

תוכנה 1

תרגול 5: שאר ירקות
רובי בוים ומתי שמרת

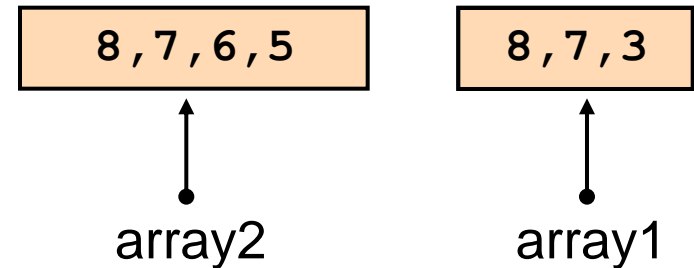
דרכים נוספות להעתקה

■ הפונקציה `arraycopy` במחלקה `java.lang.System`
מאפשרת העתקת תוכנו של מערך אחד לאחר

```
public static void arraycopy(Object src, int srcPos,  
                               Object dest, int destPos,  
                               int length)
```

```
System.arraycopy(array2, 0, array1, 0, 2);
```

1,2 in array1 are
replaced with 8,7



■ Details:

<http://java.sun.com/javase/6/docs/api/java/lang/System.html>

המרת טיפוסים פרימיטיביים

Narrowing הצרה

- short to byte or char
- char to byte or short
- int to byte, short, or char
- long to byte, short, char, or int
- float to byte, short, char, int, or long
- double to byte, short, char, int, long, or float

Widening הרחבה

- byte to short, int, long, float, or double
- short to int, long, float, or double
- char to int, long, float, or double
- int to long, float, or double
- long to float or double
- float to double

הרחבה

- מתבצעת באופן אוטומטי ע"י הקומפיילר
- הרחבה מטיפוס אינטגרלי אחד לאחר וכן מ float ל double היא מדויקת
- בהרחבה של int או long ל float או של long ל double ייתכן אובדן דיוק.

```
public static void wideningExample() {  
    int ibig = 1234567890;  
    float fapprox = ibig;  
    System.out.println(ibig - (int) fapprox);  
    long lbig = 998877665544332211L;  
    double dapprox = lbig;  
    System.out.println(lbig - (long) dapprox);  
}
```

-46

51

הצרה

- דורשת המרה מפורשת (casting) ע"י המתכנת
- ייתכן אובדן של מידע

```
private static void wideningExample() {  
    float fmin = Float.NEGATIVE_INFINITY;  
    float fmax = Float.POSITIVE_INFINITY;  
    long: -9223372036854775808..9223372036854775807  
    println("long: " + (long) fmin + ".." + (long) fmax);  
    println("int: " + (int) fmin + ".." + (int) fmax);  
    println("short: " + (short) fmin + ".." + (short) fmax);  
    println("char: " + (int) (char) fmin + ".." + (int) (char) fmax);  
    println("byte: " + (byte) fmin + ".." + (byte) fmax);  
}
```

הצרה

- דורשת המרה מפורשת (casting) ע"י המתכנת
- ייתכן אובדן של מידע

```
private static void wideningExample() {  
    float fmin = Float.NEGATIVE_INFINITY;  
    float fmax = Float.POSITIVE_INFINITY;  
  
    println("l int: -2147483648..2147483647  
    println("int: " + (int) fmin + ".." + (int) fmax);  
    println("short: " + (short) fmin + ".." + (short) fmax);  
    println("char: " + (int) (char) fmin + ".."+ (int) (char) fmax);  
    println("byte: " + (byte) fmin + ".." + (byte) fmax);  
}
```

הצרה

- דורשת המרה מפורשת (casting) ע"י המתכנת
- ייתכן אובדן של מידע

```
private static void wideningExample() {  
    float fmin = Float.NEGATIVE_INFINITY;  
    float fmax = Float.POSITIVE_INFINITY;  
  
    println("long: " + (long) fmin + " .. " + (long) fmax);  
    println("int: " + (int) fmin + " .. " + (int) fmax);  
    println("short: " + (short) fmin + " .. " + (short) fmax);  
    println("char: " + (int) (char) fmin + " .. " + (int) (char) fmax);  
    println("byte: " + (byte) fmin + " .. " + (byte) fmax);  
}
```

הצרה

- דורשת המרה מפורשת (casting) ע"י המתכנת
- ייתכן אובדן של מידע

```
private static void wideningExample() {  
    float fmin = Float.NEGATIVE_INFINITY;  
    float fmax = Float.POSITIVE_INFINITY;  
  
    println("long: " + (long) fmin + ".." + (long) fmax);  
    println("int: " + (int) fmin + " " + (int) fmax);  
    println("short: " + (short) fmin + " " + (short) fmax);  
    println("char: 0..65535");  
    println("char: " + (int) (char) fmin + ".." + (int) (char) fmax);  
    println("byte: " + (byte) fmin + ".." + (byte) fmax);  
}
```


הצרה

- דורשת המרה מפורשת (casting) ע"י המתכנת
- ייתכן אובדן של מידע


```
private static void wideningExample() {
    float fmin = Float.NEGATIVE_INFINITY;
    float fmax = Float.POSITIVE_INFINITY;

    println("long: " + (long) fmin + ".." + (long) fmax);
    println("int: " + (int) fmin + ".." + (int) fmax);
    println("short: " + (short) fmin + ".." + (short) fmax);
    println("char: " + (char) fmin + ".." + (char) fmax);
    println("byte: 0..-1");
    println("byte: " + (byte) fmin + ".." + (byte) fmax);
}
```

Numeric Promotion

- הרחבה של האופרנדים בביטוי מתימטי
- טיפוס משותף לשני האופרנדים לצורך ביצוע הפעולה
 - אם אחד האופרנדים הוא double גם השני יומר ל double
 - אם אחד האופרנדים הוא float גם השני יומר ל float
 - אם אחד האופרנדים הוא long גם השני יומר ל long
 - שני האופרנדים יומרו ל int

עוד דוגמאות

```
public static void main(String[] args) {  
    long l = 20000000000+20000000000; // l == -294967296  
    int i = (int) 1.999999999; // i == 1  
    float f = (float) 1.999999999; // f == 2  
    f = 5/2; // f == 2  
    f = (float) (5/2); // f == 2  
    f = (float) 5/2; // f == 2.5  
    f = 5 / (float) 2; // f == 2.5  
    short a = 2; // why is this ok?  
     short c = a*a; // compilation error: cannot convert int to short  
}
```

עוד על המרות ב-

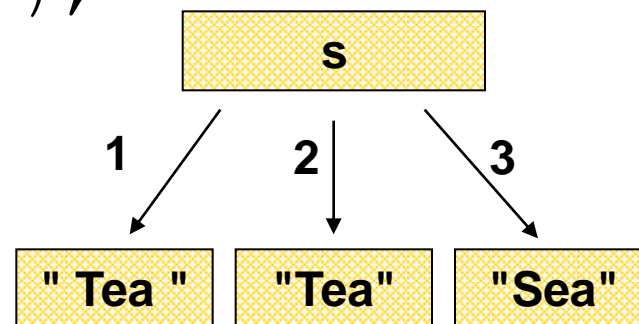
http://java.sun.com/docs/books/jls/third_edition/html/conversions.html

מחרוזות

■ מרגע שנוצרה המחרוזת היא אינה ניתנת לשינוי (immutable)

■ ההפניה למחרוזת כמובן יכולה להשתנות

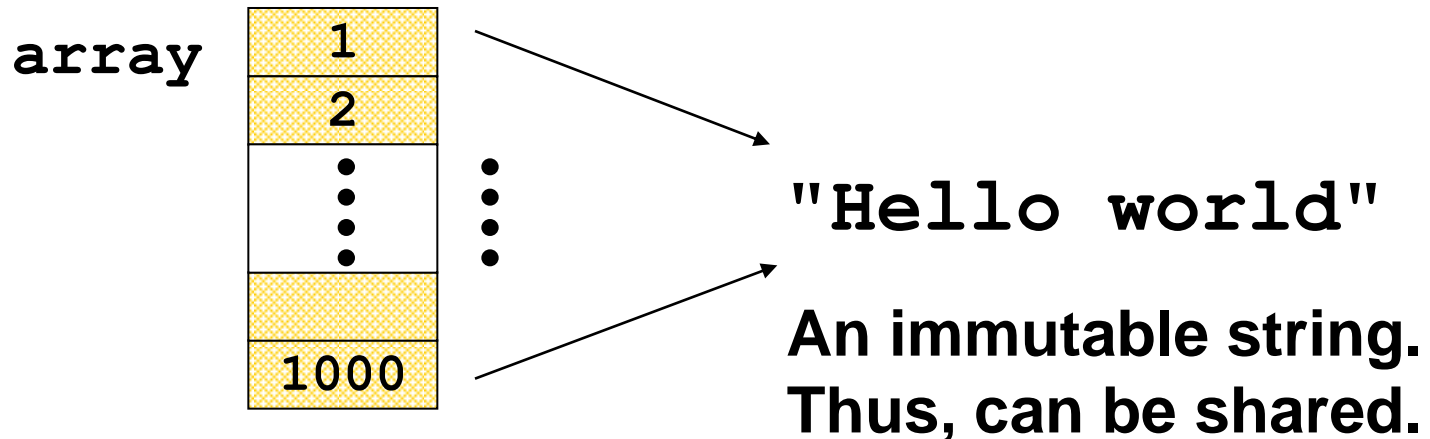
```
String s = " Tea ";  
s = s.trim();  
s = s.replace('T', 'S');
```



Interning

■ מכיוון שמחרוזות הן קבועות ניתן לשתף אותן

```
String[] array = new String[1000];  
for (int i = 0; i < array.length; i++) {  
    array[i] = "Hello world ";  
}
```



Interning-ל אמת

```
String hello = "Hello", lo = "lo";
```

String literals

```
System.out.println(hello == "Hello");
```

```
System.out.println(Other.hello == hello);
```

```
System.out.println(hello == ("Hel"+"lo"));
```

```
System.out.println(hello == ("Hel"+lo));
```

```
System.out.println(hello == ("Hel"+lo).intern());
```

Interning-ל אמת

```
String hello = "Hello", lo = "lo";
```

```
System.out.println(hello == "Hello");
```

System.out.println(hello == "Hello");
Literal strings within the same class represent references to the same String

```
System.out.println(hello == ("Hel"+"lo"));
```

```
System.out.println(hello == ("Hel"+lo));
```

```
System.out.println(hello == ("Hel"+lo).intern());
```

Interning-ל אמת

```
String hello = "Hello", lo = "lo";
```

```
System.out.println(hello == "Hello");
```

```
System.out.println(Other.hello == hello);
```

System.out.println(hello == ("Hel"+lo));

Literal strings within different classes represent references to the same String object

```
System.out.println(hello == ("Hel"+lo));
```

```
System.out.println(hello == ("Hel"+lo).intern());
```


Interning-ל אמת

```
String hello = "Hello", lo = "lo";
```

```
System.out.println(hello == "Hello");
```

```
System.out.println(Other.hello == hello);
```

```
System.out.println(hello == ("Hel"+"lo"));
```

Syst Strings computed by constant expressions are computed at compile time and then treated as if they were literals

```
System.out.println(hello == ("Hel"+lo).intern());
```

Interning-ל אמת

```
String hello = "Hello", lo = "lo";
```

```
System.out.println(hello == "Hello");
```

```
System.out.println(Other.hello == hello);
```

```
System.out.println(hello == ("Hel"+"lo"));
```

```
System.out.println(hello == ("Hel"+lo));
```

Strings computed by concatenation at run time are newly created and therefore distinct

Interning-ל אמת

```
String hello = "Hello", lo = "lo";
```

```
System.out.println(hello == "Hello");
```

```
System.out.println(Other.hello == hello);
```

```
System.out.println(hello == ("Hel"+"lo"));
```

```
System.out.println(hello == ("Hel"+lo));
```

```
System.out.println(hello == ("Hel"+lo).intern());
```

Explicitly interning a String returns a reference to the interned String object. If such a String was previously interned the returned value will refer to that object

static field

- שדה המוגדר static (class variable) אינו משויך לאובייקט מסוים
- קיימת בדיוק התגלמות אחת שלו
- אם שדה אינו מוגדר static (instance variable) בכל פעם שנוצר אובייקט חדש נוצר משתנה חדש המשויך לאובייקט זה.

static method

■ מתודה סטטית (class method) אינה משויכת לאובייקט

■ אין אובייקט נוכחי (this)

■ מתודה שאינה סטטית (instance method) נקראת תמיד תוך התייחסות לאובייקט מסוים. אובייקט זה הוא האובייקט הנוכחי (this) במתודה

■ נוכל לגשת רק לשדות שהוגדרו סטטיים

static example

```
class Point {  
    int x, y, useCount;  
    Point(int x, int y) { this.x = x; this.y = y; }  
    static Point origin = new Point(0, 0);  
}
```

```
class Test {  
    public static void main(String[] args) {  
        Point p = new Point(1,1);  
        Point q = new Point(2,2);
```

```
        p.x = 3; p.y = 3; p.useCount++;  
        p.origin.useCount++;
```

גישה לשדה סטטי
יש להשתמש בשם מחלקה ולא
באובייקט

```
        System.out.println("(" + q.x + ", " + q.y + "  
        System.out.println(q.useCount);  
        System.out.println(q.origin == Point.origin);  
        System.out.println(q.origin.useCount);
```

true – גישה לאותה הפניה

```
    }
```

```
}
```

final fields

- ניתן לבצע השמה יחידה למשתנה המוגדר `final`
- נאכף על ידי הקומפיילר
- שדה סטטי יאותחל יחד עם ההגדרה*
- שדה מופע חייב להיות מאותחל עד סיום כל אחד מבנאי המחלקה
- יחד עם ההגדרה או בבנאי
- שדה שהוגדר `final` אבל לא אותחל עם הגדרתו נקרא `blank final`