

# HASHCODE-1 EQUALS

---

# תזכורת: המחלקה Object

```
package java.lang;

public class Object {
    public final native Class<?> getClass();

    public native int hashCode();

    public boolean equals(Object obj) {
        return (this == obj);
    }

    protected native Object clone() throws CloneNotSupportedException;

    public String toString() {
        return getClass().getName() + "@" +
            Integer.toHexString(hashCode());
    }
    ...
}
```

# מה יודפס?

```
public class Name {
    private String first, last;
    ...

    public static void main(String[] args) {
        Name name1 = new Name("Mickey", "Mouse");
        Name name2 = new Name("Mickey", "Mouse");
        System.out.println(name1.equals(name2));

        List<Name> names = new ArrayList<Name>();
        names.add(name1);
        System.out.println(names.contains(name2));
    }
}
```

# הבעיה

- רצינו השוואה לפי תוכן אבל לא דרסנו את equals
- מימוש ברירת המחדל הוא השוואה של מצביעים

```
public class Object {  
    ...  
    public boolean equals(Object obj) {  
        return (this == obj);  
    }  
    ...  
}
```

# החזקה של equals

- **רפלקסיבי**

- `x.equals(x)` יחזיר `true`

- **סימטרי**

- `x.equals(y)` יחזיר `true` אם `y.equals(x)` יחזיר `true`

- **טרנזיטיבי**

- אם `x.equals(y)` מחזיר `true` וגם `y.equals(z)` מחזיר `true` אז `x.equals(z)`

- **עקבי**

- סדרת קריאות ל `x.equals(y)` תחזיר `true` (או `false`) באופן עקבי אם מידע שדרוש לצורך ההשוואה לא השתנה

- **השוואה ל null**

- `x.equals(null)` תמיד תחזיר `false`

# מתכון ל equals

```
public boolean equals(Object obj) {  
    if (this == obj)  
        return true;  
    if (obj == null)  
        return false;  
    if (getClass() != obj.getClass())  
        return false;  
    Name other = (Name) obj;  
    return first.equals(other.first) &&  
        last.equals(other.last);  
}
```

1. ודאו כי הארגומנט אינו מצביע לאובייקט הנוכחי

2. ודאו כי הארגומנט אינו null

3. ודאו כי הארגומנט  
הוא מהטיפוס  
המתאים להשוואה

4. המירו את הארגומנט לטיפוס הנכון

5. לכל שדה "משמעותי", בידקו ששדה זה בארגומנט תואם לשדה באובייקט הנוכחי

## טעות נפוצה

- להגדיר את הפונקציה equals כך:

```
public boolean equals(Name name) {  
    return first.equals(other.first) &&  
        last.equals(other.last);  
}
```

- זו אינה דריסה (overriding) אלא העמסה (overloading)
- שימוש ב @Override יתריע על הבעיה

# אז הכל בסדר?

```
public class Name {  
    ...  
    @Override public equals(Object obj) {  
        ...  
    }  
  
    public static void main(String[] args) {  
        Name name1 = new Name("Mickey", "Mouse");  
        Name name2 = new Name("Mickey", "Mouse");  
        System.out.println(name1.equals(name2));  
  
        List<Name> names = new ArrayList<Name>();  
        names.add(name1);  
        System.out.println(names.contains(name2));  
    }  
}
```

true יודפס

true יודפס



# כמעט – עדיין יש בעיה בשימוש באוספים שמבוססים על Hash-Table

```
public class Name {  
    ...  
    @Override public equals(Object obj) {  
        ...  
    }  
  
    public static void main(String[] args) {  
        Name name1 = new Name("Mickey", "Mouse");  
        Name name2 = new Name("Mickey", "Mouse");  
        System.out.println(name1.equals(name2));  
  
        Set<Name> names = new HashSet<Name>();  
        names.add(name1);  
        System.out.println(names.contains(name2));  
    }  
}
```

true יודפס

false יודפס

hashCode | equals

חובה לדרוס את hashCode בכל מחלקה  
שדורסת את equals!

# החזרה של hashCode

## • עקביות

- מחזירה אותו ערך עבור כל הקריאות באותה ריצה, אלא אם השתנה מידע שבשימוש בהשוואת **equals** של המחלקה

## • שוויון

- אם שני אובייקטים שווים לפי הגדרת equals אזי hashCode תחזיר ערך זהה עבורם

## • חוסר שוויון

- אם שני אובייקטים אינם שווים לפי equals לא מובטח ש hashCode תחזיר ערכים שונים
- החזרת ערכים שונים יכולה לשפר ביצועים של מבני נתונים המבוססים על hashing (לדוגמא, HashSet ו HashMap)

# מימוש hashCode

```
@Override public int hashCode() {  
    return 31 * first.hashCode() + last.hashCode();  
}
```

- השתדלו לייצר hash כך שלאובייקטים שונים יהיה ערך hash שונה
- המימוש החוקי הגרוע ביותר (לעולם לא לממש כך!)

```
@Override public int hashCode() {  
    return 42;  
}
```

# תמיכה באקליפס

- אקליפס תומך ביצירה אוטומטית (ומשולבת) של equals ו- hashCode
  - בתפריט Source ניתן למצוא Generate hashCode() and equals()