.println(num);
        }
    }
```

קוד זה ידפיס לנו בכל פעם מספר שונה לחלוטין, למרות שהקוד נשאר זהה לחלוטין, ואין שום התערבות מצד המשתמש. ננתח את הקוד:

- בשורה מספר 1, ייבאנו את הספריה Random אל תוך הקוד שלנו.
- בשורות 2,3 הגדרנו את המחלקות והשיטות שלנו.
- בדומה לתהליך בו אנחנו מקבלים קלט מהמשתמש, אנחנו מגדירים כאן אובייקט. את משמעות השורה נבין בהמשך הספר.
- נגדיר את ערך המשתנה בדומה לקלט, רק שנכתוב rand. בתוך הסוגריים נכתוב את המספר המקסימלי ביותר אותו אנחנו רוצים לקבל. (לדוגמה 0 - 100)
- לאחר מכן, נדפיס את המשתנה.

שאלות:

1. כתבו קטע קוד שמקבל מספר מהמשתמש, לוקח מספר רנדומלי, ומחשב את ההפרש ביניהם. הניחו שהשיטות והמחלקות כתובות.

_____
_____
_____
_____
_____
_____
_____

2. כתבו תוכנית שלוקחת 2 מספרים רנדומליים ומדפיסה את הגדול מבינהם, השתמשו ב2 ספריות.

_____
_____
_____
_____

_____
_____
_____
_____

סיכום פרק מספר 1: קלט, פלט ומידע בשפת Java.

רשימת פקודות שנלמדו בפרק:

מבנה קוד בסיסי בשפת JAVA.	`public class Main {`
	`  public static void main(String[] args) {`
	`  }`
	`}`

משמשת להדפסה ב-Console.	`System.out.println();`

סוגי המשתנים השונים והדרך להדפיס אותם. תבנית ליצירת משתנה.	`String Text = "Hey!";`
	`int Num = 18-3;`
	`double Onum = 19.81-0.3;`
	`char character = 'A';`
	`boolean Real = false;`

תבנית לייבוא ספריות.	`import java.util.Scanner;`
יצירת Scanner לטובת קבלת input.	`Scanner in = new Scanner(System.in);`
תבנית לקבלת input מהמשתמש.	`int age = in.nextInt();`
יצירת rand.	`Random rand = new Random();`
תבנית לקבלת rand.	`rand.nextInt();`

פרק ב':

תנאים ולולאות

תנאי בסיסי

תנאי בוליאני

שימוש ב"או" בתנאי

שימוש ב"גם" בתנאי

תנאי מסוג "אחרת" (Else)

לולאת While

לולאת Do While

תנאי מתקדם בלולאת While

לולאת For

לולאות מקננות

לולאה מקננת מורכבת

Switch & Case

תנאי בסיסי

תנאים הם הדרך האופטימלית בשביל אלגוריתם מצויין. עם תנאי נוכל לבצע חלקי קוד (מונח מקצועי - "בלוק") כשנרצה רק אם תנאי מסוים קיים.

דוגמה לתנאי שנדפיס - נבדוק האם a גדול מ-b:

```
public class Main {
    public static void main(String[] args) {
            int a = 5;
            int b = 7;
            System.out.println(a > b) //false
        }
    }
```

התנאים שיש:

a < b	האם a גדול מ b
a > b	האם a קטן מ b
a == b	האם a שווה ל b, הבהרה למטה.
a != b	האם a **לא** שווה ל b
a >= b	האם a גדול **ושווה** b, כלומר אם שניהם שווים ל5 הביטוי יהיה נכון, במידה והיינו כותבים רק גדול מ - 5 הביטוי יהיה שלילי. (הביטוי יכול להיות גם קטן ושווה)

סימן '==' בשפות תכנות:

בשפות תכנות, יש שימוש בשני סימני שוויון == ולא בסימן אחד = כדי להבדיל בין השמה (assignment) לבין השוואה (comparison). הבחנה זו מונעת בלבול ושגיאות בקוד, ומאפשרת לקומפיילר להבין האם הכוונה היא להציב ערך במשתנה או להשוות בין שני ערכים.

כך כל פעם שנרצה לכתוב ביטוי שמשווה בין 2 ערכים, נייחס שני סימני שוויון!

כתיבת בלוק קוד-

תבנית:

```
if (condition){
      System.out.println("Hello, World!");
                        .
                        .
                        .
}
```

בדוגמה הנ"ל נוכל להכניס תנאי לתוך הסוגריים וגם להכניס קוד לתוך הסוגריים המסולסולות כך שהקוד יבוצע רק אם התנאי נכון.

דוגמה -

```
Scanner scanner = new Scanner(System.in);
int age = scanner.nextInt();
if (age > 18){
      System.out.println("You are an Adult");
}
```

שאלות:

1. כתבו תנאי המקבל מהמשתמש קלט שבוליאני (Boolean) שבודק אם המשתמש חבר מועדון, אם כן הדפיסו פלט מתאים, אם לא הדפיסו פלט מתאים.

_____
_____
_____
_____
_____
_____
_____
_____

שימוש ב -"או" ו ב - "גם" בתנאי

בהרבה מקרים נרצה להשתמש בתנאי המורכב מעוד תתי תנאים, לדוג' נרצה לדעת אם גילו של המשתמש בין 13 ל-18 שנים, במקרה זה נצטרך להשתמש בסימן "גם" - (&&) כך נכתוב -

```
int age = Scanner.nextInt();
if (age > 13 && age < 18){
      System.out.println("You are a Teenager")
```

```
}
```
// הגדרנו שני תנאים וחיברנו אות במילת ה "וגם" כך שרק אם שני תנאים שלידו נכונים, הקוד שבתוך התנאי יבוצע.

כך גם במילת ה "או" שתבצע את הקוד רק אם **לפחות** אחד מהתנאים נכונים, סימן ה"או" נכתוב ככה "||", שהוא שני קווים אנכיים. להלן דוגמה שבודקת אם צרכן זכאי להנחה:

```
int age = 65;
boolean isMember = false;
if (age >= 60 || isMember)
      System.out.println("Eligible for a discount.");
```

העשרה:

כאשר בתנאי יש שורת קוד יחידה שמבוצעת אם התנאי נכון, לא צריך להוסיף סוגריים מסולסלות, כנ"ל לגבי else (אחרת - עמוד הבא).

שאלות:

2. כתבו תנאי המורכב גם מ || וגם עם **&&.**

_____
_____
_____
_____
_____
_____
_____
_____

תנאי מסוג "אחרת" (Else)

כאשר יש לנו תנאי, והתנאי הזה לא נכון, אז הבלוק קוד לא ירוץ. במצב כזה, נרצה לבצע פעולה אחרת. לדוגמה:

נניח שיש לנו משתנה בשם מספר, ואנחנו רוצים לבדוק אם הוא גדול מ-5. אם התנאי מתקיים (כלומר, אם המספר גדול מ-5), נדפיס הודעה מתאימה. אם התנאי לא מתקיים (כלומר, אם המספר קטן או שווה ל-5), נדפיס הודעה אחרת.

```
int number = 3;

if (number > 5)
      System.out.println("Number is bigger than 5");
```

```
 else
     System.out.println("Number is smaller than 5");
```

עוד דוגמה:

```
double price = 49.90;
double AvialableMoney = 89.50;
if (price < AvialableMoney)
 System.out.println("You Have enough money");
else
 System.out.println("You Don't Have Enough Money");
```

תנאים מקוננים

תנאים מקוננים הם תנאים המופעלים בתוך תנאים אחרים. הם מאפשרים לנו לבנות לוגיקה מורכבת על ידי שילוב מספר תנאים יחד. תנאים מקוננים שימושיים במיוחד כאשר יש צורך לבדוק סדרה של תנאים בצורה מסודרת ומובנית.

דוגמה לתנאים מקוננים

נניח שאנחנו רוצים לבדוק את הציונים של תלמיד ולהחליט אם הוא עובר או נכשל, ואם הוא מצטיין או לא. נשתמש בתנאים מקוננים כדי לבדוק זאת:

```
int score = 85;
boolean isExtraCreditCompleted = true;
if (score >= 60) {
     System.out.println("Passed");
      if (score >= 90) {
```

```
            System.out.println("Excellent");
      } else if (score >= 80) {
            System.out.println("Very Good");
            if (isExtraCreditCompleted) {
                  System.out.println(" Extra Credit"); }
                   } else { System.out.println("Good"); }
                  } else { System.out.println("Failed"); }
```

מהלך התוכנית -

1. התנאי הראשון בודק אם הציון הוא מעל 60. אם כן, התלמיד עבר.
2. בתוך התנאי הראשון יש תנאי מקונן שבודק אם הציון הוא מעל 90. אם כן, התלמיד מצטיין.
3. אם הציון הוא בין 80 ל-90, התנאי הבא בודק אם התלמיד קיבל ציון "Very Good" ואם השלים את המשימה הנוספת.
4. אם אף אחד מהתנאים לא מתקיים, התלמיד מקבל ציון "Good".
5. אם הציון הוא פחות מ-60, התלמיד נכשל.

דוגמה נוספת -

```
boolean isAdmin = true;
boolean isLoggedIn = true;
boolean hasAccessRights = false;
if (isLoggedIn) {
      if (isAdmin) {
            System.out.println("Access granted: Admin");
            } else {
            if (hasAccessRights) {
                  System.out.println("Access granted: User");
                   } else {
                   System.out.println("Access denied: Insufficient
            rights");
                  }
            }
      } else {
      System.out.println("Access denied: Not Loged");
       }
```

שאלות:

1. הסבירו את מהלך התוכנית הנ"ל.

_____
_____
_____
_____
_____
_____
_____
_____

2. מה ההבדל בין - = ל - ==, הסבירו את השימוש של שניהם.

_____
_____
_____
_____

3. כתבו תוכנית שמקבלת שני מספרים a ו-b, ובודקת אם a שווה ל-b. אם הם שווים, התוכנית תדפיס "a is equal to b". אם הם לא שווים, התוכנית תדפיס "a is not equal to b".

_____
_____
_____
_____

4. כתבו תוכנית שמקבלת משתנה בשם age ומשתנה בשם grade. אם הגיל הוא בין 18 ל-25 והציון הוא מעל 80, התוכנית תדפיס "Eligible for the program". אחרת, התוכנית תדפיס "Not eligible for the program".

_____
_____
_____
_____

5. כתבו תוכנית שמקבלת שני משתנים: temperature (טמפרטורה) ו-humidity (לחות). אם הטמפרטורה היא מעל 30 מעלות והלחות היא מעל 70%, התוכנית תדפיס "It's a hot and humid day". אחרת, התוכנית תדפיס "The weather is normal".

_____
_____

_____
_____

6. כתבו תוכנית שמקבלת משתנה בשם ציון (ערך מספרי בין 0 ל-100). אם הציון הוא 60 ומעלה, התוכנית תדפיס "עבר". אחרת, התוכנית תדפיס "נכשל". הניחו שהשיטות ומחלקות כתובות.

_____
_____
_____
_____

לולאות While

הערה: לפני הקריאה, מומלץ לחזור על תנאים.

לולאת While, היא הדרך שלנו לגרום לפעולות לקרות שוב ושוב, ולהתנות את הקיום שלהם בתנאי. נסתכל על הקוד הבא:

```
public class Main{
    public static void main(String[] args) {
        int i = 1;

        while (i <= 5) {
            System.out.println(i);
            i++;
        }
    }
}
```

ננתח את הקוד:

- בתחילת הקוד, הגדרנו את השיטות והמחלקות שלנו.
- בשורה מספר 3, הגדרנו משתנה i, אותו הגדרנו ל1.
- לולאת הwhile מזכירה תנאי במבנה שלה, לאחר רישום מילת המפתח while, הקלדנו את התנאי, שכל עוד הוא מתקיים התוכנית תרוץ. התנאי שהגדרנו הוא "כל עוד i קטן או שווה ל5".
- בתוך ה{}, הגדרנו לתוכנית להדפיס את i, ולעלות את i באחד.

כלומר, לאחר שנריץ את קוד זה נקבל את הפלט הבא:

```
1
2
3
4
5
```

- בעזרת הביטוי i++, ניתן להעלות את i באחד. במקום לרשום "i= i+1".

שאלות:

1. קבל מהמשתמש מספר, והדפס את כל המספרים הזוגיים שקטנים ממנו (עד ל-0). הנח שהשיטות והמחלקות כתובות.

_____
_____
_____
_____
_____
_____
_____
_____

2. קבל מהמשתמש מספרים זוגיים, ועצור את התוכנית כאשר המשתמש מזין מספר אי זוגי. בסיום התוכנית הדפס את סכום כל המספרים. הנח שהשיטות והמחלקות כתובות.

_____
_____
_____
_____
_____
_____
_____
_____

3. כתוב תוכנית הקולטת מספרים, כאשר יוקלד המספר 0, התוכנית תעצור ותדפיס את המספר הגבוה ביותר שהוזן.

_____
_____
_____
_____
_____
_____
_____
_____

העשרה:

לשפה ++C קוראים כך כי היא נועדה להיות שדרוג לשפת C. סימן ה-**++** מסמל פעולה של הוספה (increment) בשפות תכנות רבות (גם בשלנו - Java), ומשמעותו בשמה של ++C היא שהיא "צעד אחד קדימה" לעומת שפת C. כלומר, מדובר בשפה המציעה יכולות נוספות ושיפורים על פני שפת C.

לולאות For

לולאות for היא לולאה שרצה לרוב מספר קבוע של פעמים. ללולאת For יש שימושים רבים, ואותם נכיר בהמשך יחד עם טכניקות חדשות לשימוש בלולאה.

```
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        for (int i = 0; i < 6; i++) {
            System.out.println(i);
        }
    }
}
```

ננתח את הקוד:

- בתחילת הקוד, הגדרנו את השיטות והמחלקות שלנו.
- בשורה מספר 3, כתבנו את מילת המפתח "for", ובתוך הסוגריים רשמנו שלוש פקודות שונות:
  - הגדרנו משתנה i מסוג int, המשתנה רלוונטי רק לתוך לולאת ה-for.
  - הגדרנו מעיין תנאי i קטן משש. שימו לב לשימוש ב"קטן" ולא ב"קטן שווה".
  - לאחר מכן, רשמנו i++, שמעלה את i באחד.
- בתוך הלולאה, אנחנו מדפיסים את לולאת ה-while.

- לולאת ה-for שכתבנו, מאתחלת בהתחלה משתנה i, שניתן לשימוש רק בתור לולאת ה-for, ולאחר מכן מגדירה את התנאי לקיום של הפונקציה, כל עוד i קטן משש.
- לאחר כל ריצה, כלומר כל הדפסה, הלולאה מעלה את i באחד, כך שבסוף נגיע למצב בו i=6, והלולאה תעצור.
- הרצת הקוד תדפיס את הפלט הבא:

```
0
1
2
3
4
5
```

לולאות מקננות

לולאה מקננת היא לולאה שמופעלת בתוך לולאה אחרת. במילים אחרות, מדובר בלולאה אחת שנמצאת בתוך גוף הלולאה השנייה. לולאות מקננות שימושיות במיוחד כאשר אנו רוצים לבצע חישובים על מערכים דו-ממדיים, או כאשר יש צורך בביצוע פעולות שחוזרות על עצמן בצורה מסודרת.

דוגמה:

נניח שאנחנו רוצים להדפיס לוח כפל מ-1 עד 10. נשתמש בלולאות מקננות כדי להשיג זאת:

```
for (int i = 1; i <= 10; i++) {
      for (int j = 1; j <= 10; j++) {
            System.out.print(i * j + "\t");
 }
      System.out.println();
}
```

בדוגמה זו:

1. הלולאה החיצונית פועלת על המשתנה i ומתחילה ב-1 עד 10.
2. הלולאה הפנימית פועלת על המשתנה j ומתחילה גם היא ב-1 עד 10.
3. בתוך הלולאה הפנימית, מבוצעת פעולה של הדפסת תוצאת הכפל של i ו-j.
4. לאחר סיום כל מחזור של הלולאה הפנימית, מודפס מעבר שורה (System.out.println()) כדי להתחיל שורה חדשה.

שימושים נפוצים בלולאות מקננות:

עבודה עם מערכים דו-ממדיים.

הדפסת דפוסים (כמו משולשים, ריבועים וכו').

חישובים מרובים בהם כל שלב תלוי בתוצאה הקודמת

הערה:

בשביל משתני "אינדקס" נשתמש באותיות: `i, j, k`

לולאות מקננות מורכבות

כרגע בקורס פתיחה הזה לא יהיה לנו שימוש בלולאות מורכבות מקננות כי הם לא שימושיות בחומר שנלמד.

שימושים של לולאות מקננות מורכבות -

1. הדפסות מטריצות תלת מימדיות
2. דפוסי גרפיקה

3. ניתוח נתונים
4. הסתברויות מתקדמות

דוגמה עם לולאה מקננת נוספת, שמדפיסה את כל הצירופים של המספרים 1, 2, 3:

```
for (int i = 1; i <= 3; i++) {
      for (int j = 1; j <= 3; j++) {
            for (int k = 1; k <= 3; k++) {

      System.out.println("a=" + i + ", b=" + j + ", c=" + k);

            }
      }
}
```

שאלות:

1. מה תפקידה של כל חלק בסוגריים של הלולאה.

_____
_____
_____
_____

2. כיצד מגדירים לולאת While ומה תפקידה.

_____
_____
_____
_____

Switch & Case

הפקודה `switch` משמשת לביצוע בחירה מבין מספר רב של אפשרויות על סמך הערך של משתנה. זה מאפשר לכתוב קוד קריא ומסודר יותר כאשר יש לנו מספר רב של תנאים להשוות.

דוגמה:

ככה ננסה לבדוק איזה יום זה בשבוע בלי הרבה תנאים.

```
int day = 2; // נניח שהיום הוא יום שני

switch (day) {
      case 1:
```

```
            System.out.println("Sunday");
            break;
        case 2:
            System.out.println("Monday");
            break;
                                .
                                .
                                .
        default:
            System.out.println("Unknown Day");
             break;
}
```

הערה:

במבנה `switch`, הקטע `default` הוא קטע קוד שמבוצע כאשר אף אחד מהערכים שבמקרה (cases) לא תואם לערך של הביטוי הנבדק. זה דומה לקטע `else` בתנאי `if-else`. השימוש ב-`default` מאפשר לטפל במקרים לא צפויים או ערכים שאינם מתאימים לאף אחד מהמקרים שהוגדרו.

שאלות:

1. כתוב תוכנית המדפיסה את המפרים 1 -9 בעזרת לולאת For

_____

_____

_____

_____

_____

2. כתוב תוכנית בג'אווה שמחברת את כל המספרים מ-1 עד 100 ומדפיסה את הסכום.

_____

_____

_____

3. קבל קלט מהמשתמש שהוא ציון מספרי של מבחן (1- 100) ותדפיסי את הציונים בתור A,B,C,D,F בעזרת Switch - Case.

פרק ג':

פונקציות ומערכים

מערכים חד מימדיים

סריקת מערך חד מימדי בעזרת לולאה

לולאת For - Each

מערכים דו מימדיים

פונקציות בסיס

פונקציות מתקדמות עם מילות מפתח

פרמטרים בפונקציות

החזרה בפונקציות

מערכים חד מימדיים

מערכים הם "קבוצה" או "אוסף" של ערכים מאותו סוג, לדוגמה מערך יכול להיות המספרים 1- 10 או הפירות והירקות האהובות עליכם, לכל מערך יש שם וגם "תאי זיכרון" לכל ערך, את תאי הזיכרון צריך להגדיר ביחד עם שם המערך. בוא נראה דוגמה:

```
int[] array = new int[5];
```

הגדרנו פה את שם המערך וסוגו, בדוגמה הזאת שמו של המערך היא array (מערך באנגלית) ואת סוגו, שהוא int - מספרים שלמים, אם נכניס למערך ערך שהוא לא int תהיה לנו שגיאה.
הגדרנו גם את כמות תאי הזיכרון שתהיה למערך פה למשל תהיה 5 תאים.

במידה ונרצה להכניס את כל הערכים שלנו בהתחלה נוכל לעשות כך-

```
int[] array = {1, 2, 3, 4, 5};
```

גישה לערכים (Elements):

בשביל שנוכל לגשת לערכים שבמערך נוכל להשתמש בסוגריים מרובעות

```
int firstElement = array[0];
int secondElement = array[1];
System.out.println(firstElement + secondElement) // 3
```

דגש:

תמיד בכל מצב בJava ובכלל בשפות תוכנה אחרות, ה - Index (הערך) ההתחלתי הוא **0** ולא **1** כמו שמקובל בספירת base 10 (ראו בהערות).
לכן **0** יהיה הערך הראשון, **1** השני, וכן הלאה.

דוגמאות נוספות:

בחירת חברת רכב אהובה -

```
String[] array = {"BMW", "Ford", "Ferrari"};
String firstElement = array[0];
 System.out.println("My Favorite Car is a " + firstElement);
```

בחירת נכון או לא נכון

```
 boolean[] array = {true, false};
boolean firstElement = array[1]; System.out.println(firstElement);
// false
```

רשימת מספרים מיוחדים.

```
double[] array = {3.141, 2.718, -1.0};
double firstElement = array[2]; System.out.println(firstElement); //
-1.0
```

שאלות:

1. כתבו תוכנית שיוצרת מערך של ציוני תלמידים ועוד מערך של שמות התלמידים.

_____

_____

_____

_____

_____

_____

סריקת מערך חד מימדי בעזרת לולאה

בהרבה מקרים נרצה לדעת את כל התכולה של מערך מסוים, לדעת מה הערך הגבוהה ביותר או סתם למיין את המערך, במקרים האלה נשתמש בלולאת For בשביל לעבור כל הערכים שבמערך, להלן דוגמה שסורקת את המערכת ומוצאת את הערך הגדול ביותר בו -

```
int[] array = {1,2,3,4,5,6,7,8,9,};
int max = array[0];
for (int i = 1; i < array.length; i++){
      if (array[i] > max) {
            max = array[i];
      }
}
 System.out.println("Maximum value: " + max);
```

מהלך התוכנית:

- הגדרנו מערך שמכיל בתוכו מספרים שלמים.
- הצבנו במשתנה עזר את הערך הראשון של המערך.
- בעזרת לולאת For וגם בעזרת הפעולה `Array.length` קיבלנו את הגודל של המערך ועברנו על כל תא זיכרון.
- בכל פעם שהלולאה עברה בדקנו האם המספר כרגע ברשימה (ה i מסמן את המיקום הנוכחי), ובדקנו אם הוא גדול יותר מהמספר עזר (שכרגע המקום הראשון) אם המספר גדול יותר אז הוא מחליף את מספר העזר וממשיך, אם בהמשך המערך יהיה מספר גדול יותר הוא יחליף את המספר הנוכחי במשתנה עזר, ככה עד לסוף המערך \ לולאה.
- הדפסנו את משתנה העזר שהוא המספר הגדול ביותר.

שאלות:

1. כתבו תוכנית שיש פה 2 מערכים של פירות ושל ירקות, קחו ערך רנדומלי מכל מערך והציגו למשתמש שיבחר איזה אחד הוא אהב יותר, הציגו את הפלט הזה. כתבו בעזרת המנחה.

_____

_____

_____

_____

_____

_____

לולאת For - Each

לולאת `for-each` (או לולאת "enhanced for") היא לולאה מיוחדת המאפשרת לעבור על כל האלמנטים במערך או באוסף (collection) בצורה פשוטה ונוחה. היא מציעה דרך קריאה ונוחה לעבור על כל האלמנטים בלי הצורך להתעסק באינדקסים ידניים.

מבנה הלולאה -

```
for (type element : array){
// Code Block
}
```

type - סוג האלמנט במערך או באוסף.
element - משתנה זמני שמחזיק את הערך הנוכחי בלולאה.
array - המערך או האוסף שעליהם עוברים.

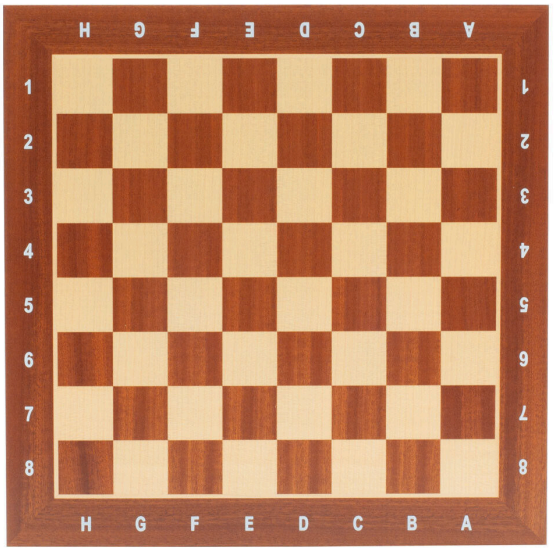
דוגמה המדפיסה את כל המספרים במערך:

```
int[] numbers = {1, 2, 3, 4, 5};
for (int number : numbers) {
      System.out.println(number);
}
```

הערה:
לולאת for-each מיועדת בעיקר לקריאה של ערכים ולא לשינויים בהם. כלומר, אם תנסה לשנות ערך של אלמנט במערך או באוסף באמצעות המשתנה הזמני בתוך הלולאה, השינוי לא יתבצע על הערך המקורי במערך או באוסף, אלא רק על ההעתק המקומי של אותו ערך. אם יש צורך לשנות את הערכים, עדיף להשתמש בלולאה רגילה עם אינדקס.

מערכים דו מימדיים

מערכים דו מימדיים הם הדרך לאחסן בתוך מערך, יותר ממימד אחד של מידע. נביט בלוח השחמט:



בלוח השחמט, אנחנו מכירים את ה"קוארדינטות" בצדדי הלוח. אנחנו יודעים שניתן לסמן משבצת כ-H8, ניתן לסמן משבצת כ-A1 וכו'. כך בנויים גם מערכים דו מימדיים.

נביט בקוד לדוגמא של מערך דו מימדי:

```
public class Main {
    public static void main(String[] args) {
        int[][] array = new int[10][10];
        for (int i = 0; i < array.length; i++) {
            for (int j = 0; j < array.length; j++) {
                array[i][j] = 0;
            }
        }
    }
}
```

בשונה ממערך רגיל, השתמשנו פעמיים בסוגריים מרובעים, ובהם הגדרנו את הגודל של המערך, לדוגמא, עכשיו הגדרנו מערך בגודל 10X10, בכך שרשמנו [10][10].

על מנת לגשת לתא של מערך, ניגש באמצעות התבנית הבאה:

```
array[i][j] = 0;
```

כאשר i ו-j הן הקוארדינטות של התא אליו אנו רוצים לגשת.

באתחול של מערך דו מימדי, יש צורך בשימוש בלולאה מקוננת, על מנת לגשת אחד מהאינדקסים, גם כאשר הוא דו מימדי. לדוגמא, בלולאה הזאת:

```
    for (int i = 0; i < array.length; i++) {
```

```
        for (int j = 0; j < array.length; j++) {
            array[i][j] = 0;
        }
    }
```

עוברים על כל אינדקס ומאתחלים את הערך שלו ל0.

ניתן להגדיר באותו עיקרון מערכים עם יותר מימדים, על ידי הוספת סוגריים מרובעות, אך זה לא רלוונטי ולא בחומר הקורס.

שאלות:

1. כתוב תוכנית בג'אווה שמשתמשת בלולאת `for-each` כדי לחשב את סכום המספרים במערך שלם.

_____
_____
_____
_____
_____
_____
_____

2. כתוב תוכנית בג'אווה שמשתמשת בלולאת `for-each` כדי לחשב את האורך הכולל של כל המחרוזות במערך של מחרוזות.

_____
_____
_____
_____
_____

פונקציות בסיס

פונקציה (Function) או **מתודה** (Method) היא בלוק קוד שירוץ כל פעם שנקרא לפונקציה. כלומר, לכל פונקציה יש בתוך את ההוראות שלה לדוגמה להדפיס את המספר 5, לכל פונקציה יש גם שם משלה.

דוגמה לפונקציה:

```
public static void Greet(){
     System.out.println("Hello");
}
Greet();
Greet();
Greet();
```

ננתח את התוכנית -

- הגדרנו פונקציה עם המילים public, static, void שאותם נסביר בהמשך.
- קראנו לפונקציה שלנו בשם Greet עם סוגריים מסולסלות.
- בתוך הפונקציה יש פעולה אחת והיא להדפיס שלום.
- קראנו לפונקציה 3 פעמים.

פלט -

```
5
5
5
```

פונקציות מתקדמות עם מילות מפתח

בפונקציות בג'אווה, יש שלוש מילות מפתח חשובות שצריך להבין:

- **public**
- **static**
- **void**

נסביר כל אחת מהן:

public

ב-Java ובשפות תכנות רבות אחרות, יש דגש על אבטחה ופרטיות של משתנים, פונקציות ומחלקות. המילה public, או בעברית "ציבורי", מציינת שהגישה לערך (משתנה, מחלקה או פונקציה) שהיא מגדירה יכולה להיעשות מכל מקום בקוד, אפילו ממחלקות שונות. כלומר, אם פונקציה מוגדרת כ-public, ניתן לקרוא לה מכל מקום בתוכנית.

static

המילה static בג'אווה מציינת שמשאב מסוים (כגון פונקציה או משתנה) שייך למחלקה עצמה ולא לאובייקט ספציפי של המחלקה. פונקציה סטטית יכולה להיקרא בלי ליצור אובייקט של המחלקה. זה מאוד שימושי כאשר יש פונקציה כללית שנרצה לגשת אליה ישירות.

לדוגמה, פונקציה סטטית בשם main היא נקודת ההתחלה של כל תוכנית ג'אווה. היא מוגדרת כסטטית כדי שניתן יהיה לקרוא לה מבלי ליצור אובייקט של המחלקה שמכילה אותה. (על מחלקות ואובייקטים נלמד בפרק הבא)

void

המילה void מציינת שסוג החזרה של הפונקציה הוא ריק. כלומר, הפונקציה לא מחזירה ערך כלשהו. פונקציות שצריכות לבצע פעולה מסוימת מבלי להחזיר תוצאה מוגדרות כ-void. (בהמשך נלמד על החזרה של פנקציות)

פרמטרים בפונקציות

פרמטרים בפונקציות הם משתנים המועברים לפונקציה בעת הקריאה לה. הם מאפשרים לפונקציה לקבל קלט ולבצע פעולות שונות בהתאם לערכים המועברים לה.

הצהרה של פרמטרים:

כשמגדירים פונקציה, ניתן להגדיר את הפרמטרים שלה בתוך הסוגריים העגולים אחרי שם הפונקציה. כל פרמטר כולל סוג נתון ושם.

תבנית -

```
public static myMethod(type var1, type var2 ...){
                .
                .
                .
}

myMethod(var1 -> type , var2 -> type ...)
```

בדוגמה להלן אנחנו מגדירים פונקציה שמקבלת 2 משתנים מסוג int ותדפיס את הסכום שלהם.

```
public static int add(int num1, int num2){
      System.out.println(num1 + num2)
}
add(3,5) // 8
```

דוגמה נוספת:

דירים פונקציה שמקבלת 2 משתנים מסוג int ותחזיר את הסכום שלהם.

```
public static int greet(String name){
      System.out.println("Hello, " + name)
}
greet("Yogev")
```

דוגמה מתקדמת:

בתוכנית להלן נקבל מערך ומקדם (Factor) לתוך פונקציה שתדפיס את כל האלמנטים כפול המקדם.

```
public static void main(String[] args) {
      int[] numbers = {1, 2, 3, 4, 5};
      multiplyArray(numbers, 2);
      for (int number : numbers) {
```

```
            System.out.println(number);
            }
        }
public static void multiplyArray(int[] array, int factor) {
      for (int i = 0; i < array.length; i++) {
            array[i] *= factor;
        }
}
```

שאלות:

1. כתבו פונקציה בשם `calculateRectangleArea` שמקבלת שני מספרים שלמים (אורך ורוחב) ומדפיסה את שטח המלבן.

_____
_____
_____
_____
_____
_____

2. כתבו פונקציה בשם `isEven` שמקבלת מספר שלם ומדפיסה `true` אם המספר זוגי ו-`false` אם לא.

_____
_____
_____
_____
_____
_____

החזרה בפונקציות

פונקציות בשפת Java יכולות להחזיר ערכים מסוגים שונים למקום שבו הן נקראות. פעולה זו מתבצעת באמצעות מילת המפתח `return`. כאשר פונקציה מחזירה ערך, היא מסיימת את ביצועה ומחזירה את הערך למקום שבו היא נקראה.

כאשר מגדירים פונקציה שמחזירה ערך, יש לציין את סוג הערך שהיא מחזירה בהצהרת הפונקציה. אם הפונקציה לא מחזירה ערך, יש להשתמש במילת המפתח `void`.

דוגמה:

```
 public static int multiply(int a, int b){
     c = a * b;
     return c
}
int result = multiply(3,5);
System.out.println("The Answer is - " + result);
```

דוגמה נוספת:

```
int[] numbers = {1, 5, 3, 9, 2};
int max = findMax(numbers);
System.out.println("The maximum value is: " + max);

public static int findMax(int[] array) {
     int max = array[0];
     for (int i = 1; i < array.length; i++) {
          if (array[i] > max) {
               max = array[i];
          }
     }
        return max;
}
```

שאלות:

1. כתבו פונקציה בשם `printNumbersInRange` שמקבלת שני מספרים שלמים ומדפיסה את כל המספרים בטווח ביניהם. (השתמשו בלולאת For)

_____
_____
_____
_____
_____
_____

2. כתבו פונקציה בשם `celsiusToFahrenheit` שמקבלת טמפרטורה במעלות צלזיוס ומחזירה את הטמפרטורה במעלות פרנהייט.

_____
_____
_____

_____

_____

_____

3. כתבו פונקציה בשם `calculateAverage` שמקבלת שלושה מספרים ממשיים ומחזירה את הממוצע שלהם.

_____

_____

_____

_____

_____

_____

פרק ד':

תכנות מונחה עצמים (OOP)

מושג האובייקט
יצירת Classes ואובייקטים
יצירת פעולות בתוך האובייקט
יצירת בנאי - Constructor
מערכים של אובייקטים
תורשה
פולימורפיזם
מערכים של אובייקטים
פונקציות גנריות

מושג האובייקט

אובייקט הוא אחד מהיסודות המרכזיים בתכנות מונחה עצמים (Object Oriented Programming או OOP). האובייקט מייצג חפץ בעולם האמיתי או רעיון מופשט.

אובייקט הוא מופע של מחלקה (Class). כל אובייקט שנוצר מתוך מחלקה מסוימת חולק את אותם מאפיינים והתנהגויות של האובייקטים האחרים שנוצרים מאותה מחלקה , אך לכל אובייקט יש ערכים ייחודיים משלו למאפיינים הללו. לדוגמה, אם ניצור שני אובייקטים מתוך המחלקה "מכונית", כל אחד מהם יוכל להיות בצבע שונה או מדגם שונה, אך שניהם יהיו מסוג "מכונית" ויוכלו לבצע את אותן פעולות כמו להניע את המנוע או לבלום.

השימוש באובייקטים מאפשר לנו לבנות תוכניות מורכבות ומאורגנות יותר. כל אובייקט מכיל את כל המידע והפעולות הדרושות לו, מה שמאפשר להסתיר את הפרטים הפנימיים שלו ולהגן עליהם משינויים בלתי רצויים.

1. **הסתרת מידע** : מאפשרת להסתיר את המצב הפנימי של האובייקט ולחשוף רק את הפונקציונליות הנדרשת.
2. **שימוש חוזר בקוד**: ניתן ליצור מחלקות ולהשתמש בהן מחדש בפרויקטים שונים.
3. **תחזוקה קלה**: חלוקה לאובייקטים ברורים ומובנים מקלה על התחזוקה והשדרוג של התוכנה.
4. **פולימורפיזם** (Polymorphism): מאפשר להשתמש באובייקטים מסוגים שונים בצורה אחידה.
5. **הורשה** (Inheritance): מאפשרת ליצור מחלקות חדשות המבוססות על מחלקות קיימות, ובכך לשפר ולשדרג את הקוד בצורה יעילה.

שימוש באובייקטים:

כאשר נשתמש באובייקטים בתוכנית שלנו, נוכל לחשוב עליהם כעל יחידות עצמאיות שמבצעות פעולות מסוימות או מחזיקות מידע מסוים. לדוגמה, אם נבנה מערכת לניהול ספריה, נוכל ליצור אובייקטים מסוג "ספר", "מחבר" ו"מנוי". כל אחד מהאובייקטים הללו יוכל להכיל מידע ופעולות הרלוונטיים לו:

- **ספר**: יכיל פרטים כמו שם הספר, שם המחבר, ותאריך הפרסום. הוא יוכל לבצע פעולות כמו השאלת הספר והחזרתו.
- **מחבר**: יכיל פרטים כמו שם המחבר, תאריך לידה, ורשימת ספרים שכתב. הוא יוכל לבצע פעולות כמו הוספת ספר חדש לרשימה שלו.
- **מנוי**: יכיל פרטים כמו שם המנוי, כתובת, ותאריך ההצטרפות. הוא יוכל לבצע פעולות כמו השאלת ספר והחזרת ספר.

הגישה לאובייקטים אלו ולפעולותיהם תאפשר לנו לבנות מערכת ניהול ספריה מסודרת, קריאה ותחזוקתית.

יצירת Classes ואובייקטים

בשביל ליצור מחלקה חדשה עקבו אחרי ההוראות הבאות (בשביל IntelliJ IDEA):

- ננווט לתיקייה שלנו
- נלחץ לחיצה ימנית
- נבחר ב - new
- נבחר ב - Java Class
- נקרא למחלקה בשם רצוי - לדוגמה "Car"
- נלחץ על ENTER

בצורה זאת ניצור אובייקט והמחלקה שלו במסמך שונה:

```
public class Car {
    public String model = "Ford";
    public String color = "Blue";
    public int year = 2018;
```

מהלך התוכנית:

שורה 1: הגדרנו מחלקה "פומבית" עם מילת המפתח class ושם המחלקה.

שורות 2 - 4: יצרנו שלושה משתנים של המחלקה שהם פומביים, אז נוכל לגשת אליהם גם מהמחלקה Main

תבנית גישה מ Main:

```
ClassName ObjectName = new ClassName(); // תבנית
Car myCar = new Car(); // myCar - הוא האובייקט
```

גישה למשתנים ופעולות של האובייקט:

```
Car myCar = new Car();
System.out.println(myCar.model);
System.out.println("I Car Color is " + myCar.color);
System.out.println("I Bought My Car " + (2025 - Car.year) + "Years
Ago");
```

יצירת פעולות בתוך האובייקט:

בתוך מחלקה נוכל להכין פעולה (פונקציה) שיכולה לעשות מגוון רחב של פעולות, את הפונקציה הזאת נסמן בתור public בשביל שנוכל לגשת אליה דרך האובייקט ממחלקת Main.

דוגמה:

כך ניצור 2 פונקציות, אחת בשביל לחשב בן כמה הרכב שלנו ושנייה תדפיס את הפרטים שלו.

```
public class Car {
    public String model = "Ford";
    public String color = "Blue";
    public int year = 2018;

public int getCarAge(){
     return 2025 - year;
}

public void getDetails(){
System.out.println(model + "" + year + "" + color)
}
```

שימוש במחלקה Main:

```
Car myCar = new Car();
System.out.println(myCar.getCarAge());
myCar.getDetails();
```

בפונקציות נוכל לעשות פעולות get / set, פעולות אלו כמו שנשמעות גורמות או לנו להציב מחדש ערך במחלקה (set) או לקבל פרטים (get).

```
public void setYear(int newYear){
     year = newYear;
}
public String getModel(){
     return model;
}
```

שימוש במחלקת Main:

```
Car myCar = new Car();
myCar.setYear(2023);
String carModel = myCar.getModel();
```

שאלות:

1. כתבו מחלקה בשם `Student` עם מאפיינים `name` (שם התלמיד) ו-`grade` (ציון). לאחר מכן, צרו אובייקט מסוג `Student` והדפיסו את פרטי התלמיד.

_____
_____
_____
_____
_____
_____
_____
_____
_____

2. כתבו מחלקה `LibraryBook` עם מאפיינים `title` (שם הספר) ו-`isAvailable` (האם הספר זמין להשאלה). הוסיפו פונקציות `borrowBook` ו-`returnBook` שמשנות את מצב הזמינות של הספר.

_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____

יצירת בנאי - Constructor

כאשר נרצה להתאים מאפיינים ייחודיים לאובייקט כשניצור אותו לדוגמה אם ניצור שלושה רכבים כל אחד אם שנה מיחדות בה יוצר, צבע מסויים ואף דגמים שונים. נוכל להשתמש ב"בנאי" או Constructor, בשביל ליצור קונסקרטור.

בקונסטרקטור אנחנו נשתמש במילת המפתח `this`, המילה `this` מאפשרת לנו להתייחס למשתנה בתוך המחלקה שלנו, אם נרצה לדוגמה, לקבל משתנה מחוץ למחלקה , לדוגמה צבע, אז נשתמש במילה `this.ObjectVariable` בשביל להתייחס לצבע מכונית של האובייקט שלנו במחלקה שלנו.

נבנה קונסטרקטור וניצור את האובייקט המתאים בתבנית הזאת:

```
public ClassName(type Var, type Var2 ...){
      }
---------
ClassName myObject = new ClassName(Var:type, Var2:type ...);
```

בתוך הסוגריים שלנו נשים את המאפיינים של המחלקה שלנו בדיוק כמו פרמטרים בפונקציה. עם המאפיינים האלו נוכל לייחס אותם לאובייקט שלנו. שימו לב שהצבעים מותאמים למשתנים הנכונים.

להלן דוגמה:

```
public class Car {
      private String model;
      private int year;
      private String color;

      public Car(String model, int year, String color) {
            this.model = model;
            this.year = year;
            this.color = color;
      }

      public int getYear() {
       return year;
       }
}
```

מהלך התוכנית:

נגדיר את שלושת המשתנים - "המאפיינים" של אותו אובייקט - המכונית.

אנחנו "בונים" קונסטרקטור כמו בתבנית שלמדנו, ובתוך המאפיינים כשנרצה שהמשתמש יזין את מודל המכונית, צבע המכונית והשנה שבה היא נוצרה.

בתוך הקונסטרקטור, נייחס לכל אחד המאפיינים שלנו **שהוגדרו לפני הקונסטרקטור** (רא את הצבעים המתאימים). אל כל אחד מהמשתנים האלו נייחס את הערכים המתאימים **שבתוך הסוגריים של הקונסטרקטור בדיוק כמו פונקציה**.

ככה נוכל להשתמש במאפיינים שהמתמש נתן לנו.

לדוגמה - ליצור פונקציה שמחזירה את השנה שבה נוצרה המכונית.

דוגמה ליצירת אובייקט:

```
Car car1 = new Car("Ferrari", 2023, "Yellow");
System.out.println(car.getYear()); // 2023
```

שאלות:

1. הכינו מחלקה, קורנסטרקטור ואובייקט של "אדם" שיש לו את המאפיינים "שם" ו"גיל", הכינו 2 פונקציות פנימית שמחזירות את המאפיינים.

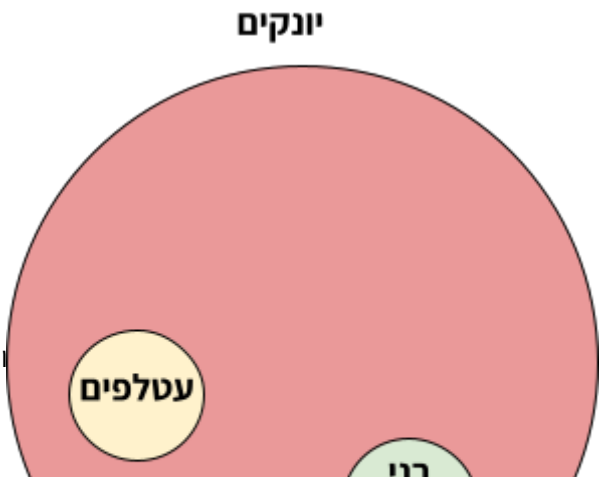
_____
_____
_____
_____
_____
_____
_____
_____

2. צרו את האובייקט של המחלקה והשתמשו בפונקציות הפנימיות ככל שתחפצו.

_____
_____
_____
_____

תורשה בין אובייקטים

תורשה, זה אחד האלמנטים המתקדמים בתכנות-מונחה-עצמים.
על מנת להבין את עיקרון התורשה, נביט על דוגמא מהחיים האמיתיים.
נסתכל על היונקים. כידוע, שכל בני האדם וגם כל העטלפים מכילים את **כל** התכונות של היונקים, אך לא כל היונקים מכילים את התכונות של בני האדם.
כלומר, קיימות תכונות קבועות אצל יונקים, שנשארות אצל כל היונקים, אך בתוך "תתי המשפחות" של היונקים הם משתנים.
לדוגמא, בשרטוט אחר, כל תכונה מסומנת בצבע:



בעזרת הכלי תורשה בשפת JAVA, ניתן להעביר תכונות בין אובייקטים.
נגדיר אובייקט "יונק" (mammal): (נניח כתובות לו פעולות get וset)

```
public class Mammal {
    public int Weight;
    public int Age;
    public int Height;

    public Mammal(int Weight, int Age, int Height) {
        this.Weight = Weight;
        this.Age = Age;
        this.Height = Height;

   public void eat() {
        System.out.println("Yum");
    }
   }
```

בנוסף, נגדיר עוד שני אובייקטים, "אדם" ו"עטלף". מכיוון שגם בני אדם וגם עטלפים הם יונקים, נוסיף להם את מילת המפתח extends שתגרום להם לקבל את התכונות של היונקים.
המחלקה אדם:

```
public class Human extends mammal{
    public String name;
    public String gender;
    public String country;

    public Human(String name, String gender, String country, int
Weight, int age, int height) {
        super(Weight, age, height);
        this.name = name;
        this.gender = gender;
        this.country = country;
```

```
    }
}
```

כעת, הוספנו את המילה super, שתגרום לכך שנוכל לקבל פרמטרים מהמחלקה mammal , ולהשתמש בפונקציות יחודיות למחלקה mammal. כך, השורה הבאה:

```
Human Ori = new Human("Ori", "Male", "Israel", 3, 5,1);
 Ori.eat();
```

תקינה, למרות שאנו משתמשים בפונקציות שנכתבו בmammal.

במחלקה עטלף, לא נשתמש במילת המפתח super, וכך למרות הגדרת התורשה לא נוכל לנצל אותה:

```
public class bat extends Mammal{
    public String food;
    public String type;

    public bat(String food, String type) {
        this.food = food;
        this.type = type;
    }
}
```

שאלות:

1. כתבו מחלקה בסיסית בשם Person עם מאפיינים name ו-age. צרו מחלקה יורשת בשם Student שמוסיפה את המאפיין studentId.

_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____

_____

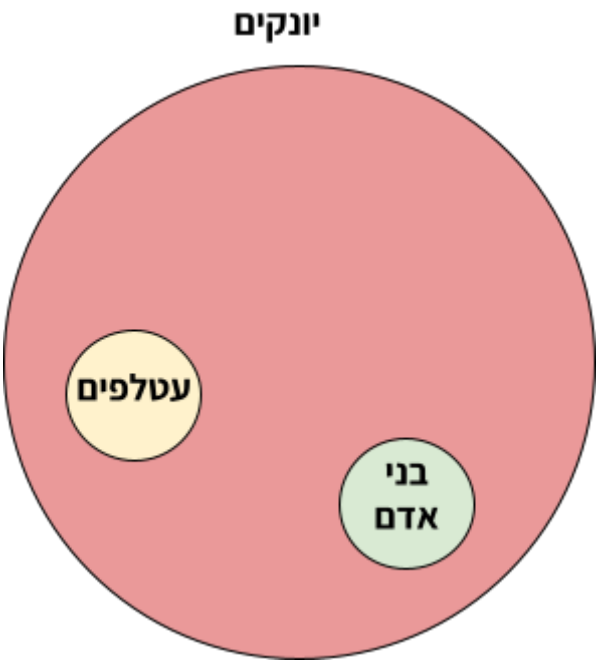
_____

_____

_____

פולימורפיזם

פולימורפיזם, הוא עיקרון נוסף בתכנות-מונחה-עצמים המבוסס על העיקרון הבא:
נחזור לגרף שלנו:



אנחנו יודעים, שבני אדם ועטלפים הם גם יונקים, ולכן, בעזרת פולימורפיזם ניתן ליצור אובייקט מסוג Human ומסוג Bat על ידי שימוש במילת המפתח mammel. כלומר:

```
mammal Ori = new Human("Ori", "Male", "Israel", 3, 5,1);
Ori.eat();
```

הקוד הזה, לא יחזיר לנו שגיאה, על פי עיקרון התורשה.S
שימו לב, שכן נצטרך להצהיר שאנו יוצרים בן-אדם, ולכן כן נצטרך בהמשך לציין שמדובר בבן-אדם.

מערכים של אובייקטים:

בדומה למערכים של משתנים, ניתן ליצור ולהגדיר מערך של אובייקטים בעזרת התבנית הבאה:

```
Human[] humans = new Human[2];
```

```
 humans[0] = new Human("Alice", "Female", "USA", 60, 30, 160);
humans[1] = new Human("Bob", "Male", "Canada", 70, 35, 170);

for (Human human : humans) {
      System.out.println("Name: " + human.getName());
      System.out.println("Gender: " + human.getGender());
      System.out.println("Country: " + human.getCountry());
      System.out.println("Age: " + human.getAge());
      System.out.println("Height: " + human.getHeight());
      human.eat();
      human.speak();
}
```

בקוד הזה, הגדרנו שני אובייקטים מסוג אדם, ולאחר מכן בלולאת for עברנו על כל אחד מהאובייקטים שנמצאים במערך אדם, והדפסנו את התכונות של כל אחד מהם.

כך, עבודה עם מערכים של אובייקטים, מזכירה מאוד עבודה עם רשימות בלולאת for.

פונקציות גנריות

פונקציות "גנריות" הם הדרך שלנו לכתוב גרסה אחת של פונקציה שדיכולה ללהתעסק עם יותר מטיפוס מידע אחד כפרמטר, לדוגמה אם נרצה להדפיס תכולה של ארבעת המערכים הבאים בעזרת פונקציה.

```
Integer[] IntArray = {1,2,3,4};
Double[] DoubleArray = {1.2, 1.3, 1.4, 1.5};
Character[] CharArray = {'A', 'B', 'G'};
String[] StringArray = {"HELLO", "My", "Name", "Is"};
```

<u>שימו לב לשימוש של טיפוסי מידע פרימטיביים!</u>

נרצה להדפיס את כל התכולה של כל מערך בעזרת לולאת for - each, נראה זאת בעזרת דוגמה.

```
public static void DisplayArray(Integer[] Array){
      for (Integer x:Array){
            System.out.print(" "+x);
      }
}
public static void DisplayArray(Double[] Array){
}

public static void DisplayArray(Character[] Array){
}

public static void DisplayArray(String[] Array){
}
```

נראה כי ההדוגמה לעיל היא לא יעילה, ולא מתאימה את הסטנדרט שלנו (הערה 6 - DRY), היא עושה את אותה פעולה רק צריך לשנות את הטיפוס מידע, לפיכך נוכל לייחס טיפוס מידע **גנרי** כך שיעזור לנו.

לפני הדוגמה נרצה להגדיר סטנדרט לקריאת השם של הפנקציה הגנרית, הדרך המקובלת היא לקרוא לטיפוס מידע בשם T , בקורס זה נוכל במקום לקרוא לו המילה Thing בשביל להקל על כתיבת הקוד.

תבנית לכתיבת פונקציה גנרית:

```
public static <T> void Myfunc(T MyArray){
      for (T i : MyArray){
            System.out.println(x);
      }
}
```

נראה כי כתבנו לפני מילת הvoid את שם הטיפוס מידע שאנחנו יוצרים, לאחר מכן הצבנו פרמטר מאותו סוג טיפוס מידע שיצרנו.
לאחר מכן כתבנו לולאת for-each בשביל להדפיס את תכולת המערך.

שאלות:

1. כתבו פונקציה גנרית המחזירה את הערך הראשון של מערך מכל סוג.

_____
_____
_____
_____

2. כתבו פונקציה גנרית לשימוש משלכם.

_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____

הערות

1. רשימת מילות המפתח ב-JAVA ותפקידיהן:

1.1. מילים מסוג בקרת זרימה -

break - יציאה מלולאה או switch

case - הגדרת ענף במשפט switch

continue - דילוג על האיטרציה הנוכחית של לולאה

default - הגדרת בלוק בריר ת מחדל במשפט switch

do - התחלת לולאה do-while

else - הגדרת בלוק קוד שיופעל אם התנאי לא מתקיים

for - התחלת לולאת for

if - התחלת משפט תנאי

return - יציאה מפונקציה והקצאת ערך

switch - התחלת משפט switch

while - התחלת לולאת while

1.2. מילים מסוג טיפוסי מידע -

boolean - משתנה בוליאני אשר הוא נכון או לא נכון

byte - משתנה בגודל בייט שני בין 128 ל 127-

char - משתנה המייצג תו יחיד מטבלת ה-ASCII

double - משתנה המתאים לערכים מספריים ממשים ומדוייקים

float - משתנה מספרי בעל מספר עשרוני ברמת דיוק אחת

int - מספר שלם בגודל של 32 סיביות

long - מספר שלם בגודל כפול אז 64 סיביות

short - משתנה של מספר שלם קטן בין 32,767 ל 32,768-

void - מציין שפונקציה לא מחזירה ערך כלשהו

1.3. טבלה של מילים מסוג תכנות מונחה עצמים -

class - הכרזת מחלקה

extends - ירושה ממחלקת על

final - הגדרת ישות שלא ניתן לשנותה או לרשת אותה

implements - ממימוש של ממשק

interface - הכרזת ממשק

new - יצירת אובייקט חדש

package - הגדרת חבילה

private - משתנה נגיש רק בתוך המחלקה

protected - משתנה נגיש בתוך מחלקה, החבילה ותתי מחלקות

public - משתנה נגיש מכל מקום
this - הפנייה למופע נוכחי של המחלקה
super - הפנייה למחלקת העל של המופע הנוכחי

1.4. טבלה של מילים מסוג טיפול חריגות ושונות -
abstract - מחלקה או פונקציה שתמומש בהמשך בתת מחלקה
assert - בדיקה בזמן ריצה
catch - תפיסת חריגות הנוצרות על ידי try
finally - בלוק קוד שתמיד יופעל לאחר try, ללא קשר לחריגות
throw - יצירת חריגה
throws - הגדרת החריגות שפונקציה יכולה לזרוק
try - התחלת בלוק חדש שייבדק לחריגות
import - ייבוא חבילות או מחלקות אחרות
instanceof - בדיקה אם אובייקט הוא מופע של מחלקה מסוימת
native - הגדרת מתודה במימוש קוד מקורי באמצעות JNI
static - הגדרת משתנה או מתודה ששייכים למחלקה ולא למופע
strictfp - הגבלת חישובי נקודה צפה להבטחת ניידות
synchronized - הגדרת מתודה או בלוק קוד מסונכרנים
transient - מניעת סריאליזציה של שדות
volatile - הגדרת משתנה שערכו ישתנה על ידי תהליכים שונים

2. שפת C++ ושפת C:

לשפת C++ קוראים כך כי היא נועדה להיות שדרוג לשפת C. סימן ה-++ מסמל פעולה של הוספה (increment) בשפות תכנות רבות (גם בשלנו - Java), ומשמעותו בשמה של C++ היא שהיא "צעד אחד קדימה" לעומת שפת C. כלומר, מדובר בשפה המציעה יכולות נוספות ושיפורים על פני שפת C.

3. כתיבת קוד איכותי ונקי

3.1. שמירה על קריאות הקוד

- **שמות משתנים ומשתנים**: השתמש בשמות שמתארים את התפקיד של המשתנה או המשתנה בקוד. לדוגמה, במקום לקרוא למשתנה x, תן לו שם כמו `totalSum` או `userAge`.
- **שמות פונקציות ומתודות**: השתמש בשמות שמתארים את הפעולה שהפונקציה או המתודה מבצעת. לדוגמה, במקום `func1`, תן שם כמו `calculateTotal` או `getUserDetails`.

- **פורמט והזחה**: שמור על פורמט קבוע והזחה נכונה של הקוד כדי להקל על קריאתו. השתמש ברווחים או בטאבים כדי להגדיר את ההיררכיה בקוד.
- **הפרדת קוד**: חלק את הקוד לקטעים קצרים וברורים. השתמש בשורות ריקות כדי להפריד בין בלוקים לוגיים שונים.

3.2. טיפול בשגיאות

- **בלוקים של try-catch**: השתמש בבלוקים של `try-catch` כדי לטפל בחריגות (exceptions) בצורה מסודרת. ודא שאתה מטפל בכל החריגות האפשריות ומציג הודעות שגיאה ברורות למשתמש.
- **בלוק finally**: אם יש צורך לבצע פעולה מסוימת בכל מקרה (כמו שחרור משאבים), השתמש בבלוק `finally` כדי להבטיח שהפעולה תבוצע גם אם חריגה נתפסה.

3.3. שימוש בפונקציות

- **פונקציות קטנות ושימושיות**: חלק את הקוד לפונקציות קטנות שכל אחת מהן מבצעת משימה אחת בלבד. זה יעזור להבהיר את הקוד ולהפוך אותו לקריא וקל לתחזוקה.
- **שימוש בפרמטרים**: השתמש בפרמטרים כדי להעביר מידע לפונקציות במקום להשתמש במשתנים גלובליים. זה יהפוך את הפונקציות לגמישות יותר וקלות יותר לבדיקה.

3.4. תיעוד והערות

- **הסבר חלקים מורכבים**: הוסף הערות (comments) להסברת חלקים מורכבים בקוד. הסבר את הלוגיקה והסיבות לבחירות תכנוניות מסוימות.
- **תיעוד פונקציות**: הוסף תיעוד לפונקציות באמצעות הערות מתאימות שמסבירות את התפקיד שלהן, הפרמטרים שהן מקבלות והערך שהן מחזירות.
- **פורמט הערות**: שמור על פורמט עקבי להערות כדי להקל על קריאתן.

4. טבלת ASCII

טבלת ASCII (American Standard Code for Information Interchange) היא תקן לקידוד תווים המשמש במחשבים ובמערכות תקשורת. כל תו בטבלה מקבל ערך מספרי בין 0 ל-127, שמאפשר להעביר מידע טקסטואלי בין מערכות שונות. הטבלה כוללת תווים כגון אותיות, ספרות, סימני פיסוק ותווים בקרה שונים.

(טבלה בעמוד הבא)

DEC	OCT	HEX	BINARY	SYMBOL	DEC	OCT	HEX	BINARY	SYMBOL	DEC	OCT	HEX	BINARY	SYMBOL	DEC	OCT	HEX	BINARY	SYMBOL
0	0000	00	00000000	NULL	32	0040	20	00100000		64	0100	40	01000000	@	96	0140	60	01100000	`
1	0001	01	00000001	SOH	33	0041	21	00100001	!	65	0101	41	01000001	A	97	0141	61	01100001	a
2	0002	02	00000010	STX	34	0042	22	00100010	"	66	0102	42	01000010	B	98	0142	62	01100010	b
3	0003	03	00000011	ETX	35	0043	23	00100011	#	67	0103	43	01000011	C	99	0143	63	01100011	c
4	0004	04	00000100	EOT	36	0044	24	00100100	$	68	0104	44	01000100	D	100	0144	64	01100100	d
5	0005	05	00000101	ENQ	37	0045	25	00100101	%	69	0105	45	01000101	E	101	0145	65	01100101	e
6	0006	06	00000110	ACK	38	0046	26	00100110	&	70	0106	46	01000110	F	102	0146	66	01100110	f
7	0007	07	00000111	BEL	39	0047	27	00100111	'	71	0107	47	01000111	G	103	0147	67	01100111	g
8	0010	08	00001000	BS	40	0050	28	00101000	(	72	0110	48	01001000	H	104	0150	68	01101000	h
9	0011	09	00001001	HT	41	0051	29	00101001	)	73	0111	49	01001001	I	105	0151	69	01101001	i
10	0012	0A	00001010	LF	42	0052	2A	00101010	*	74	0112	4A	01001010	J	106	0152	6A	01101010	j
11	0013	0B	00001011	VT	43	0053	2B	00101011	+	75	0113	4B	01001011	K	107	0153	6B	01101011	k
12	0014	0C	00001100	FF	44	0054	2C	00101100	,	76	0114	4C	01001100	L	108	0154	6C	01101100	l
13	0015	0D	00001101	CR	45	0055	2D	00101101	-	77	0115	4D	01001101	M	109	0155	6D	01101101	m
14	0016	0E	00001110	SO	46	0056	2E	00101110	.	78	0116	4E	01001110	N	110	0156	6E	01101110	n
15	0017	0F	00001111	SI	47	0057	2F	00101111	/	79	0117	4F	01001111	O	111	0157	6F	01101111	o
16	0020	10	00010000	DLE	48	0060	30	00110000	0	80	0120	50	01010000	P	112	0160	70	01110000	p
17	0021	11	00010001	DC1	49	0061	31	00110001	1	81	0121	51	01010001	Q	113	0161	71	01110001	q
18	0022	12	00010010	DC2	50	0062	32	00110010	2	82	0122	52	01010010	R	114	0162	72	01110010	r
19	0023	13	00010011	DC3	51	0063	33	00110011	3	83	0123	53	01010011	S	115	0163	73	01110011	s
20	0024	14	00010100	DC4	52	0064	34	00110100	4	84	0124	54	01010100	T	116	0164	74	01110100	t
21	0025	15	00010101	NAK	53	0065	35	00110101	5	85	0125	55	01010101	U	117	0165	75	01110101	u
22	0026	16	00010110	SYN	54	0066	36	00110110	6	86	0126	56	01010110	V	118	0166	76	01110110	v
23	0027	17	00010111	ETB	55	0067	37	00110111	7	87	0127	57	01010111	W	119	0167	77	01110111	w
24	0030	18	00011000	CAN	56	0070	38	00111000	8	88	0130	58	01011000	X	120	0170	78	01111000	x
25	0031	19	00011001	EM	57	0071	39	00111001	9	89	0131	59	01011001	Y	121	0171	79	01111001	y
26	0032	1A	00011010	SUB	58	0072	3A	00111010	:	90	0132	5A	01011010	Z	122	0172	7A	01111010	z
27	0033	1B	00011011	ESC	59	0073	3B	00111011	;	91	0133	5B	01011011	[	123	0173	7B	01111011	{
28	0034	1C	00011100	FS	60	0074	3C	00111100	<	92	0134	5C	01011100	\	124	0174	7C	01111100	|
29	0035	1D	00011101	GS	61	0075	3D	00111101	=	93	0135	5D	01011101	]	125	0175	7D	01111101	}
30	0036	1E	00011110	RS	62	0076	3E	00111110	>	94	0136	5E	01011110	^	126	0176	7E	01111110	~
31	0037	1F	00011111	US	63	0077	3F	00111111	?	95	0137	5F	01011111	_	127	0177	7F	01111111	DEL

5. ספירת base 10

בספירת בסיס 10, או decimal, משתמשים בעשר ספרות (0 עד 9) על מנת לייצג כל מספר. הבסיס הזה הוא הנפוץ ביותר ונקרא גם המערכת העשרונית.

בסיס 2 (בינארי): משתמש רק ב-2 ספרות (0 ו-1) וכל מיקום מייצג חזקה של 2.

בסיס 16 (הקסדצימלי): משתמש ב-16 ספרות (0-9 ו-A-F) וכל מיקום מייצג חזקה של 16.

בסיס 60 (שעתי): משתמש ב60 בשביל לעבר בין דקות לשעות.

6. כתיבת קוד יעיל (DRY)

DRY או **D**on't **R**epeat **Y**ourself, האו עיקרון בהנדסת תוכנה שעקרונו הוא לכתוב קוד מודולרי המאפשר כתיבת קוד יעיל כשהוא משומש במלואו, לדוגמה כתיבת פונקציות שונות שתכליתן היחיד הוא השימוש בטיפוסי מידע שונים, הא קוד לא יעיל שלא עומד בסטנדרט שלנו.

פרוייקט סיום

פרוייקט סיום - לאחר סיום לימוד החומר הנ"ל, תוכלו לבנות מערכת גדולה בעזרת הרבה כלים שרכשת במהלך הקורס, הפרויקט שעליכם לבנות תלוי בכם, לפניכם 3 אופציות לבחירת פרויקט הסיום.

- **מערכת ניהול ספריות (ציון מקסימלי 120 נקודות)**
- **מערת ניהול של מכונת שתייה (ציון מקסימלי 100 נקודות)**
- **מערכת בקרת מעליות (ציון מקסימלי 110 נקודות)**

דרישות הפרויקט -

- שימוש לפחות **250** שורות קוד שנוצלו בכל המחלקות
- שימוש בתכנות מונחה עצמים
- שימוש של **2** מחלקות לא כולל Main (לדוג, ספר או מנוי)
- שימוש בספריות Random, Math & Scanner
- שימוש ב Switch & Case **אחד** לפחות
- שימוש ב 6 פונקציות
- שימוש בתנאים מתקדמים

קבלת ניקוד מקסימלי כציון סופי -

- שימוש במערך דו מימדי
- שימוש בלולאה מקוננת
- פולימורפיזם
- מחלקה גנרית

הערות

1. **המונח "שימוש" מתייחס לשילוב רלוונטי מטרת הפרויקט הנבחר.**
2. **ניתן לפנות למנחה לצורך קבלת סיוע והגשת בקשות חריגות.**
3. **הערכת הציונים תתבצע על ידי המנחה ועל פי שיקול דעתו, וכן תיעשה בחינה על ידי התלמידים בעת הגשת הפרויקט.**
4. **פרויקטים המצריכים קלט מהמשתמש, הקפידו על בדיקת תקינות הקלט.**
5. **השתמשו הערות בקוד להסביר חלקים מורכבים.**

חל איסור מוחלט להשתמש בכלים אוטומטיים, תוכנות צד שלישי או בבינה מלאכותית לצורך כתיבת הקוד או ההערות. במקרה של חשד לשימוש בכלים אלו, הפרויקט ייפסל!