שיעור שני – מבוא לשפת ג'אווה

<u>חזרה על שיעור קודם:</u>

בכל פעם שמצהירים על משתנה- תופסים מקום במרחב הזיכרון בין 0 ל-9.

מערך דו ממדי- ליצירת מערך דו ממדי: int [] [] arrX5= new int[5][5]; זה יוצר מערך כזה-

		100		

גישה לכל תא במערך כך: arr[2][2]=100;

<u>CLASS = מחלקה</u>

השיטה main

כל תוכנית ג'אווה כוללת את השיטה הראשית main(....) שממנה מתחיל ביצוע התוכנית.

public static void main(String[] args)

ג'אווה היא שפת תכנות מונחה עצמים- כלומר, העולם נחלק לעצמים וניתן ייצוג לכל עצם.
מחלקה בג'אווה מהווה את התבנית ליצירת אובייקטים בעלי אותן תכונות. האובייקטים מהווים חלק מהמודל שאנו יוצרים, מהמציאות של "עולם הבעיה". כאשר כותבים תוכנית בשפה מונחת עצמים, אין אנו יוצרים אובייקט מסוים, אלא, יוצרים **מחלקה** של אובייקטים.
כאשר כותבים תוכנית בג'אווה מעצבים ויוצרים קבוצה של מחלקות. כאשר התוכנית בג'אווה "רצה" נוצרים אובייקטים של המחלקות בהתאם למחלקות שהגדרת, והאובייקטים מזמנים שיטות (פונקציות)המוגדרות על האובייקטים הללו, זאת כדי, לבצע את המשימה שנועדה לתוכנית.

יצירת פרויקט חדש---- פתיחת new package ב-src ---- פתיחת new class **וסימון** האופציה:
```
public static void main(String[] args)
```
פתיחת new class נוסף, הפעם ללא סימון האופציה הנ"ל.
ופתיחת קלאסים נוספים על פי הצורך.

יצירת קלאס, מבנה:

המילה class , שם המחלקה ושני סוגריים "מסולסלים" לתיחום המחלקה.
המחלקה מגדירה את תכונות המחלקה (אופציונלי: פונקציות, קלאסים פנימיים)

```
class Book {
    String name;
    int numPages;
    double price;
      }
```

קלאסים (מחלקות) מייצרים טיפוסים חדשים, טיפוסים שאנחנו רוצים. בהזמנה...

כל אחד מהטיפוסים תופסים קטע במרחב הזיכרון, גודל קבוע שמובנה בסוג הטיפוס.
בתוך הקלאס מכניסים משתנים כך: Car abc= new Car();
ואז מזינים את הערכים, כך: abc.company="Honda";.

בניגוד להגדרת משתנים בסיסיים (int, boolean וכיו"ב) המקצה זיכרון למשתנים ומאפשרת גישה אליהן, מאותו הרגע, ההכרזה על אובייקט איננה מקצה זיכרון אלא רק מגדירה למפרש שם שיכיל בהמשך את הפנייה לעצם מסוג מסויים ללא יצירת העצם. כדי לאפשר גישה לעצם (חברים, שיטות וכו') עלינו להקצות זיכרון לאותו האובייקט ולעדכן את ההפניה לעצם שנוצר והמקום שקיבל בזכרון – פעולה המתבצעת ע"י האופרטור **new**. המופעים נכתבים בקובץ הראשי והם למעשה ההשמה של האובייקטים במחלקה.

מה מבצע האופרטור new?

כאשר משתמשים באופרטור new מתרחשים מספר דברים, מופע חדש של המחלקה שמופיעה במשפט נוצר, זיכרון מוקצה למופע של המחלקה לפי גודל הזיכרון הנדרש, והשם שנתנו לאובייקט מכיל את ההפניה לאובייקט.

יצירה של מופע מחלקה :

```
chair stool = new chair();
```

הסוגריים מציינים שמדובר בשיטה(השיטה הבונה).

הקצאת ערכים לאובייקטים.

```
stool.legs = 1;
stool.color ='g';
```

מספר הרגליים וצבע הם תכונות המחלקה ונקראים גם member כלומר נתונים חברים במחלקה. הקישור ביו אובייקט לבין נתונים, החברים במחלקה של האובייקט, הוא באמצעות **סימון נקודה** או **"חיבור באמצעות נקודה"** או **"יחוס נקודה"**

<u>תרגול כיתה:</u>

צור את הקלאסים הבאים:

1. ירק: סוג, משקל, מחיר.
2. בניין: קומות, מעלית כן/לא, כתובת.
3. מטוס: סוג, מרחק טיסה, כמות מושבים.
4. מסך: מידה, רזולוציה, מהירות.

ייצור קלאס כך: יצרת קלאס, הצהרה על משתנים, גישה אליהם והזנת ערכים ב-main.

שיטות במחלקה – Methods

הקלאס מייצג שבלונה מסוימת, וכל אובייקט מייצג משתנה אחר שלו.

בראשי (main) בניית אובייקט של הקלאס (פירות---- בננה= פירות=קלאס, בננה= אובייקט)
הגדרת משתנה בראשי והשמה בתוכו את הפונקציה עם קידומת של שם האובייקט, משתני האובייקט הם אלו שנשלחים כפרמטרים אל הפונקציה והתוצאה חוזרת אל המשתנה שהגדרנו.
הפונקציה עצמה מוצהרת בדף הקלאס.
קריאה לפונקציה בראשי: טיפוס שם משתנה= אובייקט.שם פונקציה.
כלל: בתוך הקלאס נרשום את כל מה שקשור לקלאס! לא לערבב עם הראשי! כלומר, בקלאס נגדיר את המשתנים וגם נכתוב בו את הפונקציות.

- הנקודה הזאת מאוד חשובה היום כשכותבים אפליקציות והן תומכות בגרסאות ישנות כי הקלאסים הישנים נשארים ורק מוסיפים עליהם עדכונים.

שיטות הן חלק מהמחלקה, שורות קוד המגדירות פעולות שניתן לבצע על אובייקטים או שיטות מחלקה, כדי לבצע משימות דרושות.
המחלקה או אובייקט של מחלקה, יכולים לזמן שיטה השייכת למחלקה או למחלקה אחרת כדי לבצע משימה לדוגמא:

- להעביר מידע על שינוי לאובייקט אחר
- לבקש מאובייקט אחר לבצע משימה

<u>ישנם 2 סוגי שיטות</u>

Instance methods - שיטות מופע

שיטות הפועלות על האובייקטים שיצרנו ונקראים בקיצור methods

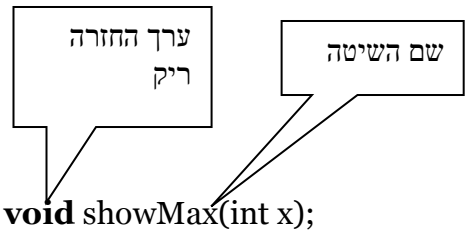
חתימת שיטה המחזירה ערך

public double sum(int num);

שיטות המחזירות ערך מבצעות זאת באמצעות המשפט return, השיטה יכולה לקבל פרמטרים או לא לקבל פרמטר כלל ויופיעו רק סוגריים (חייבים לכתוב את הסוגריים גם אם אין פרמטרים). כותרת השיטה נקראת "חתימת השיטה".

שיטה שאינה מחזירה ערך

השיטה מבצעת חישובים ופעולות שונות אולם, אינה מחזירה ערך לשיטה שזימנה אותה. שיטה זו מקבילה למה שמתכנתים בשפות פרוצדורליות קוראים "פרוצדורה" או "שגרה"



public void showMax(int x);

Class methods – שיטות מחלקה

שיטות מחלקה הן פעולות שהמחלקה עצמה מבצעת ולא אובייקט של המחלקה (מופע). שיטות אלו ניתן לזמן גם בלי לבנות מופעים של המחלקה.

<u>עבודת כיתה</u>- צור קלאס ריבוע, בתוכו ערכים: אורך, רוחב, עומק.

- צור את הפונקציות הבאות:
  - פונקציה להחזרת שטח הפנים,
  - פונקציה להחזרת נפח,
  - פונקציה להדפסת הריבוע ב- "*" לפי ערך אורך ורוחב.

הגדרת פונקציה, תחביר: public int funName(){return *nosch mevocesht*}

privat

הרשאה, אם ניתן את זה כקידומת לשם של משתנה הוא ימנע את הגישה אל אותו משתנה, אך לא מונע את הגישה לפונקציות שמשתמשות באותו משתנה.
במצבים שנרצה לעשות ולידאציות לקלאס שלנו, שמירה על חלק מהמשתנים שלא יהיו נגשים.
כל המשתנים בפרייבט תמיד יכולים לקבל:
get- לקבל תוצאה
Set- לשלוח משתנה.שם משתנה נקודה שם הפונקציה של גט או סט.

לעמוד בקובץ ה class המבוקש------ source------ generate getters and setters---- לסמן את התכונות הרצויות.

עבור כל משתנה הוא מוסיף 2 פונקציות גטר וסטר. לא מאפשר פנייה ישירה לאותו משתנה!
חייבים להית בסטטוס פרייבט כדי שתהיה גישה אליהם לפונקציות גטר וסטר ליצור להם משתנים ופונקציונליות.
להסבר מפורט וברור יותר בנושא זה: לחץ כאן

שיעורי בית- להמציא 2 קלאסים, בכל תחום.

- לכל אחד מהקלאסים שיצרנו בשיעור להגדיר privet וליצור להם גטר וסטר.

נספח- סיכומים וקיצורים

שם משתנה בג'אווה - כללים

- התו הראשון בשם של משתנה חייב להיות אות "קטנה" (lowercase)
- אם שם המשתנה מורכב מכמה מילים, כל מילה חדשה תתחיל באות גדולה (capital letter)
- כל שאר האותיות בשם המשתנה הן אותיות קטנות או גדולות, וניתן גם להשתמש בספרות ו- הסימנים _ $
- שם מחלקה מתחיל באות גדולה, רצוי שכל חלק מהשם המהווה מילה בעלת משמעות יתחיל באות גדולה.

חשוב

ג'אווה "רגישה" לאות גדולה או אות קטנה, המשתנה num שונה מהמשתנה Num . שני המשתנים הללו בתוכנית אחת יהיו שונים.

כל הכרזה של משתנה חייבת להגדיר את סוג המשתנה, סוג המשתנה. יהיה אחד מהסוגים הבאים:

- אחד מסוגי המשתנים הבסיסים (משתנה מספרי או לוגי או תו).
- שם של מחלקה או ממשק (לדוגמא class Point) .
- מערך.

ישנם משתנים עם ייצוג שונה המגדיר את אופן האחסון של סוג המשתנה ואת גודל הזיכרון המוקצה

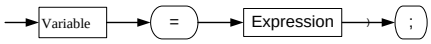
סיכום כללים בכתיבת תוכנית בג'אווה

ג'אווה כמו שפות תכנות אחרות מחייבת כללי כתיבה קשיחים, המהדר של השפה לא מסיים בהצלחה את תהליך התרגום של התוכנית, אם לא מקפידים על תחביר הפקודות, כמו כן בג'אווה ישנם כללי כתיבה, שעבור חלק מהם אי שמירה על הכללים אינה פוגעת בתהליך הידור התוכנית, אולם, הכללים אומצו ע"י המתכנתים בכל העולם ואימוצם מקל על הבנת התוכנית.

1. נקודה-פסיק בסוף כל פקודה. –חובה
2. הקפדה על אינדנטציה – ההזחה של בלוקים באופן שהמעיין יודע לאיזה קטע או בלוק שייכת כל הוראה - מומלץ.
3. שמות ברורים ובעלי משמעות. - מומלץ
4. שם מחלקה - מתחיל באות כל מילה חדשה בשם באות גדולה – מומלץ
5. שם משתנה - מתחיל באות קטנה וכל מילה נוספת בשם מתחילה באות גדולה – מומלץ
6. מילים שמורות נכתבות באותיות קטנות - חובה.

משפט השמה בג'אווה

מבנה תחבירי של משפט השמה



במשפט השמה יש שני צדדים. בצד הימני של סימן השוויון יש ביטוי בג'אווה, בצד השמאלי יש משתנה שהוכרז. ביצוע משפט ההשמה "אומר" למחשב לחשב את ערך הביטוי המופיע בצד ימין ואת תוצאת החישוב להכניס למשתנה המופיע בצד שמאל.

טבלת אופרטורים

אופרטור	הסבר	דוגמאות
[]	משמש להגדרת מערך	int [] mark; int [] data ={1,2,3,4,5};
.	נקודה מהווה אופרטור מרכזי בג'אווה ומאפשרת גישה לתכונות ושיטות של אובייקטים	card.num=8; num.bigNumber(x,,y);
(סוג)	תרגום – שינוי טיפוסי ערכים - casting	(int)(Math.sin(num))
new	אופרטור המשמש ליצירה של אובייקטים ומערכים	chair stool = new chair();