חזרה – מחלקות ועצמים

A

B

C

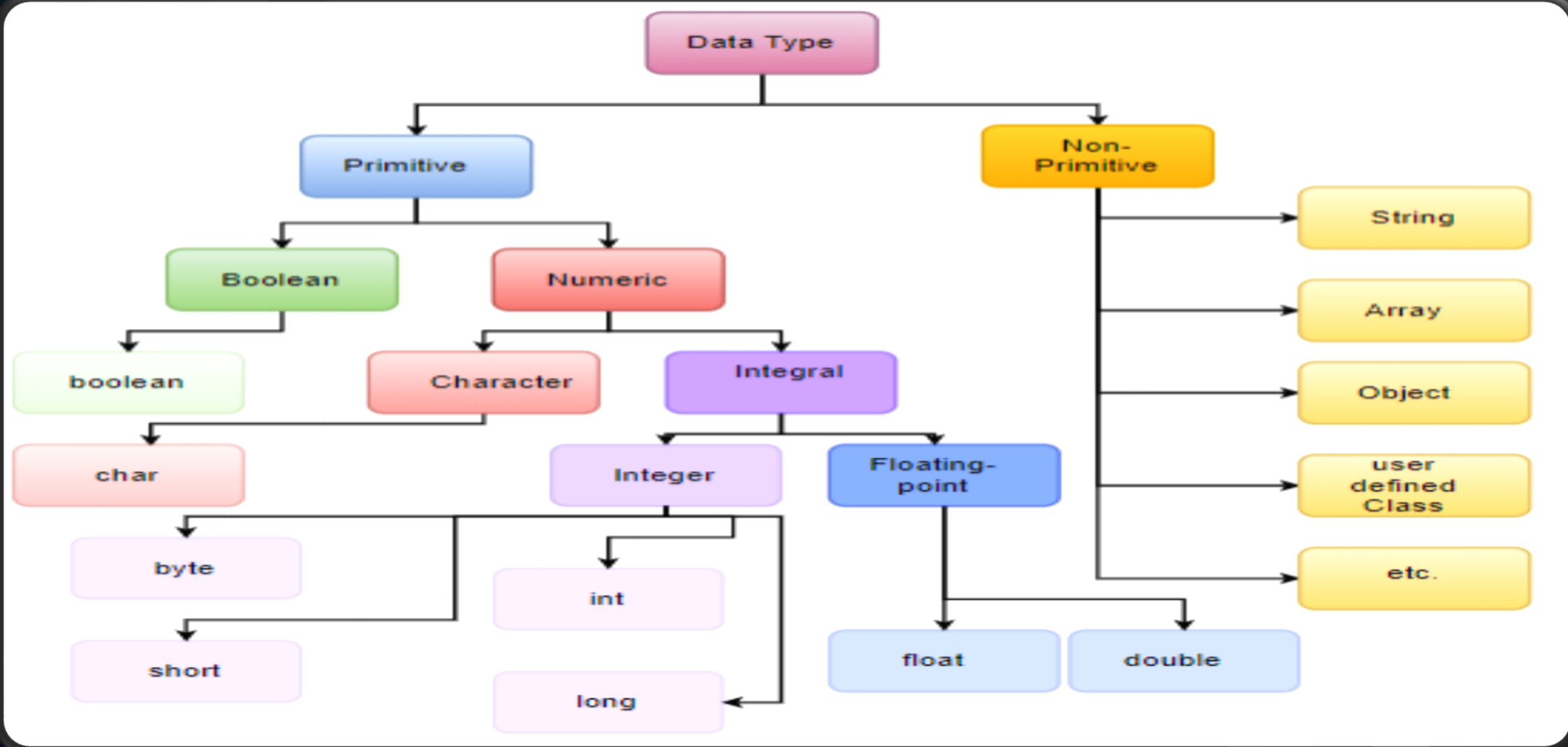
דפנה לוי רשתי

מהו משתנה? משתנה variable הוא שם סימבולי המשוייך למקום בזיכרון אשר יכול להכיל ערכים שונים במהלך ריצת התוכנית.

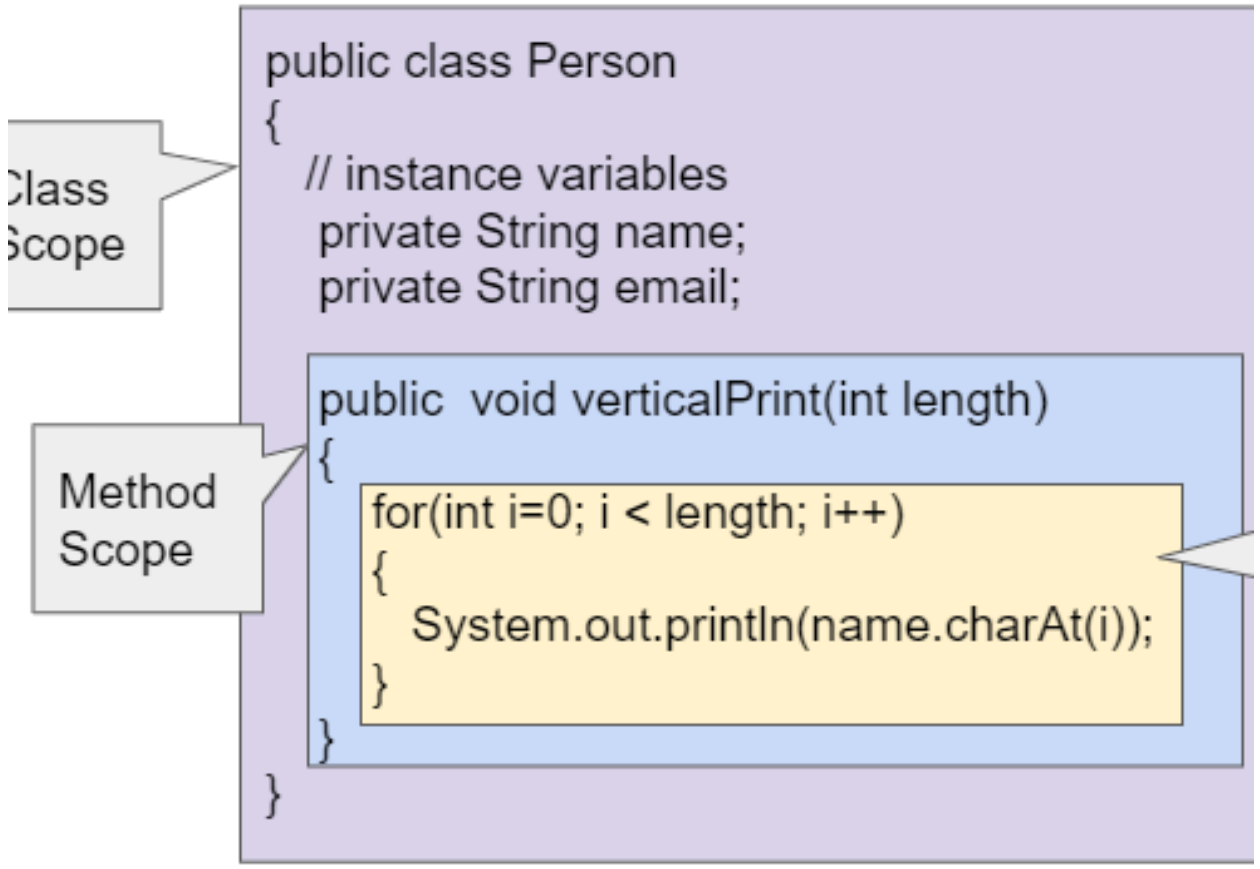
למה צריך משתנים? ניתן להשתמש במשתנים על מנת לשמור ערכים, לשימוש בשלב מאוחר יותר בתוכנית.

ב Java לכל משתנה יש טיפוס type. הטיפוס קובע אילו סוגי ערכים המשתנה יכול להכיל בזמן ריצת התוכנית, ומהן הפעולות שניתן להפעיל עליו. למעשה, הטיפוס מגדיר כמה מקום בזיכרון יהיה בשימוש עבור הערכים של המשתנה ואיך נשתמש במקום זה על מנת לייצג את הערכים השונים שיכול לקבל המשתנה.

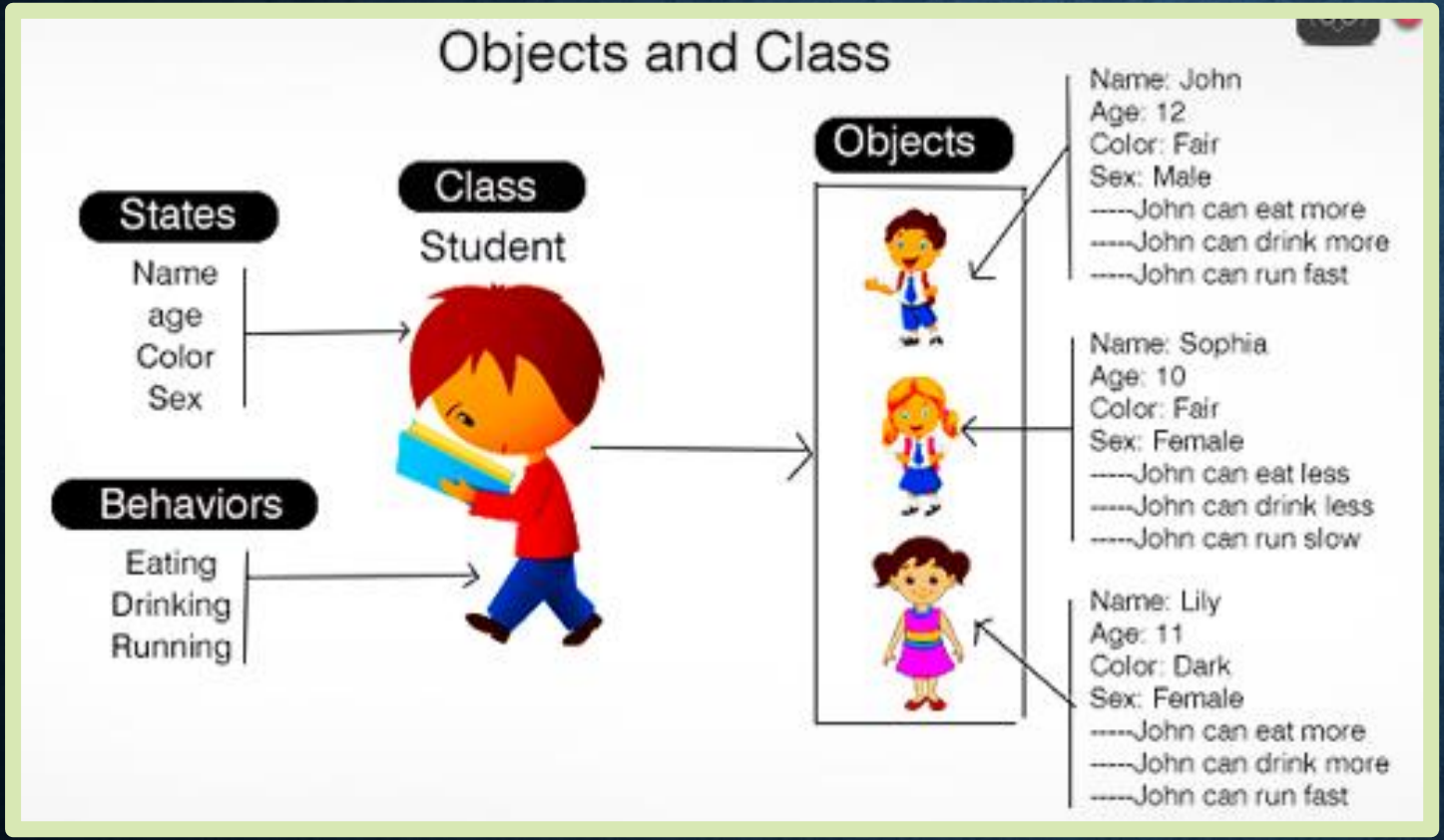
ל Java יש שני סוגים של טיפוסים: primitives ו objects.


Data Type
Primitive
Non-Primitive
Boolean
Numeric
boolean
Character
Integral
char
Integer
Floating-point
byte
int
short
long
float
double
String
Array
Object
user defined Class
etc.

טווח ההכרה

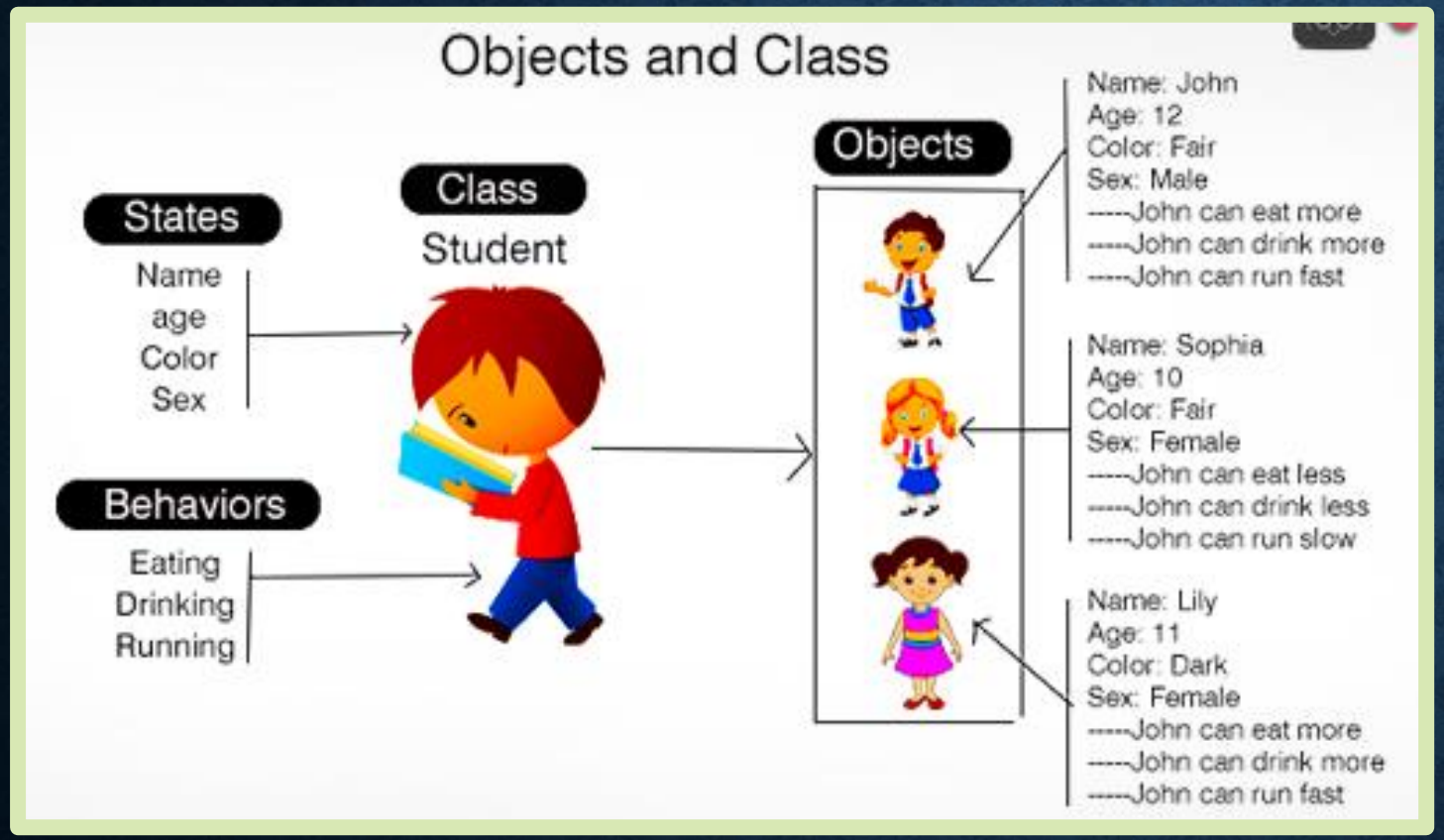


- טווח ההכרה של משתנה scope הוא האזור בתוכנית שבו המשתנה מוגדר וניתן להשתמש בו.
- טווח ההכרה של משתנה תלוי בבלוק בו הוא מוגדר (בלוק מצוין ע"י סוגריים מסולסלים).
- טווח ההכרה של המשתנה מתחיל בשורה שבו המשתנה מוגדר ומסתיים בסוף הבלוק שבו נמצאת הגדרת המשתנה.



משתנים המוגדרים בעזרת טיפוסים פרימיטיביים בלבד אינם מספיקים לכתיבה נוחה של הקוד. למשל, אם נממש מערכת המנהלת רשומות סטודנטים, נשאף כי כל נתוני הסטודנט כגון שם, מספר ת"ז, כתובת, טלפון, וציונים בקורסים יהיו מאוגדים בתוך ישות אחידה טיפוס חדש הנקרא Student

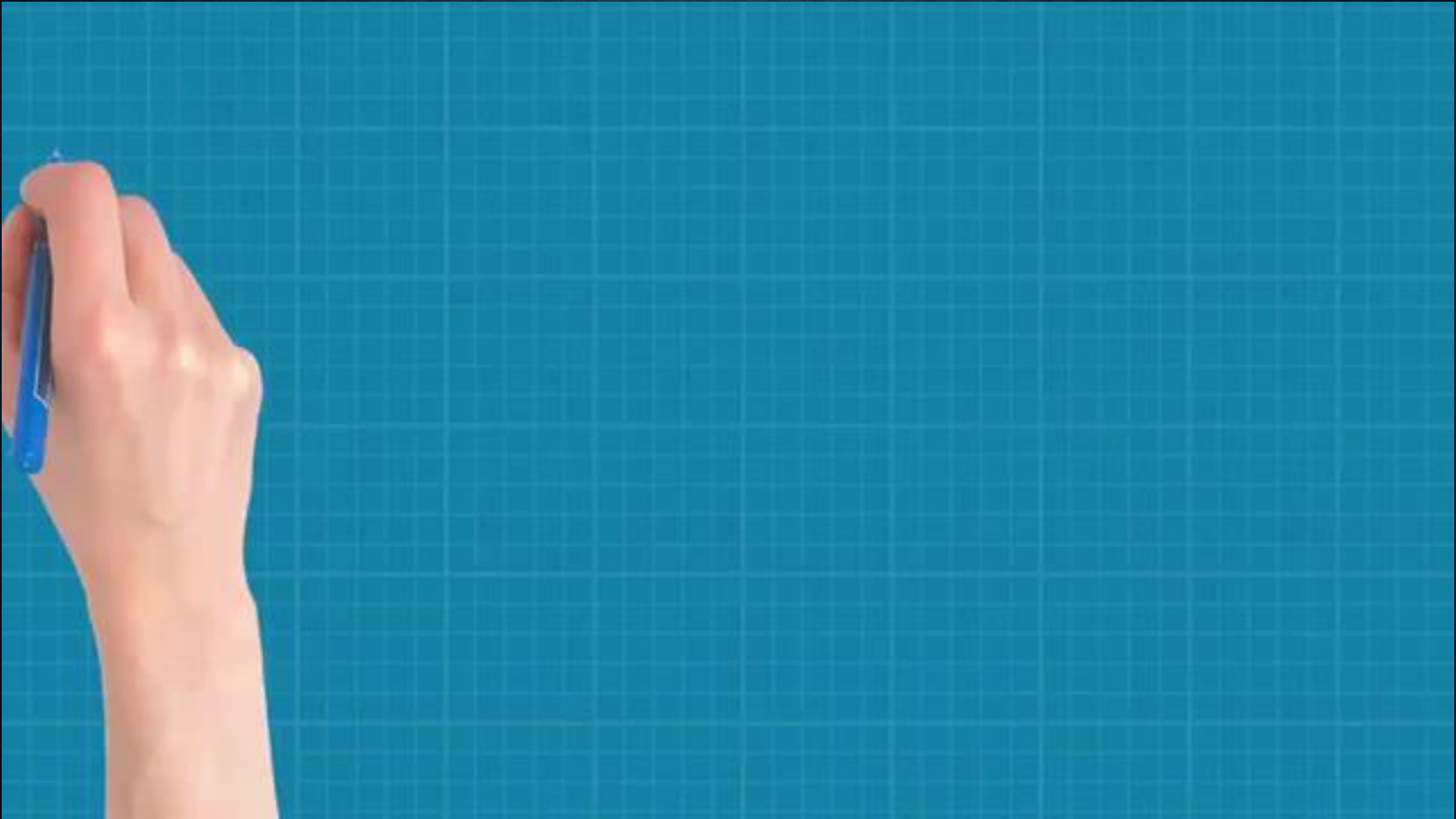
טיפוסים אלה יעזרו לנו לנהל ביתר קלות כמות גדולה של קוד. למשל ,במקום לשלוח לפונקציה מסוימת. רשימה ארוכה של פרמטרים המתארים סטודנט יחיד, נוכל עתה לשלוח פרמטר יחיד המאגד בתוכו מספר ערכים. כמו כן, נוכל ליצר מערך המכיל סטודנטים, למיין קבוצה של סטודנטים לפי ת"ז או לפי ציון ממוצע וכו .'



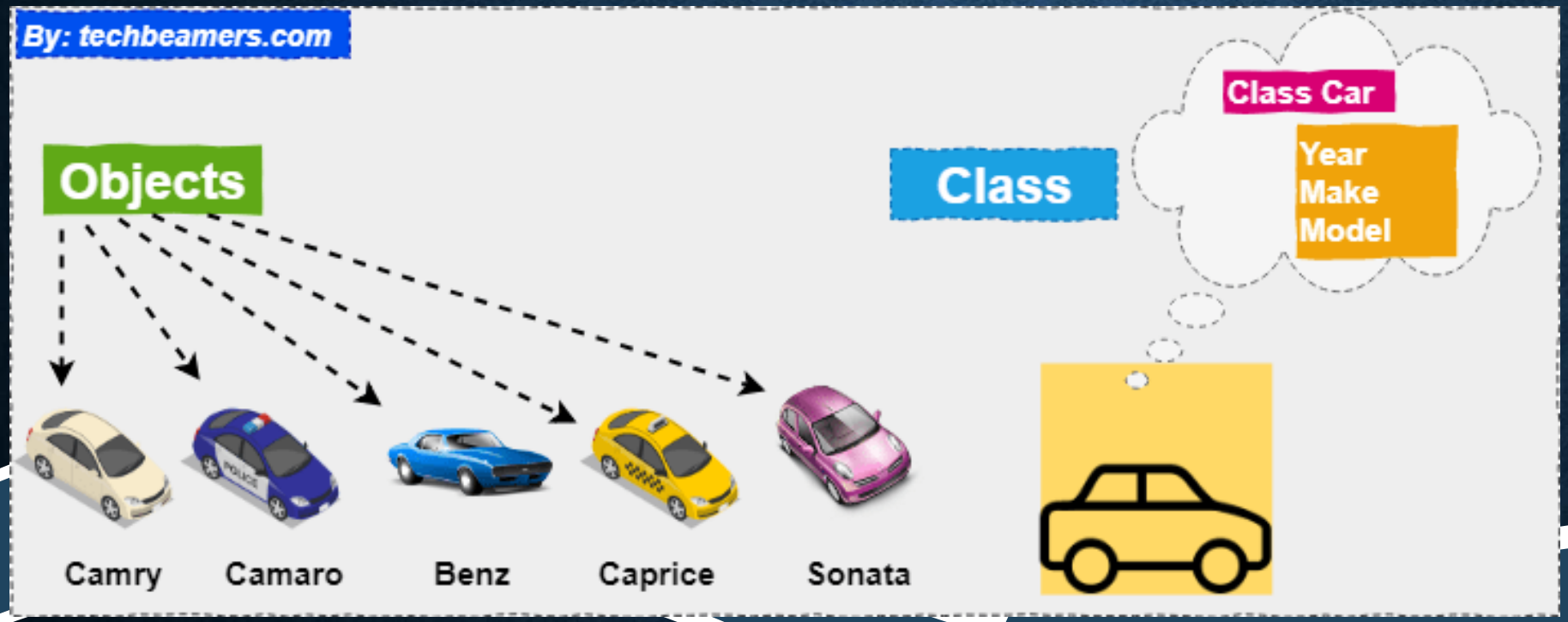
לטיפוסים חדשים אלו קוראים מחלקות classes, ולמשתנים הנוצרים מטיפוסים אלו קוראים עצמים/אובייקטים objects או מופעים instances

מחלקה מגדירה את אוסף המשתנים הפנימיים אשר כל מופע שלה יכיל הנקראים תכונות (שדות fields)

בנוסף, המחלקה גם מגדירה אוסף של פעולות אשר המופעים שלה יכולים להפעיל (שיטות/מתודות)



ABSTRUCTION

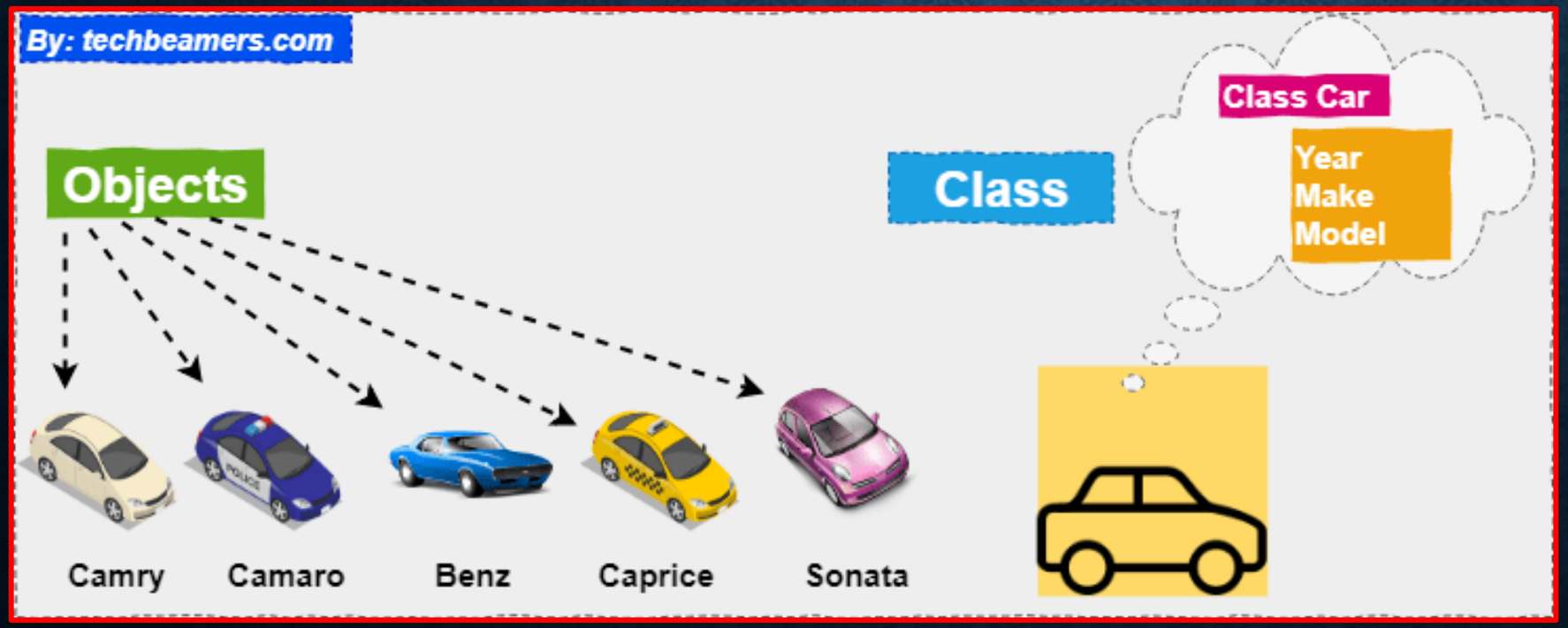


1 הפשטה

עקרונות תכנות מונחה עצמים

ABSTRUCTION

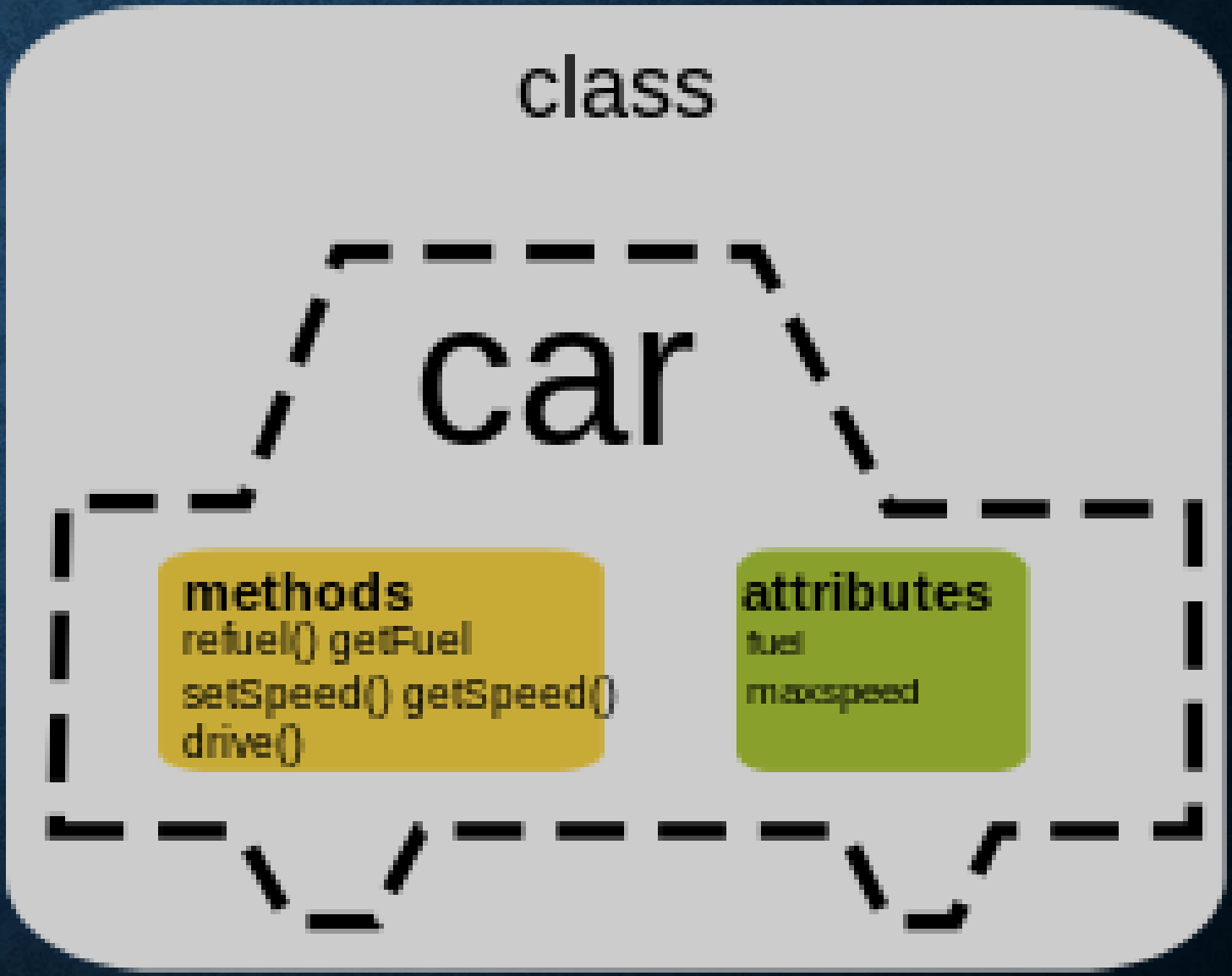
שפות מונחות עצמים מתבססות על מחלקות, שהן ישות לוגית אבסטרקטית, המהווה כמעין תבנית אשר ממנה נוצרים האובייקטים



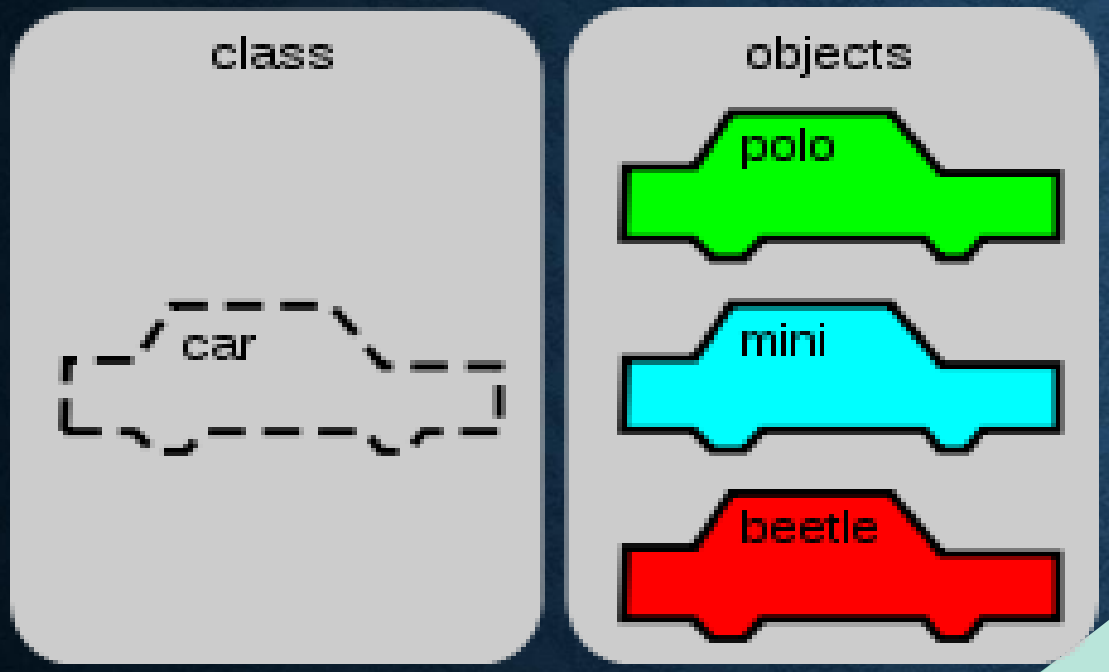
מחלקות (classes)

Attributes - משתנים באמצעותם נתאר אובייקט מהמחלקה

Methods - פונקציות אשר מתארות את ההתנהגויות (הפונקציונאליות) של האובייקט.



עצמים (objects)



אובייקט (object) הוא מופע (instance) של מחלקה

אובייקט הינו ישות ממשית התופסת משאבים, בעוד המחלקה היא ישות לוגית הגדרתית .

הפעולה הבונה

בכל מחלקה המייצרת עצמים, תהיה לפחות פעולה בונה אחת.
אם לא הוגדרה אף פעולה בונה, ניתן להשתמש בפעולה בונה ברירת מחדל שאינה מקבלת פרמטרים

שם הפעולה הבונה הוא שם המחלקה, ואין בחתימה שלה ערך מוחזר.

הפעולה הבונה מופעלת על ידי new וגורמת להקצאת שטחי זיכרון, אתחול לברירת המחדל (0, null), מבצעת את הוראות הפעולה.

הפעולה הבונה מחזירה את ההפניה למקום בו העצם נמצא בזיכרון

```java
public class PencilCup {
    private int numPencils;
    private int numPens;
    private boolean scissors;

    public PencilCup() {
        this.numPencils = 0;
        this.numPens = 0;
        this.scissors = false;
    }
    public PencilCup(int x) {
        this(x, x, false);
    }
    public PencilCup(int numPencils, int numPens, boolean scissors) {
        this.numPencils = numPencils;
        this.numPens = numPens;
        this.scissors = scissors;
    }
    public PencilCup(PencilCup p) {
        this.numPencils = p.numPencils;
        this.numPens = p.numPens;
        this.scissors = p.scissors;
    }
```

מצב שבו יש כמה פעולות באותו שם,
הנבדלות זו מזו בשורת הפרמטרים, נקרא
העמסת פעולות overloading method

פעולה בונה מעתיקה – מקבלת עצם ויוצרת
עצם חדש עם אותן ערכי תכונות
copyConstructor

זימון פעולה בונה אחרת – באמצעות this

```java
public class Square {

    private int side;
    private String color;

    public int getSide() {return this.side;}
    public String getColor() {return this.color;}
}
```

מחלקה ללא פעולה בונה

Program.java Bucket.java Square.java

```java

public class Program {

    public static void main(String[] args) {
        Square s = new Square();
        System.out.println("side: " + s.getSide());
        System.out.println("color: " + s.getColor());
    }
}
```

יצירת עצם

Problems Javadoc Declaration Console

<terminated> Program (35) [Java Application] C:\Program Files\Java\jre1.8.0_181\bin\javaw.exe (10 8:37:0

```
side: 0
color: null
```

התכונות אותחלו לערכי ברירת המחדל - מספר שלם לאפס ומחרוזת להפנייה ריקה

הפעולה הבונה

A

הגדרת שטחי זיכרון מתאימים לעצם החדש על פי הגדרת התכונות בטיפוס שלו

B

אתחול התכונות על פי ערכי ברירת המחדל של הטיפוסים שלהם

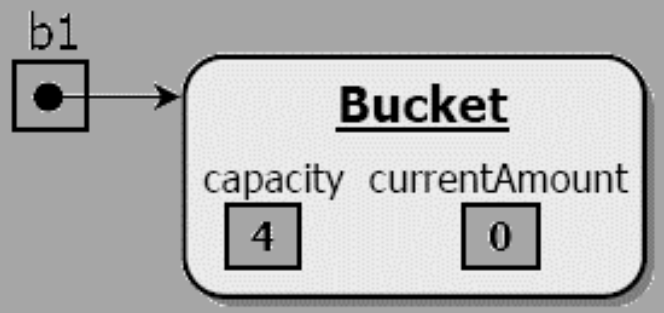
C

ביצוע גוף הפעולה הבונה

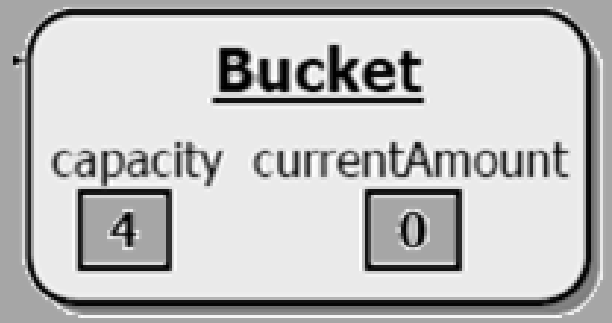
D

החזרת ההפניה הכתובת של המקום בזיכרון בו נמצאים פרטי העצם למשתנה

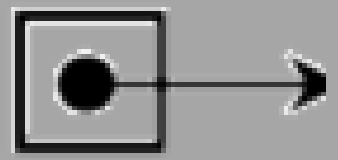
נבדיל בין שלושה מושגים



שרטוט עצם



העצם



ההפניה (כתובת)

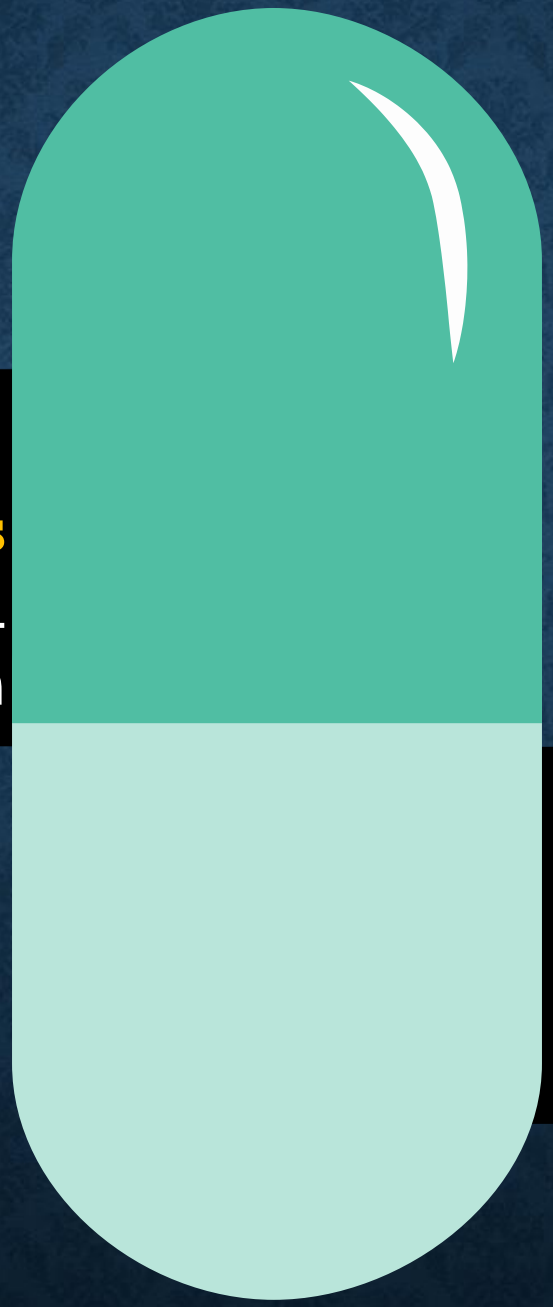
Bucket b1

המשתנה המחזיק את ההפניה

מחלקה

- תכונות
- פעולה.פעולות בונות
- פעולות מאחזרות
- פעולות קובעות.משנות ערכים
- פעולות חישוביות נוספות
- פעולת toString*
- פעולת equals*

מחלקות (classes)



Attributes - משתנים באמצעותם נתאר אובייקט מהמחלקה

Methods - פונקציות אשר מתארות את ההתנהגויות (הפונקציונאליות) של האובייקט.

חתימת פעולה

‹access level› ‹optional modifiers› ‹return value› ‹method name› (‹parameters›)

הרשאת גישה
private
protected
public

מגדירים
static
final
abstract

ערך מוחזר
כל טיפוס קיים
void

```
public double getMyFundsFromBank(String bankName)
```

signature is method name + parameters only

```java
public int getNumPencils() {
    return numPencils;
}
public void setNumPencils(int numPencils) {
    this.numPencils = numPencils;
}
public int getNumPens() {
    return numPens;
}
public void setNumPens(int numPens) {
    this.numPens = numPens;
}
public boolean hasScissors() {
    return scissors;
}
public void setScissors(boolean scissors) {
    this.scissors = scissors;
}
```

פעולות אחזור וקביעה אינן חובה. לעיתים לא נרצה לאפשר גישה או שינוי ערך התכונה.

בפעולות set אנו יכולים להגביל את השינוי לערכים מותרים בלבד (למשל רק חיוביים/שונים מ-0, לא מחרוזת ריקה או לא חוקית וכו')

```java
public int value(int price) {
    int pencilsPrice = this.numPencils * price;
    int pensPrice = this.numPens * price * 3;
    int scissorsPrice = 0;
    if (this.scissors)
        scissorsPrice = price * 10;
    return pencilsPrice + pensPrice + scissorsPrice;
}

public static int value(PencilCup p, int price) {
    int pencilsPrice = p.getNumPencils() * price;
    int pensPrice = p.getNumPens() * price * 3;
    int scissorsPrice = 0;
    if (p.hasScissors())
        scissorsPrice = price * 10;
    return pencilsPrice + pensPrice + scissorsPrice;
}
```

פעולות חישוביות נוספות לפי הפונקציונליות אותה אנו מבקשים לתת לעצמים מטיפוס המחלקה.

הפעולה מופעלת על עצם ספציפי ע"י סימן הנקודה, וערכו של המשתנה מועבר כך למשתנה המערכת this

Usage of Java this Keyword

There can be a lot of usage of java this keyword. In java, this is a reference variable that refers to the current object.

01 this can be used to refer current class instance variable.

02 this can be used to invoke current class method (implicity)

03 this() can be used to invoke current class Constructor.

04 this can be passed as an argument in the method call.

05 this can be passed as argument in the constructor call.

06 this can be used to return the current class instance from the method

```
public String toString() {
    return "PencilCup [numPencils=" + numPencils + ", numPens=" +
            numPens + ", scissors=" + scissors + "]";
}
```

הפעולה toString מחזירה מחרוזת המתארת את מצב העצם – ערכי התכונות העכשווים.

הפעולה אינה מדפיסה!

כל פעולות ההדפסה ממחלקת System מקבלות כפרמטר מחרוזת, אם הפרמטר אינו מחרוזת, אז הן מפעילות עליו את פעולת ה toString() הרלבנטית לגביו (ולכן גם כותרת הפעולה אחידה).

אם לא הוגדרה פעולת toString() ספציפית, תופעל פעולת toString() כללית, המחזירה מחרוזת הכוללת את שם המחלקה ו"כתובת" העצם - PencilCup@372f7a8d

```java
public boolean equals(PencilCup other) {

    if (numPencils != other.numPencils)
        return false;
    if (numPens != other.numPens)
        return false;
    if (scissors != other.scissors)
        return false;
    return true;
}
```

הפעולה equals קיימת כברירת מחדל בכל מחלקה. אם לא נגדיר מחדש, הפעולה תחזיר אמת אם שתי ההפניות זהות – כלומר אם מדובר באותו עצם.

```java
public boolean equals(Object obj) {
    if (this == obj)
        return true;
    if (obj == null)
        return false;
    if (getClass() != obj.getClass())
        return false;
    PencilCup other = (PencilCup) obj;
    if (numPencils != other.numPencils)
        return false;
    if (numPens != other.numPens)
        return false;
    if (scissors != other.scissors)
        return false;
    return true;
}
```

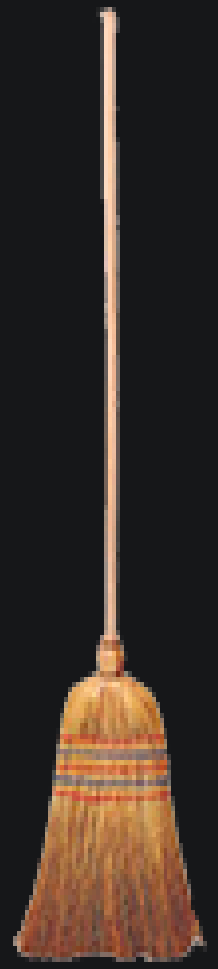
תכונות מחלקה
שיטות מחלקה

static
methods

static
variables

איבר של מחלקה

משתנה המשרת את כל עצמי המחלקה
ואינו משויך לעצם כלשהו קרוי
תכונה/איבר של מחלקה



המטאטא של כל ילדי הכיתה

איבר של מחלקה

```
public class Student {
    private static int counter = 0;
    private int id;

    private String name;
    private double grade;
```

private static int counter = 0;

private — הרשאת גישה

static — הגדרה כמשתנה מחלקה – רק מופע יחיד

= — אתחול בשלב ההגדרה

counter — יש גישה לכל מופע של המחלקה

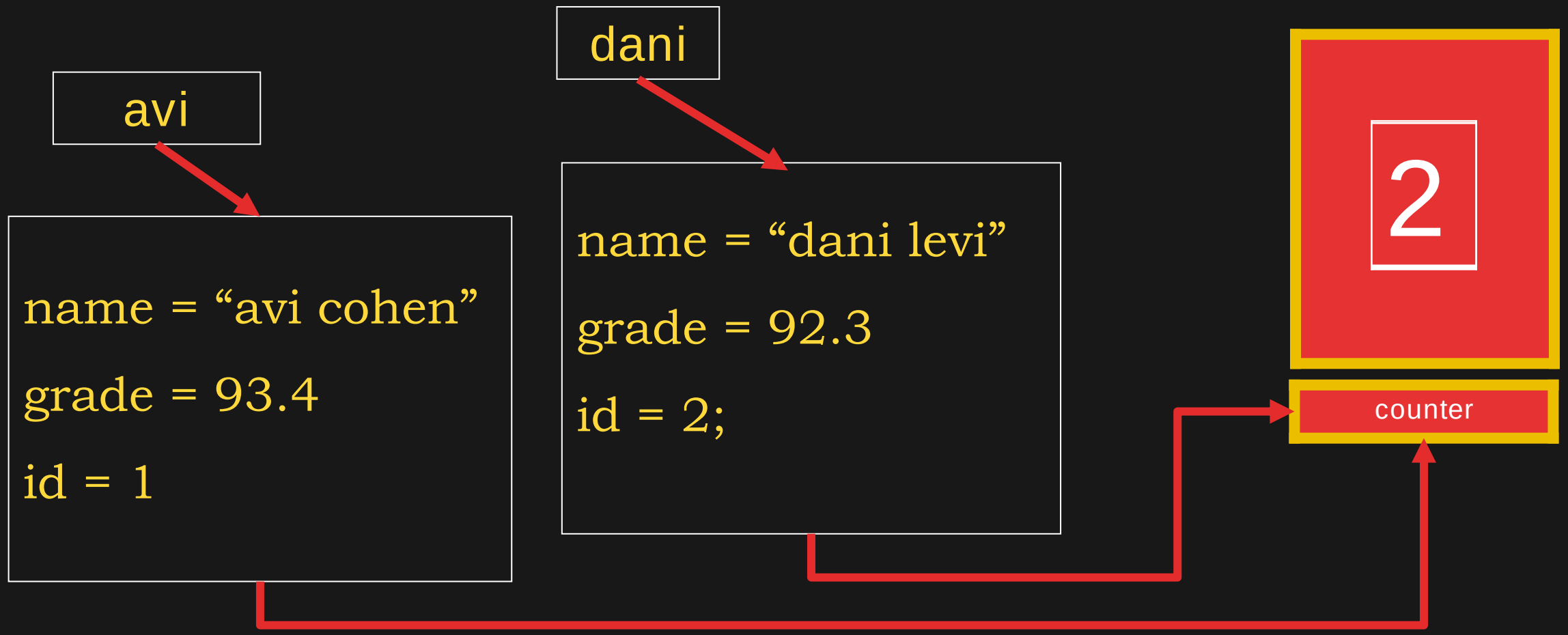
איבר של מחלקה

```
public class Student {
    private static int counter = 0;
    private int id;

    private String name;
    private double grade;
```

```java
public class Student {
    private String name;
    private double grade;
    private static int counter = 0;
    private int id;

    public Student(String name, double grade) {
        this.name = name;
        this.grade = grade;
        counter++;
        this.id = counter;
    }
}
```

יצירת עצמים

```
Student avi = new Student("avi cohen", 93.4);
Student dani = new Studen("dani_levi, 92.3);
```



תכונות מחלקה - class variables

תכונות השייכות למחלקה ולא למופע מסוים של המחלקה

תכונות אלו משקפות מצב משותף של כל העצמים שייוצרו מאותה מחלקה ולא משקפות את מצבו של עצם זה או אחר של המחלקה

שימוש במשתנה סטטי

אם תכונת המחלקה הוגדרה כפומבית אפשר לזמן אותה על ידי קריאה בשם המחלקה נקודה שם התכונה בדיוק כמו Math.Pi

למשל Bucket.counter

שימושים נפוצים – ערך קבוע (שם האוניברסיטה), מונה, Random, Scanner

שיטות מחלקה

כזכור הגישה לתכונות הינה באמצעות שיטות המשוייכות לעצם ספציפי
איך ניגש לתכונות מחלקה שאינן משוייכות לעצם ושאינן פומביות?
באמצעות **שיטות מחלקה**

```
public static int getCounter(){

        return counter;
}
```

פעולות סטטיות - תכונות סטטיות בלבד !

שימוש בשיטות מחלקה

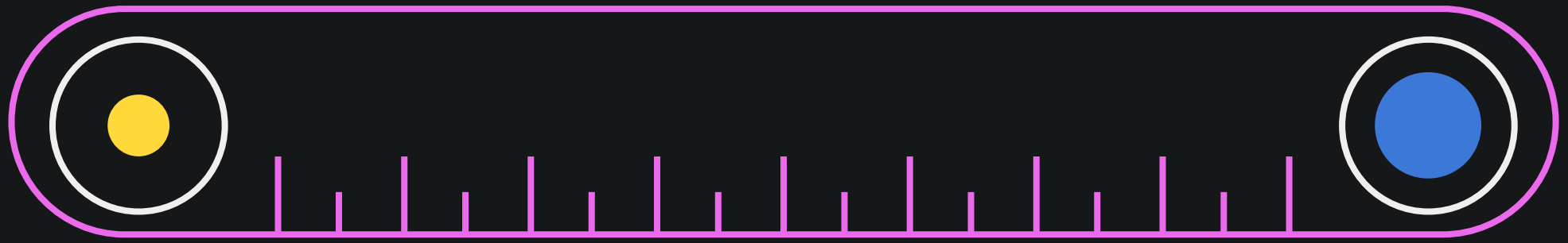
כמה דליים נוצרו ?

```
public static void main(String[] args){
        int num = Bucket.getCoutner();
        System.out.println(num);
}
```

פנייה אל המחלקה,

אליה שייכת השיטה

המשתנה הסטטי חוסך זכרון
מאפשר קשר בין עצמים
מאפשר ניהול עצמי המחלקה
מבצע בפשטות פעולה שהן מסובכות בלעדיו.



שובר את ההגיון של עצם ומחלקה סגורים.
פתח להשפעות הדדיות של עצמים זה על זה.
פתח לטעויות.

קבועים בשימוש המחלקה

- אין טעם ליצור **משתנה קבוע** עבור כל עצם
- כיוון שהוא קבוע אין חשש שישונה
- לכן נהוג ליצור קבועים של מחלקה באופן הבא

```
public static final double GRAVITY = 9.81;
```

- לקבועים אלה ניתן לגשת גם מחוץ למחלקה:

```
double newton =
        avi.getMass() * Student.GRAVITY;
```

זאת בהנחה שהגדרנו את הקבוע במחלקה Student

תכונות מחלקה קבועות

public ***static final*** **int** CAPACITY = 5

הגדרה זו פירושה ש

- יתקבל עותק אחד ויחיד בלתי ניתן לשינוי של התכונה קיבולת, שלו ערך קבוע, גלוי ומשותף לכל המופעים של המחלקה.
- ההגדרה הזו קובעת שזהו קבוע (**final**), פומבי (**public**), והוא מטיפוס הנתונים **int**.