
# Java lecturer 15

# אז על מה נדבר היום...

- Clone
- Set
- HashMap
- Collection

# Cloneable

בשיעורים הקודמים ראינו שכדי לשכפל אובייקטים אנחנו לא יכולים לעשות השמה בדרך הבאה:

```
Fish nemo = new Fish("Orange",2);
Fish nemo2 = nemo;
```

משום שבדרך הזאת נוצר רק עוד מצביע ולא משתכפל האובייקט.
הדרך שבה עשינו זאת היא לדוגמא באמצעות הבנאי:

```
public Fish(Fish other) {
   super(other);
   this.swimSpeed = other.swimSpeed;
}
```

עכשיו נוכל לשכפל את המחלקה בקלות

```
Fish nemo = new Fish("Orange",2);
Fish nemo2 = new Fish(nemo);
```

RAVTECH

# Cloneable

מה נעשה במקרה שיש לנו מערך הטרוגני של בסיס ויורשים?
אם נרצה לשכפל את כל איברי המערך נצטרך קודם כל לבדוק מהו טיפוסו של כל איבר ורק אז לשכפלו באמצעות מעבר בבנאי של המחלקה המתאימה.
לדוגמא יש לנו מערך של חיות ונרצה לשכפל את כולו למערך אחר:

```
public static void main(String[] args) {
   Animal[] noahArk = new Animal[3];
...
   Animal[] dupArk = new Animal[noahArk.length];
   for (int i=0 ; i < dupArk.length ; i++) {
       if (noahArk[i] instanceof Cat)
           dupArk[i] = new SiamiCat((SiamiCat)noahArk[i]);
       else if (noahArk[i] instanceof Horse)
           dupArk[i] = new Horse((Horse)noahArk[i]);
       else if (noahArk[i] instanceof Fish)
           dupArk[i] = new Fish((Fish)noahArk[i]);
   }
}
```

נראה קצת ארוך... וגם לא כזה יעיל במקרה שנרצה להוסיף עוד סוגים של חיות.

# Cloneable

היה נראה הרבה יותר טוב אם התוכנית שלנו לא הייתה צריכה לבדוק את האובייקט ופשוט הייתה נראית כך:

```
public static void main(String[] args) {
   Animal[] noahArk = new Animal[3];
...
   Animal[] dupArk = new Animal[noahArk.length];
   for (int i=0 ; i < dupArk.length ; i++) {

     dupArk[i] = noahArk[i].clone();

   }
}
```

הקסם הזה נעשה באמצעות המחלקה Object שיש לה את המתודה clone אשר יודעת לשכפל את נתוני האובייקט.
לכאורה פשוט.
אבל יש כמה דברים שאנחנו צריכים לעשות קודם.

# Cloneable

- המתודה clone היא private ולכן עלינו לדרוס אותה כדי שתהייה public.
הסיבה לכך שהיא private מכיוון שישנם טיפוסים שלא נרצה לאפשר את שכפולם.

- אם נרצה לעבור על מערך ולשכפל רק את האיברים שאפשרנו להם להיות משוכפלים למעשה לא תהיה לנו דרך לדעת מי ניתן לשכפול ומי לא.

לכן, מחלקה שרוצה לממש את המתודה clone צריכה להוסיף 3 דברים:

1. דריסת המתודה clone

2. הצהרה שניתן לשכפל אובייקטים מהמחלקה באמצעות המתודה clone על ידי מימוש הממשק הריק Cloneable שהוא למעשה ממשק ללא שיטות (marking interface)

3. וידוא שהמתכנת לא שכח לציין מימוש של הממשק Cloneable ע"י ציון זריקת החריגה CloneNotSupportedException אין סיבה שחריגה זו תזרק יותר מפעם אחת, שכן אם קרתה, המתכנת צריך לדעת להוסיף את ההצהרה על מימוש הממשק Cloneable

# Cloneable

אז איך נראה המימוש של מתודת ה clone

```
public abstract class Animal implements Cloneable {
...

    public Animal(String color, int numOfLegs) {
        this.color = color;
        this.numOfLegs = numOfLegs;
    }


    @Override
    public Animal clone() throws CloneNotSupportedException {
        Return (Animal)super.clone();
    }
}
```

מימוש הממשק
הריק Cloneable

הצהרה כי המתודה זורקת את החריגה
CloneNotSupportedException

covariant return type
כדי שהמתודה לא תחזיר Object

RAVTECH

# Set

Set (קבוצה) הוא (סוג של...) מערך שמכיל כל ערך פעם אחת בלבד, כלומר לא יתכנו כפילויות של הערכים
אם ננסה להוסיף ערך שכבר קיים לא יבוצע
הסיבה ש-Set הוא לא מערך זה בגלל שהוא במהותו ממשק לכל המחלקות שרוצות לממש את המבנה נתונים הזה
ניתן 2 דוגמאות:

```
public static void main(String[] args) {

    Set<Integer> numbers = new LinkedHashSet<>();
    numbers.add(10);
    numbers.add(14);
    numbers.add(8);
    numbers.add(10);
    System.out.println(numbers);

    Set<Integer> numbers1 = new TreeSet<>();
    numbers1.add(10);
    numbers1.add(14);
    numbers1.add(8);
    System.out.println(numbers1);
}
```

LinkedHashSet הינה מחלקה
הממשת את הממשק set והיא שומרת
את האיברים לפי סדר הכנסתם

TreeSet הינה מחלקה הממשת את
הממשק set והיא שומרת את האיברים
ממוינים

*מה יודפס בכל פעם?

RAVTECH

# HashMap

- HashMap הינו אוסף של נתונים שמאפשר פניה לערך מסויים באמצעות מפתח כלשהו ולא בהכרח באמצעות אינדקס מספרי
- ניתן גם להסתכל עליו כעל מערך, שהאינדקסים שלו אינם בהכרח מטיפוס int
- נשתמש בו כאשר נרצה למפות ערכים ממפתח מסוים, לערך אחר
- כל מפתח יכול להופיע פעם בלבד
- ניסיון הוספה למפתח קיים ידרוס את ערכו הקוד
- לצורך הדוגמא נייצר מחלקה בשם MyDate שמייצגת תאריך.

```
public class MyDate {
   private int day, month, year;

   public MyDate(int day, int month, int year) {
       this.day = day;
       this.month = month;
       this.year = year;
   }
   @Override
   public String toString() {
       return day + "/" + month + "/" + year;
   }
}
```

RAVTECH

# HashMap

נבנה תוכנית שמציגה ימי הולדת.

```
public static void main(String[] args) {
    HashMap<String, MyDate> birthdays = new HashMap<>();
```

יצרנו מערך של תאריכים שהאינדקס הוא מסוג חרוזת.
הדרך להוסיף ערכים היא באמצעות השיטה put המקבלת מפתח וערך.

```
    birthdays.put("Sheldon", new MyDate(26, 2, 1980));
    birthdays.put("Penny", new MyDate(2, 12, 1986));
    birthdays.put("Rajesh", new MyDate(6, 10, 1981));
```

הדרך לקבל אובייקט (**הפנייה לאובייקט שנמצא במערך**) היא באמצעות המתודה get.
אם הערך אינו קיים הפונקצייה תחזיר null

```
    MyDate sheldonBirthday = birthdays.get("Sheldon");
```

אם נרצה לעבור על כל הערכים שנמצאים במערך היא באמצעות הדרך הבאה:

```
    for (Map.Entry<String, MyDate> e : birthdays.entrySet()) {
        System.out.println(e.getKey() + " --> " + e.getValue());
    }
}
```

RAVTECH

# HashMap

המשך...

```
public static void main(String[] args) {
   HashMap<String, MyDate> birthdays = new HashMap<>();

   birthdays.put("Sheldon", new MyDate(26, 2, 1980));
   birthdays.put("Penny", new MyDate(2, 12, 1986));
   birthdays.put("Rajesh", new MyDate(6, 10, 1981));
```

כדי להמנע מלקרוא לערך או למפתח שאינו קיים נוכל להשתמש בפונקציה contains:

```
   if(birthdays.containsKey("Rajesh")){
      System.out.println("dont speak");
   }
```

אותו הדבר גם לגבי בדיקת ערך:

```
  if (birthdays.containsValue( new MyDate(26, 2, 1980))){
      System.out.println("Bazinga");
  }
```

אם נוסיף ערך למפתח קיים הערך הקודם ידרס

```
}
```

# Collections

המחלקה Collections מכילה אוסף של שיטות סטטיות לעבודה עם כל סוגי האוספים
יש למחלקה הרבה מתודות נעבור לצורך הדוגמא על חלקם.
מוזמנים לבדוק בעצמכם את שאר היכולות של המחלקה.
דבר ראשון נבנה מערך של מספרים

```
ArrayList<Integer> numbers = new ArrayList<>();
for (int i = 10; i <= 50; i += 10) {
   numbers.add(i);
}
```

בשקף הבא נראה מה אנחנו יכולים לעשות איתו.

RAVTECH

# Collections

אז אחרי שיצרנו מערך נפעיל עליו את הפונקציות הבאות:

```
System.out.println(numbers);
Collections.shuffle(numbers);
System.out.println("The numbers after shuffle: " + numbers);
Collections.sort(numbers);
System.out.println("The numbers after sort: " + numbers);
Collections.reverse(numbers);
System.out.println("The numbers after reverse: " + numbers);
Collections.rotate(numbers,2);
System.out.println("The numbers after rotate twice: " + numbers);
Collections.replaceAll(numbers, 50,8);
System.out.println("The numbers after replace 50 in 8: " + numbers);
System.out.println("The max number is: " + Collections.max(numbers));
```

Output:
[10, 20, 30, 40, 50]
The numbers after shuffle: [40, 20, 30, 10, 50]
The numbers after sort: [10, 20, 30, 40, 50]
The numbers after reverse: [50, 40, 30, 20, 10]
The numbers after rotate twice: [20, 10, 50, 40, 30]
The numbers after replace 50 in 8: [20, 10, 8, 40, 30]
The max number is: 40

RAVTECH