



Software 1

Recitation No. 3

Java memory model

Passing arguments by value

העברת ארגומנטים

■ בקריאה לשרות נעשית השמה של ערכי הארגומנטים לפרמטרים של השרות לפי הסדר לפני ביצוע גוף השרות.

```
public class MyClass {  
    public static void foo(int a, int b){  
        ...  
    }  
  
    public static void main(String[] args) {  
        ...  
        ...  
        foo(2,6);  
    }  
}
```

a = 2
b = 6

העברת ארגומנטים

- צורת העברת ארגומנטים ב-Java היא by value:
- ערך הארגומנט המועבר לשרות מועתק לפרמטר הפורמלי
- בפרט, כשהארגומנט המועבר הוא הפנייה (reference):
 - ההפנייה מועתקת לפרמטר הפורמלי
 - אין העתקה של העצם שאליו מתייחסים
- בשפות אחרות כגון C++ קיימת גם שיטת העברת ארגומנטים by reference

העברת ארגומנטים

```
public class CallByValue {  
  
    public static void setToFive(int arg){  
        arg = 5;  
    }  
  
    public static void main(String[] args) {  
        int x = 0;  
        System.out.println("Before: x=" + x);  
        setToFive(x);  
        System.out.println("After: x=" + x);  
    }  
}
```

מודל הזיכרון של Java

STACK

משתנים מקומיים
וארגומנטים – כל מתודה
משתמשת באזור מסוים של
המחסנית (frame)

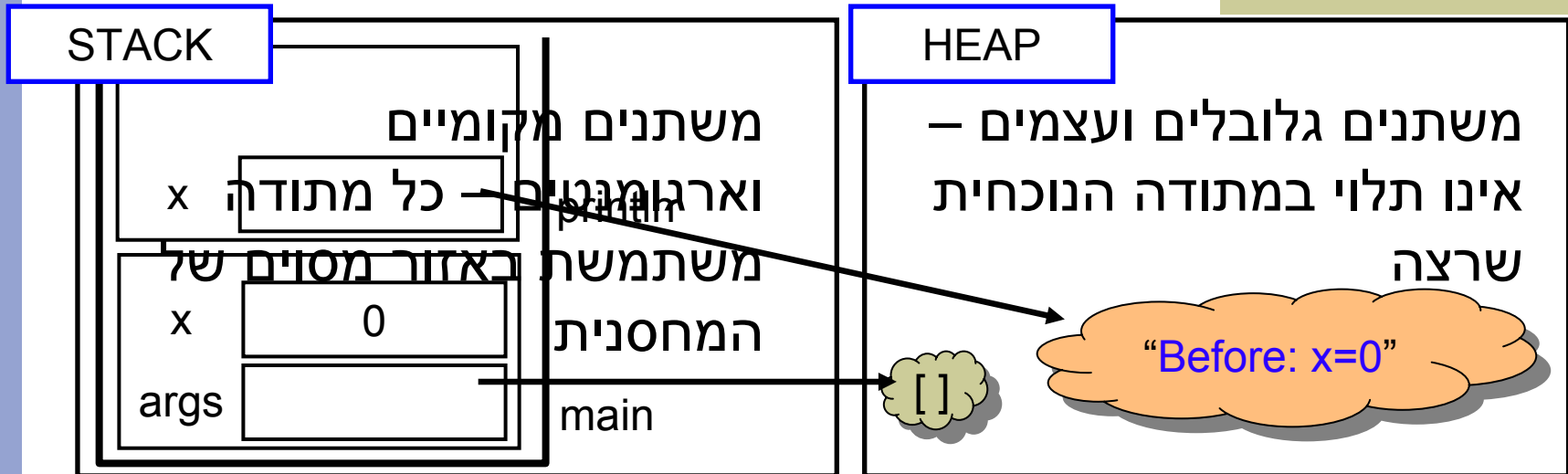
HEAP

משתנים גלובלים ועצמים –
אינו תלוי במתודה הנוכחית
שרצה

CODE

קוד התוכנית

Primitives by value

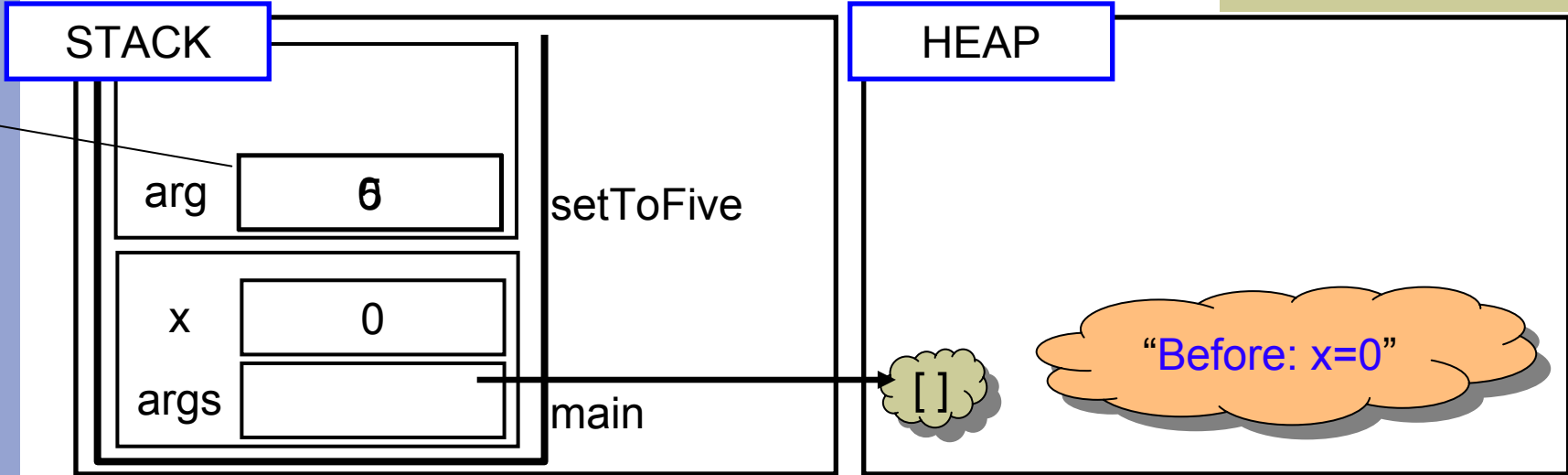


```
public class CallByValue {  
  
    public static void setToFive(int arg){  
        arg = 5;  
    }  
  
    public static void main(String[] args) {  
        int x = 0;  
        System.out.println("Before: x=" + x);  
        setToFive(x);  
        System.out.println("After: x=" + x);  
    }  
}
```

CODE

הפרמטר arg קיבל את הערך האקטואלי של הארגומנט x

Primitives by value



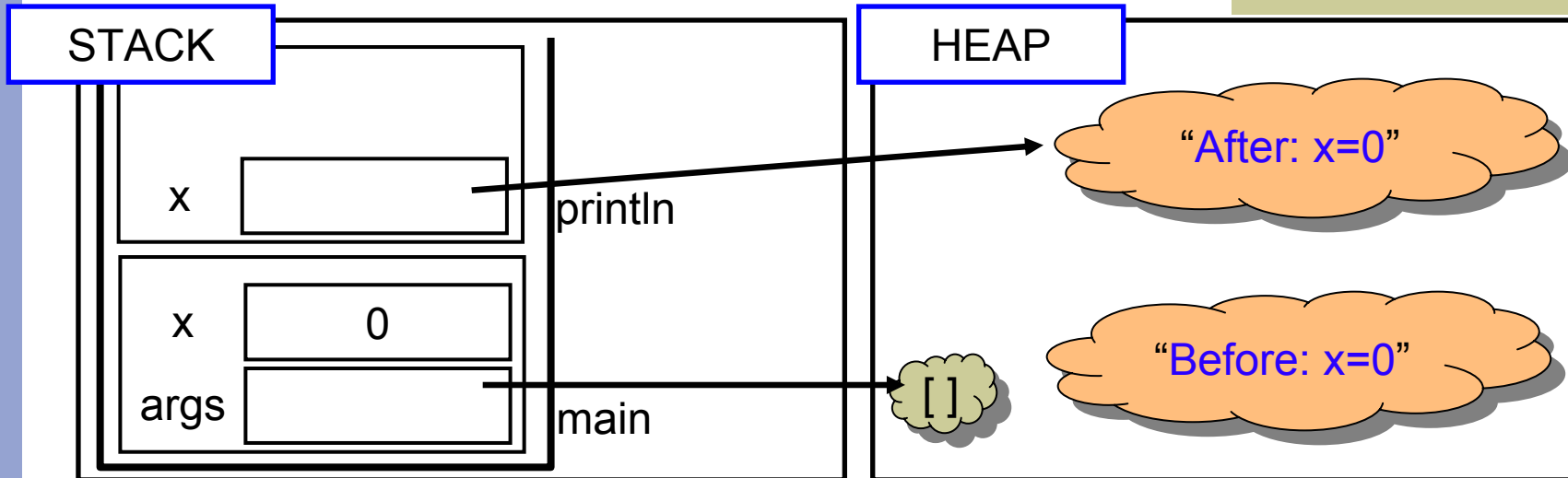
```
public class CallByValue {
    public static void setToFive(int arg){
        arg = 5;
    }

    public static void main(String[] args) {
        int x = 0;
        System.out.println("Before: x=" + x);
        setToFive(x);
        System.out.println("After: x=" + x);
    }
}
```

CODE

לא אחת `setToFive` סימית
אחרות הן מקומות שונים
עבורה עלולה `Stack` משחרר

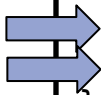
Primitives by value



```
public class CallByValue {  
  
    public static void setToFive(int arg){  
        arg = 5;  
    }  
  
    public static void main(String[] args) {  
        int x = 0;  
        System.out.println("Before: x=" + x);  
        setToFive(x);  
        System.out.println("After: x=" + x);  
    }  
}
```

CODE

לאחר ש `main` יומסי את
אוצרת מקום שמה ושקצה
עבורה על ה- Stack משוחרר



משתנים מקומיים

- בתוך בלוק של מתודה מוכרים:
 - המשתנים הגלובלים
 - הפרמטרים הפורמליים של המתודה
 - משתנים מקומיים המוכרזים בבלוק או בבלוק עוטף
- מתודה לא מכירה משתני מחסנית שאינם באזור שהוקצה לה (frame)
- לשתי מתודות שונות יכולים להיות אותם שמות משתנים

References by value

```
import java.util.Arrays;

public class CallByValue {

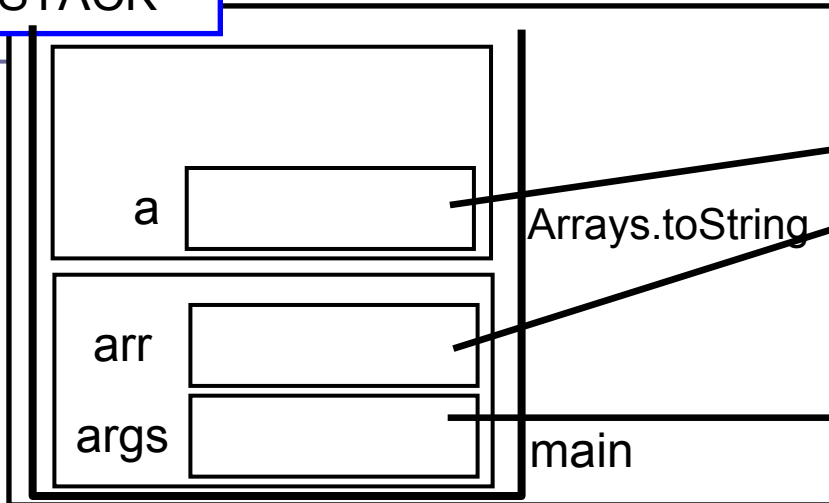
    public static void setToZero(int [] arr){
        arr = new int[3];
    }

    public static void main(String[] args) {
        int [] arr = {4,5};
        System.out.println("Before: x=" +
            Arrays.toString(arr));
        setToZero(arr);
        System.out.println("After: x=" +
            Arrays.toString(arr));
    }
}
```

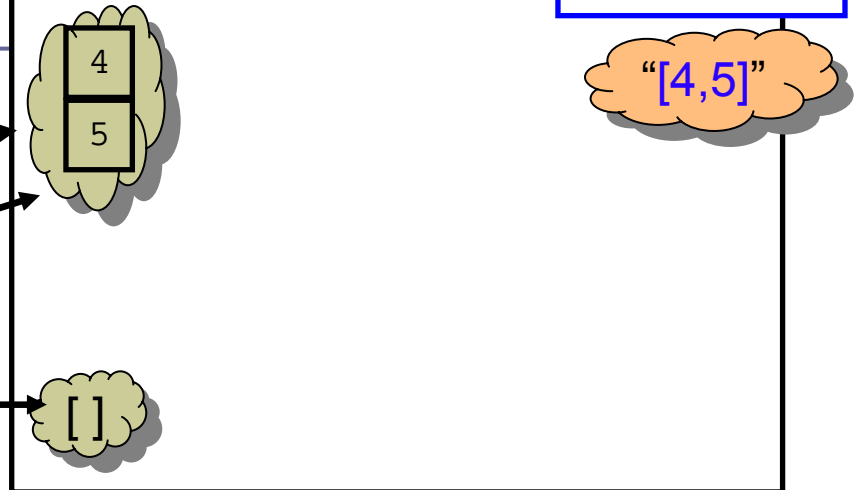
המשתנה המקומי שמועבר
הוא מטיפוס הפנייה

References by value

STACK



HEAP

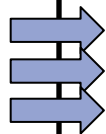


```
public class CallByValue {
```

```
    public static void setToZero(int [] arr){  
        arr = new int[3];  
    }
```

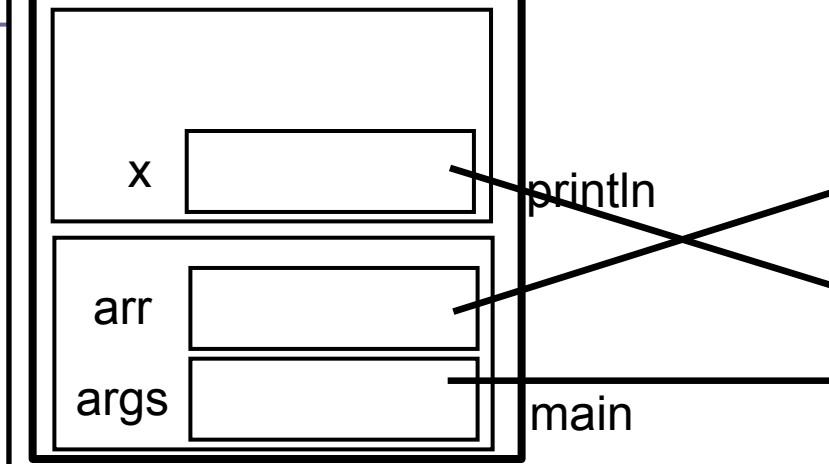
```
    public static void main(String[] args) {  
        int [] arr = {4,5};  
        System.out.println("Before: x=" + Arrays.toString(arr));  
        setToZero(arr);  
        System.out.println("After: x=" + Arrays.toString(arr));  
    }  
}
```

CODE

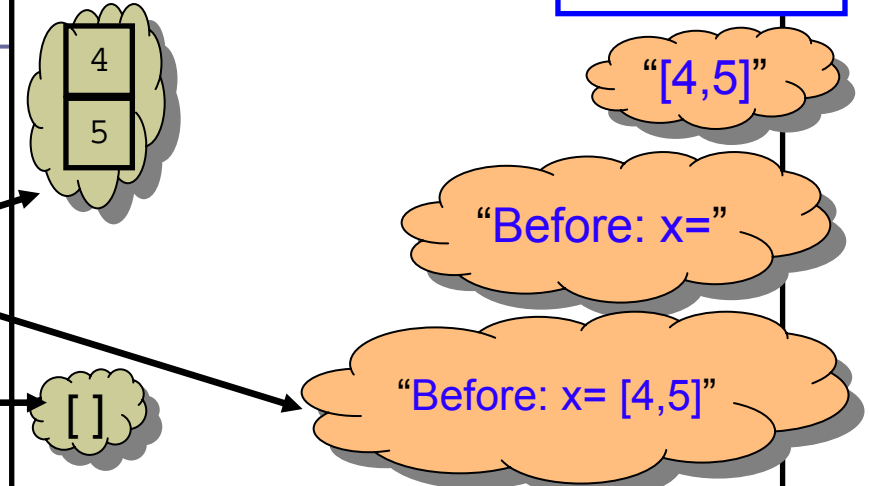


References by value

STACK



HEAP

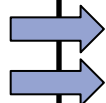


```
public class CallByValue {
```

```
    public static void setToZero(int [] arr){  
        arr = new int[3];  
    }
```

```
    public static void main(String[] args) {  
        int [] arr = {4,5};  
        System.out.println("Before: x=" + Arrays.toString(arr));  
        setToZero(arr);  
        System.out.println("After: x=" + Arrays.toString(arr));  
    }  
}
```

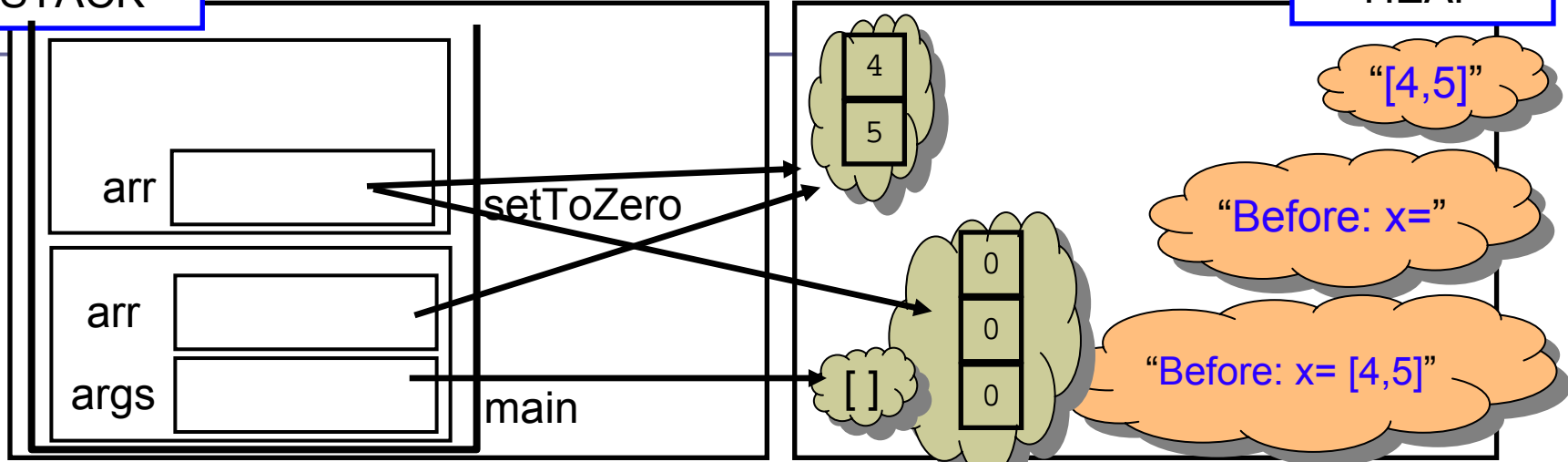
CODE



References by value

STACK

HEAP



```
public class CallByValue {
```

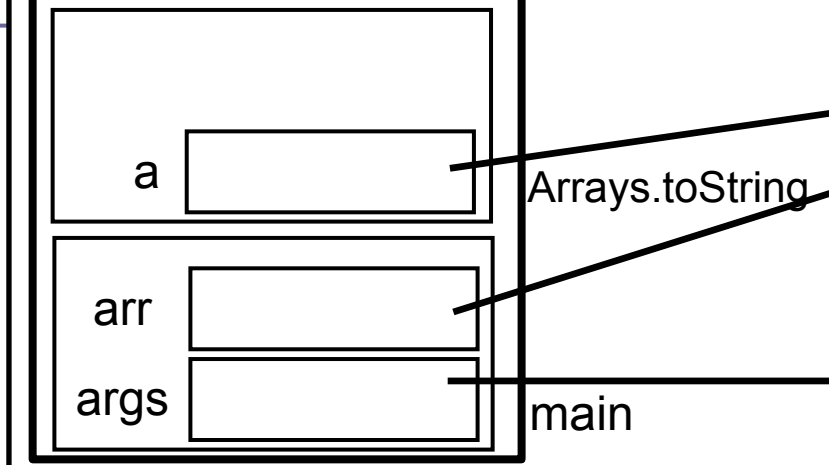
```
    public static void setToZero(int [] arr){  
        arr = new int[3];  
    }
```

```
    public static void main(String[] args) {  
        int [] arr = {4,5};  
        System.out.println("Before: x=" + Arrays.toString(arr));  
        setToZero(arr);  
        System.out.println("After: x=" + Arrays.toString(arr));  
    }  
}
```

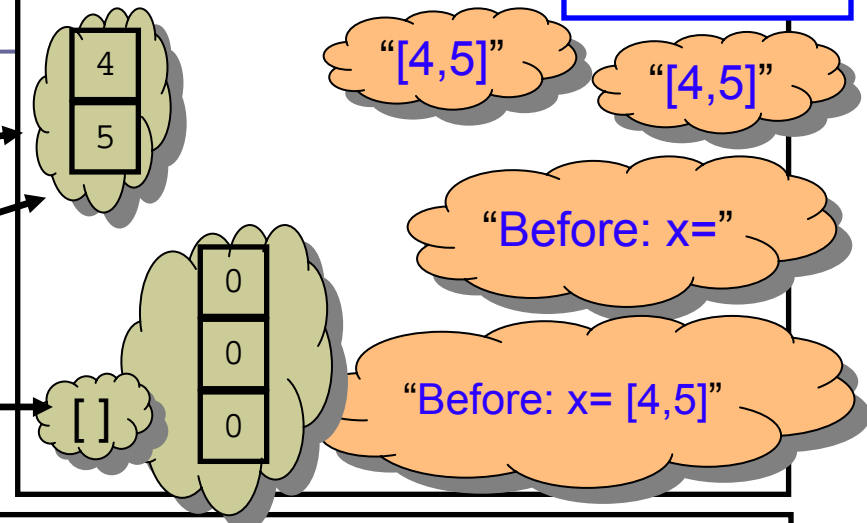
CODE

References by value

STACK



HEAP



```
public class CallByValue {
```

```
    public static void setToZero(int [] arr){  
        arr = new int[3];  
    }
```

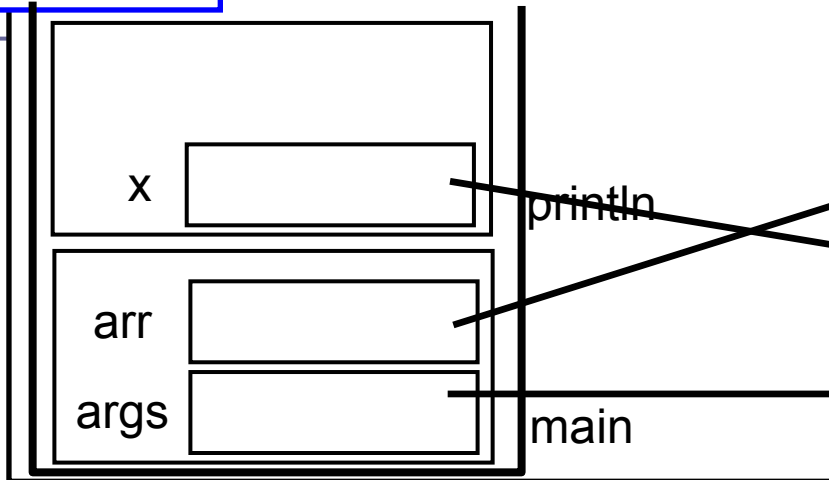
```
    public static void main(String[] args) {  
        int [] arr = {4,5};  
        System.out.println("Before: x=" + Arrays.toString(arr));  
        setToZero(arr);  
        System.out.println("After: x=" + Arrays.toString(arr));  
    }  
}
```

CODE

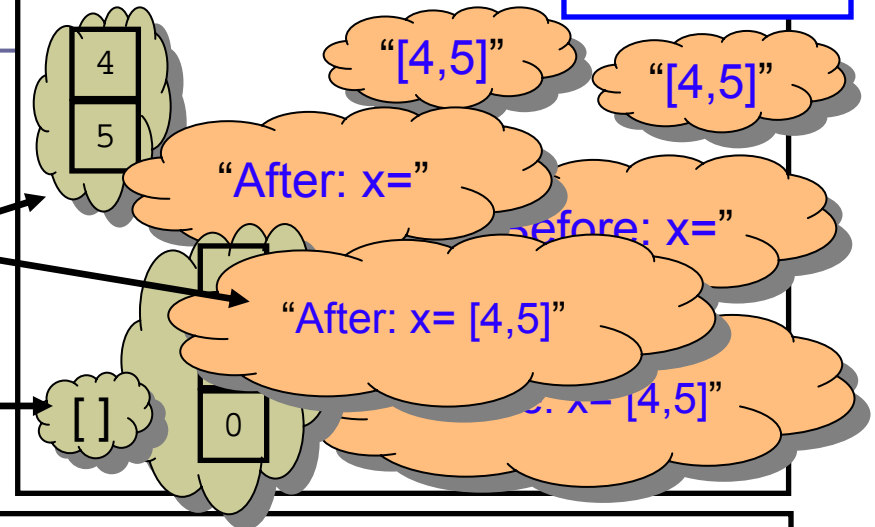


References by value

STACK



HEAP

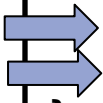


```
public class CallByValue {
```

```
    public static void setToZero(int [] arr){  
        arr = new int[3];  
    }
```

```
    public static void main(String[] args) {  
        int [] arr = {4,5};  
        System.out.println("Before: x=" + Arrays.toString(arr));  
        setToZero(arr);  
        System.out.println("After: x=" + Arrays.toString(arr));  
    }  
}
```

CODE



הפונקציה הנקראת והעולם שבחוץ

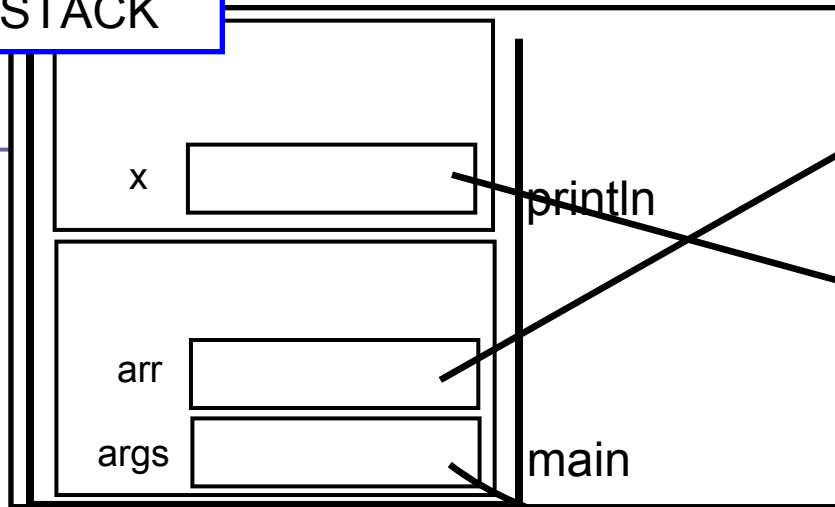
- בשיטת העברה by value לא יעזור למתודה לשנות את הארגומנט שקיבלה, מכיוון שהיא מקבלת עותק
- אז איך יכולה מתודה להשפיע על ערכים במתודה שקראה לה?
- ע"י ערך מוחזר
- ע"י גישה למשתנים או עצמים שהוקצו ב- Heap
- מתודות שמשנות את תמונת הזיכרון נקראות בהקשרים מסוימים Mutators או Transformers

מה מדפיסה התוכנית הבאה?

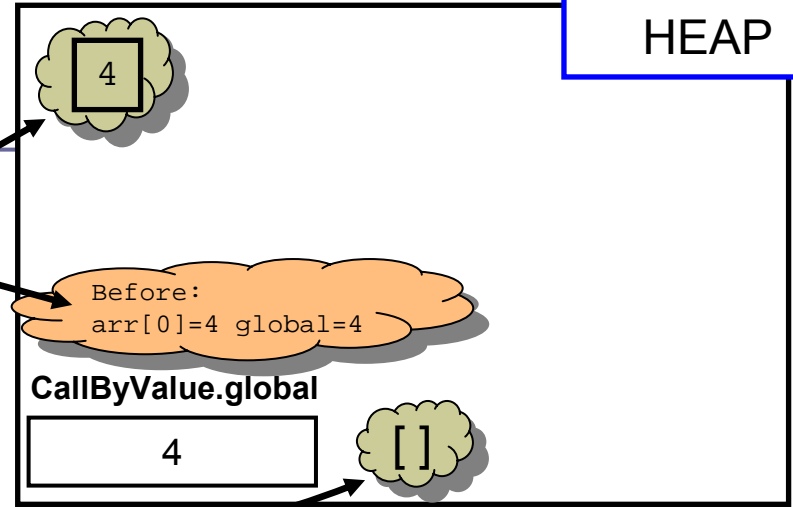
```
public class CallByValue {  
  
    static int global = 4;  
  
    public static int increment(int [] arr){  
        int local = 5;  
        arr[0]++;  
        global++;  
        return local;  
    }  
  
    public static void main(String[] args) {  
        int [] arr = {4};  
        System.out.println("Before: arr[0]=" + arr[0] +  
                            "\tglobal=" + global);  
        int result = increment(arr);  
        System.out.println("After:  arr[0]=" + arr[0] +  
                            "\tglobal=" + global);  
        System.out.println("result = " + result);  
    }  
}
```

Heap, Heap – Hooray!

STACK



HEAP



```
public class CallByValue {
```

```
    static int global = 4;
```

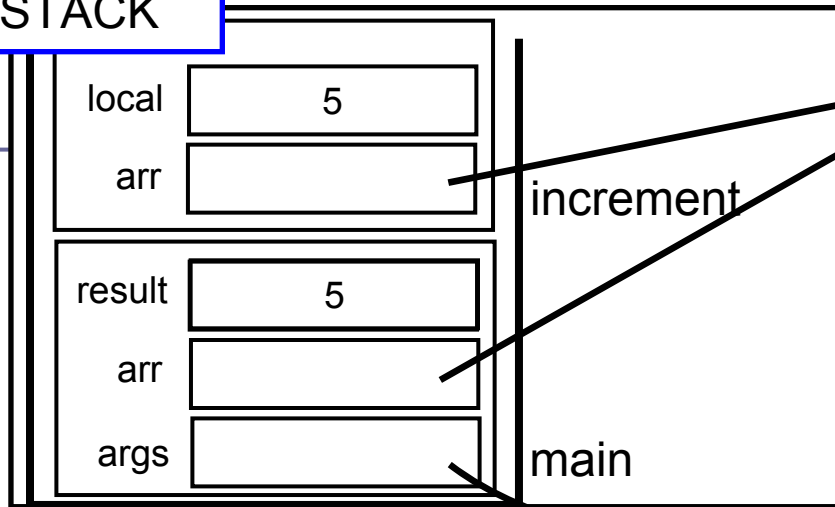
```
    public static int increment(int [] arr){  
        int local = 5;  
        arr[0]++;  
        global++;  
        return local;  
    }
```

```
    public static void main(String[] args) {  
        int [] arr = {4};  
        System.out.println("Before: arr[0]=" + arr[0] + "\tglobal=" + global);  
        int result = increment(arr);  
        System.out.println("After:  arr[0]=" + arr[0] + "\tglobal=" + global);  
        System.out.println("result = " + result);  
    }  
}
```

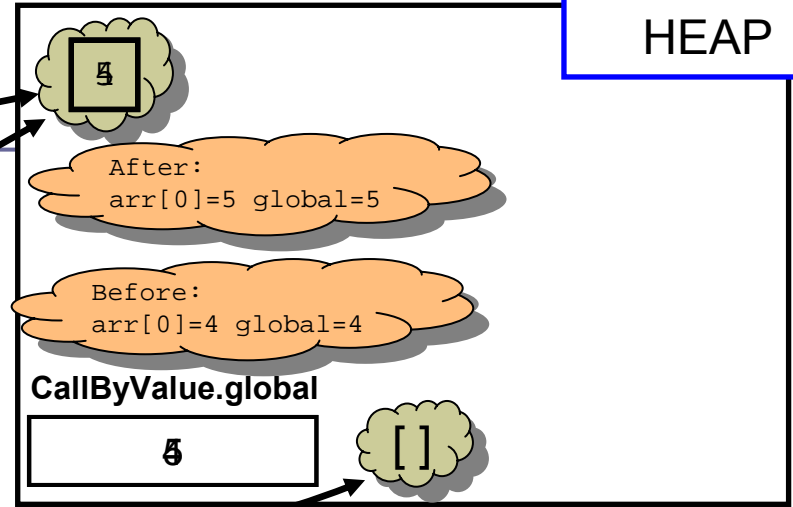
CODE

Heap, Heap – Hooray!

STACK



HEAP



```
public class CallByValue {
```

```
    static int global = 4;
```

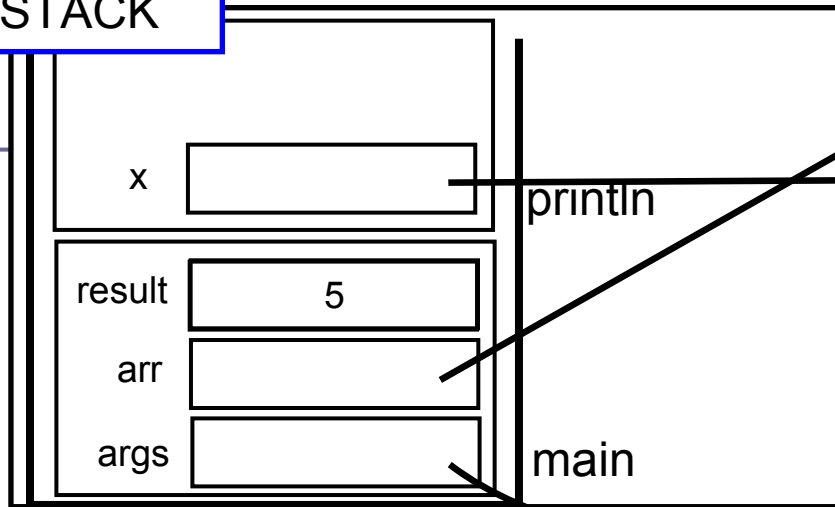
```
    public static int increment(int [] arr){  
        int local = 5;  
        arr[0]++;  
        global++;  
        return local;  
    }
```

```
    public static void main(String[] args) {  
        int [] arr = {4};  
        System.out.println("Before: arr[0]=" + arr[0] + "\tglobal=" + global);  
        int result = increment(arr);  
        System.out.println("After: arr[0]=" + arr[0] + "\tglobal=" + global);  
        System.out.println("result = " + result);  
    }  
}
```

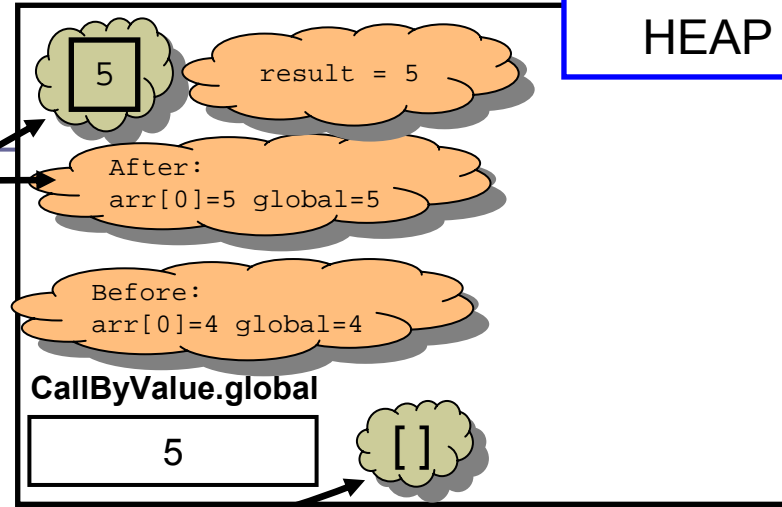
CODE

Heap, Heap – Hooray!

STACK

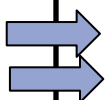


HEAP



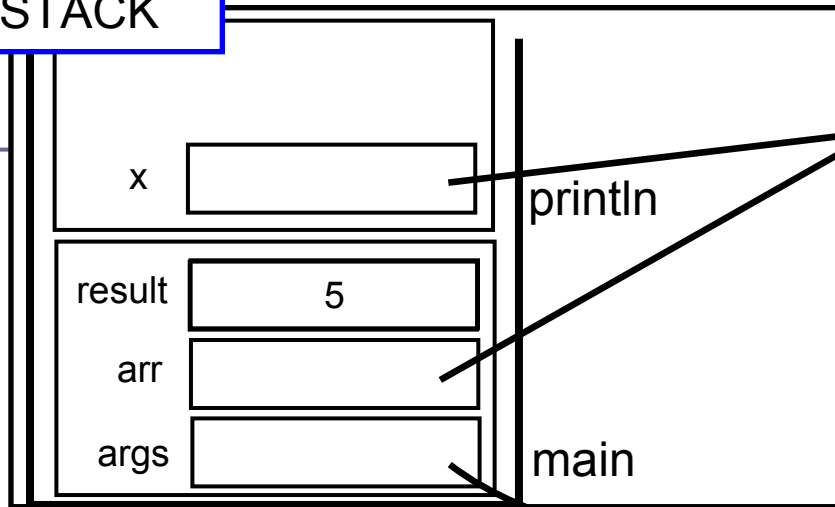
```
public class CallByValue {  
  
    static int global = 4;  
  
    public static int increment(int [] arr){  
        int local = 5;  
        arr[0]++;  
        global++;  
        return local;  
    }  
  
    public static void main(String[] args) {  
        int [] arr = {4};  
        System.out.println("Before: arr[0]=" + arr[0] + "\tglobal=" + global);  
        int result = increment(arr);  
        System.out.println("After:  arr[0]=" + arr[0] + "\tglobal=" + global);  
        System.out.println("result = " + result);  
    }  
}
```

CODE

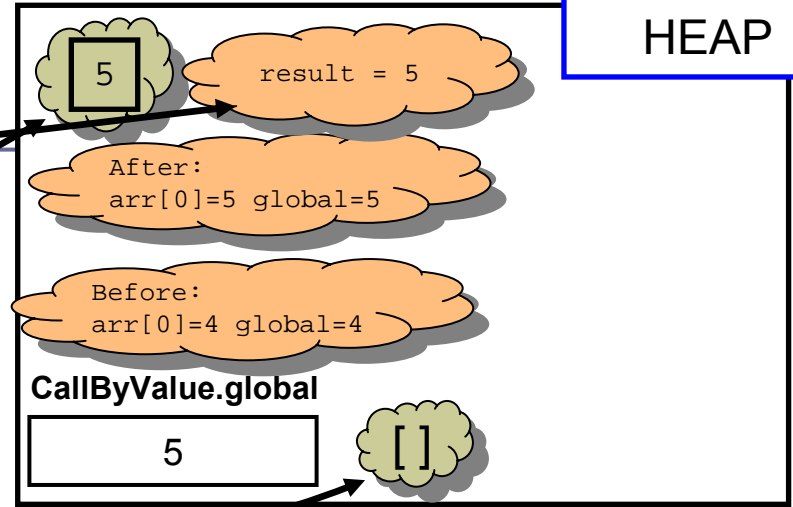


Heap, Heap – Hooray!

STACK

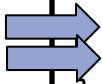


HEAP



```
public class CallByValue {  
  
    static int global = 4;  
  
    public static int increment(int [] arr){  
        int local = 5;  
        arr[0]++;  
        global++;  
        return local;  
    }  
  
    public static void main(String[] args) {  
        int [] arr = {4};  
        System.out.println("Before: arr[0]=" + arr[0] + "\tglobal=" + global);  
        int result = increment(arr);  
        System.out.println("After: arr[0]=" + arr[0] + "\tglobal=" + global);  
        System.out.println("result = " + result);  
    }  
}
```

CODE



משתני פלט (Output Parameters)

- איך נכתוב פונקציה שצריכה להחזיר יותר מערך אחד?
 - הפונקציה תחזיר מערך
- ומה אם הפונקציה צריכה להחזיר נתונים מטיפוסים שונים?
 - דרך אפשרית: הפונקציה תקבל כארגומנטים הפניות לעצמים שהוקצו ע"י הקורא לפונקציה ותמלא אותם בערכים.
- דרך נוספת תלמדו בהמשך הקורס

גושי אתחול סטטיים

- אתחול משתנים גלובלים סטטים מתרחש מיד לאחר טעינת המחלקה לזיכרון ולפני פונקציית ה-main
- ניתן לבצע פעולות נוספות (בדרך כלל אתחולים למניהם) מיד לאחר טעינת המחלקה לזיכרון
 - פעולות אלו יש לציין בתוך בלוק **static**

תמונת הזיכרון האמיתית

- מודל הזיכרון שתואר כאן הוא פשטני – פרטים רבים נוספים נשמרים על המחסנית וב- Heap
- תמונת הזיכרון האמיתית והמדויקת היא תלוית סביבה ועשויה להשתנות בריצות בסביבות השונות
- נושא זה נידון בהרחבה בקורס "קומפילציה"

