עבודה עצמית – שבוע 4

לפניכם קוד המחלקה Box:

```
public class Box {
	private static int boxCnt;
	private int i;
	private int j;

	public void print(String boxName){
		System.out.println(String.format(
				"%s: i = %d, j = %d, boxCnt = %d",
					boxName,this.i, this.j, boxCnt));
	}

	public static void main(String[] args){
		Box b1 = new Box();
		b1.print("b1");
	}
}
```

הריצו את הקוד הבא ובדקו את ההדפסה. שימו לב למספר תופעות עליהן דיברנו בשבוע שעבר:

1. למחלקה Box יש בנאי ברירת מחדל, זאת כיוון שלא הגדרנו שום בנאי אחר.
2. כל השדות של box – הסטטיים ושדות המופע מאותחלים לערכים הדיפולטיים של הטיפוסים שלהם, במקרה זה, כל ה int-ים מאותחלים לאפסים.

כעת, נגדיר בנאי למחלקה Box. הוסיפו את הבנאי הבא:

```
	public Box(int ii, int jj){
		this.i = ii;
		this.j = jj;
		boxCnt++;
	}
```

שמות הפרמטרים שהבנאי מקבל יכולים להיות זהים, ויכולים להיות גם שונים משמות השדות. שימו לב ששם השדה נקבע בהצהרה עליו, ולא בתוך הבנאי. ההצהרה, כזכור מכילה את הטיפוס של השדה, וכמו כן מאפיינים נוספים (סטטי\מופע, ניראות).
בשורה השלישית של הבנאי אנחנו עושים שימוש ב boxCnt. מכיוון שזה שדה סטטי ולא קשור למופע מסויים של המחלקה, אנחנו לא כותבים this לפניו (זה יתקמפל אבל לא נכון קונספטואלית).

הקוד של main יפסיק להתקמפל מכיוון שהבנאי הדיפולטי לא קיים יותר ברגע שהוספנו בנאי אחר. אם נרצה שבמחלקה יהיה בנאי ריק, נצטרך להגדיר אותו בעצמנו.

כעת עדכנו את הקוד של main לקוד הבא:

```
Box b1 = new Box(1,2);
b1.print("b1");
Box b2 = new Box(3,4);
b2.print("b2");
Box b3 = new Box(4,5);
b3.print("b3");
b1.print("b1");
b2.print("b2");
```

שימו לב להבדל בין השדות הסטטיים לשדות המופע. עבור כל אחד משלושת המופעים של Box יש שדה בשם i ושדה בשם j, ולכל אחד מהם יש את הערכים שלו. השדה boxCnt הוא שדה גלובלי יחיד, כלומר, קיים רק מופע אחד שלו. כל אחד מהבנאים מעדכן אותו ולכן לאחר היצירה של b3, בהדפסה של כל האובייקטים מקבלים את אותו הערך של boxCnt.

כעת, נוסיף עוד בנאי למחלקה: נרצה לייצר בנאי שמקבל רק את ערך ה i, וערך ה j שלו יהיה קבוע ל 5.
אפשרות ראשונה היא להוסיף את הבנאי הבא:

```
public Box(int ii){
	this.i = ii;
	this.j = 5;
	boxCnt++;
}
```

העמסת בנאים, כזכור, היא חוקית. הבעיה בבנאי הזה היא שהוא מייצר שכפול קוד, כיוון שהוא קורא מבצע קוד דומה לבנאי שכבר קיים. הגיוני ששני הבאים ישתמשו אחד בשני. הבנאי החדש צריך לקרוא לבנאי שכבר קיים. כיצד מבצעים את זה?
נסיון ראשון:

```
public Box(int ii){
	new Box(ii, 5);
}
```

הוסיפו את הבנאי הבא למחלקה, וב main הוסיפו את השורות:

```
Box b4 = new Box(14);
b4.print("b4");
```

הרצת הקוד מראה שהבנאי לא עובד כנדרש. מדוע?
מה שבעצם קורה הוא שבקוד של הבנאי החדש אנחנו מייצרים אובייקט חדש של Box, ולא עושים איתו כלום. הפקודה new מבוצעת פעמיים ביצירה של b4 – פעם אחת בשורה שבה הוא נוצר ופעם אחת בתוך הבנאי. האובייקט שנוצר בתוך הבנאי נוצר ואין אליו שום מצביע. האובייקט שאליו מצביע b4 נוצר עם הערכים הדיפולטיים שלו, והבנאי לא מעדכן אותם.
כיצד עושים את זה נכון? משתמשים במילה השמורה this אותה כבר ראינו.
החליפו את הבנאי הלא תקין במימוש הבא:

```
public Box(int ii){
	this(ii, 5);
}
```

וודאו כי ההדפסות של b4 תואמות למצופה.
שימו לב שאם יש עוד פעולות שצריכות להתבצע בבנאי, הן צריכות לבוא אחרי הקריאה ל this. כלומר, הקריאה ל this תמיד תהיה השורה הראשונה. נסביר את הסיבה לזה בהמשך הקורס.