# מעבדה – JVM Memory Model & GC

## מטרה

הכרות עם GC ומניפולציות GC

## דרישות

- פרויקט Java פשוט
- התקנה של VisualVM או עבודה עם כלי ניטור דומה

## משימה 1

בדקו באיזה סוג GC תשמש JVM במכונה שלכם?

## משימה 2

הריצו לולאת while true ובגוף הלולאה:

- יצירת אובייקט חדש

פתחו את כלי הניטור VisualVM וראו את התנהגות ה-Generational Heap

## משימה 3

הריצו לולאת while true ובגוף הלולאה:

- יצירת אובייקט חדש
- השמת ה-reference בערך null
- הפעלת המתודה System.gc()

פתחו את כלי הניטור VisualVM וראו את התנהגות ה-Generational Heap

## משימה 4

צרו את מחלקת Student באופן הבא:

- שדה name מסוג String
- בנאי מלא
- ממשו את המתודה toString
- דרסו את המתודה finalize() כך שתדפיס: "Finalizing Student"

צרו את מחלקת App באופן הבא:

```
public class App {


    public static void main(String[] args) {
        Student student = new Student("Kobi");
        // TODO: Code goes here


    }

    public static Student getFromCache(){
        // TODO: Code goes here
        return null;
    }

    public static void putInCatch(Student s){
        // TODO: Code goes here

    }
}
```

**סעיף א**

נהלו את ה-Cache באמצעות Weak Reference

1. צרו אובייקט Student בשם kobi
2. שמרו את האובייקט ל-cache
3. אתחלו את ה-reference לאובייקט המקורי ב-null
4. הפעילו GC
5. אחזרו את האובייקט מה-cache

מה ההתנהגות?

**סעיף ב**

כעת, בצעו שינויים קלים והחליפו את ניהול הCache באמצעות Soft Reference

מה ההתנהגות?

**סעיף ג**

עמדו על ההבדלים בין Strong, Weak & Soft References

# בהצלחה!