

תוכנה 1

מסטר ב' תשע"ב

תרגול 6: מנשקים, דיאגרמות, ועוד *

יעל אמסטרדר ואלכסיי זגלסקי

== vs equals

```
Point p1 = new Point(1,2)
```

```
Point p2 = new Point(1,2)
```

```
p1 == p2
```

```
p1.equals(p2)
```

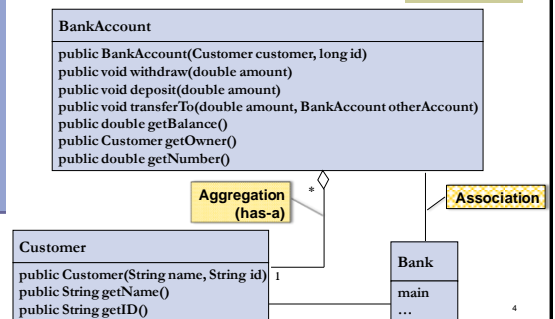
- מתי נכון להשתמש בכל אחד מהם?
- שימו לב, במחלקה שכתבתם בעצמכם יש לכתוב מתודת equals על מנת להשתמש בה.
- אין להשתמש במתודת "ברירת מחדל" (יוסבר בהמשך הקורס)

המערכת הבנקאית

- נתאר את מערכת התוכנה שלנו בעזרת דיאגרמות
- דיאגרמות סטטיות:
- תיאור היחסים בין המחלקות השונות במערכת
- דיאגרמות דינאמיות:
- תיאור ההתנהגות של המערכת בזמן ריצה
- מצב האובייקטים
- תיאור של תרחיש



Class Diagram



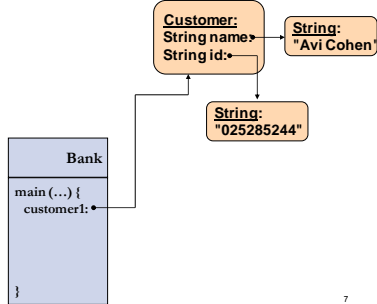
המחלקה Customer

```
public class Customer {
    public Customer(String name, String id) {
        this.name = name;
        this.id = id;
    }
    public String getName() {
        return name;
    }
    public String getID() {
        return id;
    }
    private String name;
    private String id;
}
```

Toy Bank Program

```
public class Bank {
    public static void main(String[] args) {
        Customer customer1 = new Customer("Avi Cohen", "025285244");
        Customer customer2 = new Customer("Rita Stein", "024847638");
        BankAccount account1 = new BankAccount(customer1, 1234);
        BankAccount account2 = new BankAccount(customer2, 5678);
        BankAccount account3 = new BankAccount(customer1, 2984);
        account1.deposit(1000);
        account2.deposit(500);
        account1.transferTo(100, account3);
        account2.withdraw(300);
        System.out.println("account1 has " + account1.getBalance());
        System.out.println("account2 has " + account2.getBalance());
    }
}
```

Object Diagram



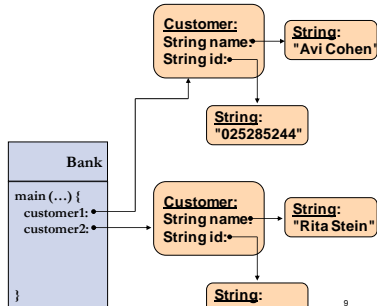
7

Toy Bank Program

```
public class Bank {
    public static void main(String[] args) {
        Customer customer1 = new Customer("Avi Cohen", "025285244");
        Customer customer2 = new Customer("Rita Stein", "024847638");
        BankAccount account1 = new BankAccount(customer1, 1234);
        BankAccount account2 = new BankAccount(customer2, 5678);
        BankAccount account3 = new BankAccount(customer1, 2984);
        account1.deposit(1000);
        account2.deposit(500);
        account1.transferTo(100, account3);
        account2.withdraw(300);
        System.out.println("account1 has " + account1.getBalance());
        System.out.println("account2 has " + account2.getBalance());
    }
}
```

8

Object Diagram



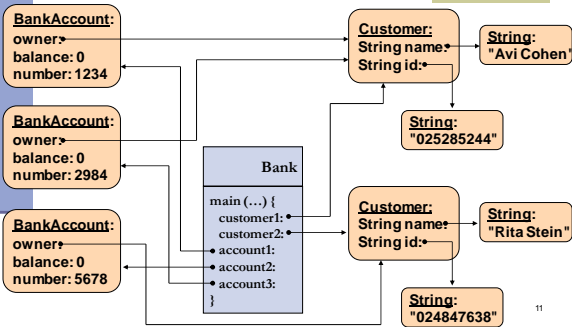
9

Toy Bank Program

```
public class Bank {
    public static void main(String[] args) {
        Customer customer1 = new Customer("Avi Cohen", "025285244");
        Customer customer2 = new Customer("Rita Stein", "024847638");
        BankAccount account1 = new BankAccount(customer1, 1234);
        BankAccount account2 = new BankAccount(customer2, 5678);
        BankAccount account3 = new BankAccount(customer1, 2984);
        account1.deposit(1000);
        account2.deposit(500);
        account1.transferTo(100, account3);
        account2.withdraw(300);
        System.out.println("account1 has " + account1.getBalance());
        System.out.println("account2 has " + account2.getBalance());
    }
}
```

10

Object Diagram

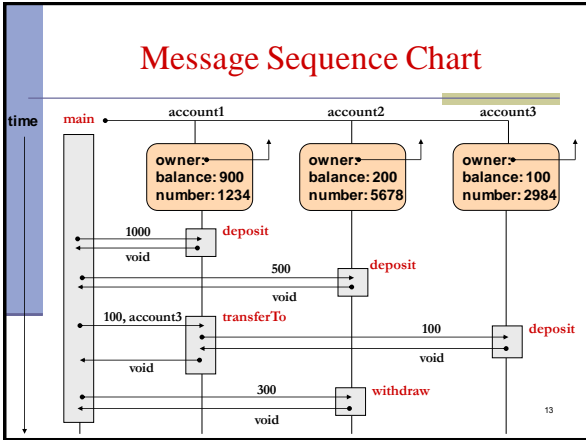


11

Message Sequence Chart

```
public class Bank {
    public static void main(String[] args) {
        Customer customer1 = new Customer("Avi Cohen", "025285244");
        Customer customer2 = new Customer("Rita Stein", "024847638");
        BankAccount account1 = new BankAccount(customer1, 1234);
        BankAccount account2 = new BankAccount(customer2, 5678);
        BankAccount account3 = new BankAccount(customer1, 2984);
        account1.deposit(1000);
        account2.deposit(500);
        account1.transferTo(100, account3);
        account2.withdraw(300);
        System.out.println("account1 has " + account1.getBalance());
        System.out.println("account2 has " + account2.getBalance());
    }
}
```

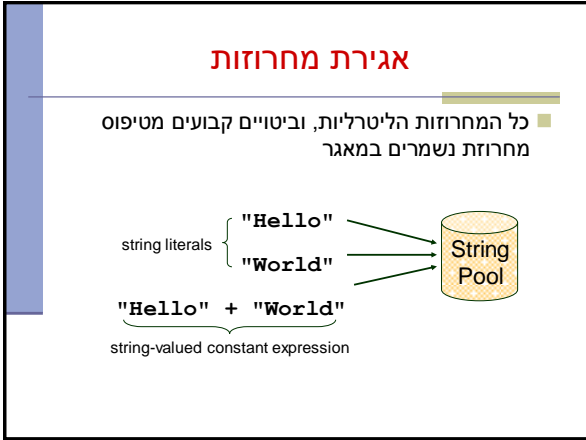
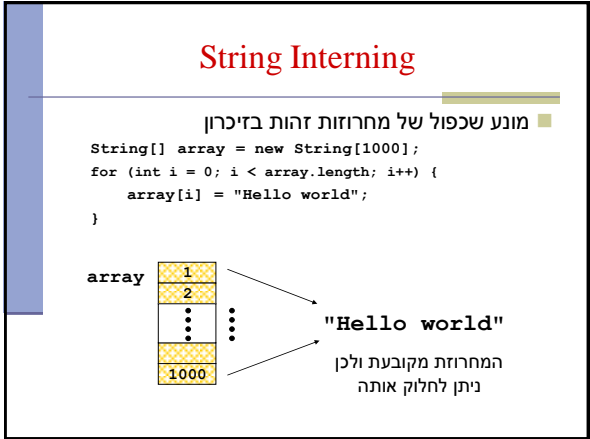
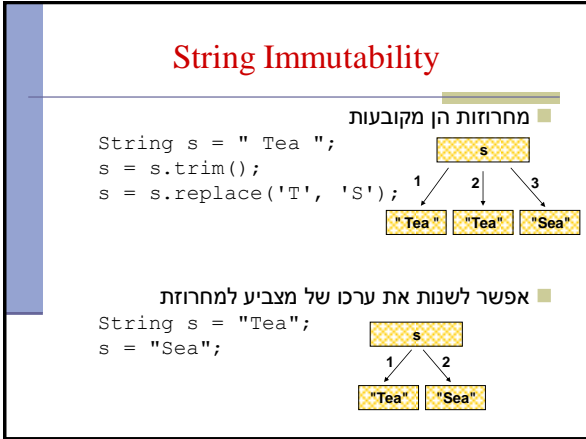
12



Output

```
public class Bank {
    public static void main(String[] args) {
        Customer customer1 = new Customer("Avi Cohen", "025285244");
        Customer customer2 = new Customer("Rita Stein", "024847638");
        BankAccount account1 = new BankAccount(customer1, 1234);
        BankAccount account2 = new BankAccount(customer2, 5678);
        BankAccount account3 = new BankAccount(customer1, 2984);
        account1.deposit(1000);
        account2.deposit(500);
        account1.transferTo(100, account3);
        account2.withdraw(300);
        System.out.println("account1 has " + account1.getBalance());
        System.out.println("account2 has " + account2.getBalance());
    }
}
```

output: account1 has 900.0
account2 has 200.0



הפנמת מחרוזות

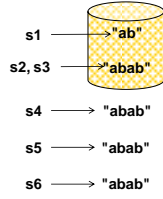
המחלקה String מחזיקה מאגר סטטי פרטי של מחרוזות.
 המחרוזות מוכנסות למאגר דרך המתודה intern

```
myString.intern() :
```

```
If ∃s ∈ pool : myString.equals(s)
    return s;
Else
    add myString to the pool
    return myString;
```

הפנמת מחרזות - דוגמא

```
String s1 = "ab";
String s2 = "ab" + "ab";
String s3 = "aba" + "b";
String s4 = s1 + s1;
String s5 = s1 + s1;
String s6 = s1 + "ab";
```



```
s4.equals(s2) //true
s4 == s2 //false
s4 == s5 //false
s2 == s3 //true
s2 == s6 //false
s4.intern() == s2 //true
s4.intern() == s5.intern() //true
```

ומה עם הבנאי?

- ניתן ליצור מחרוזות בעזרת קריאה לבנאי
- String s1 = new String("Hello");
- גורם תמיד להקצאה של זכרון
- יצירת מחרוזת ליטרלית
- String s1 = "Hello";
- נוצרת מחרוזת חדשה רק אם היא עדין לא במאגר

מנשקים

- מנשק (interface) הוא מבנה תחבירי ב Java המאפשר לחסוך בקוד לקוח
- קוד אשר משתמש במנשק יוכל בזמן ריצה לעבוד עם מגוון מחלקות המממשות את המנשק הזה (ללא צורך בשכפול הקוד עבור כל מחלקה)
- דוגמא: גנן מוזיקה אשר מותאם לעבוד עם קובצי מוזיקה (mp3) ועם קובצי וידאו (mp4)

Playing Mp3

```
public class MP3Song {
    public void play() {
        // audio codec calculations,
        // play the song...
    }
    // does complicated stuff
    // related to MP3 format...
}

public class Player {
    private boolean repeat;
    private boolean shuffle;

    public void playSongs(MP3Song[] songs) {
        do {
            if (shuffle)
                Collections.shuffle(Arrays.asList(songs));

            for (MP3Song song : songs)
                song.play();

        } while (repeat);
    }
}
```

Playing VideoClips

```
public class VideoClip {
    public void play() {
        // video codec calculations,
        // play the clip ...
    }
    // does complicated stuff
    // related to MP4 format ...
}

public class Player {
    // same as before...

    public void playVideos(VideoClip[] clips) {
        do {
            if (shuffle)
                Collections.shuffle(Arrays.asList(clips));

            for (VideoClip videoClip : clips)
                videoClip.play();

        } while (repeat);
    }
}
```

שכפול קוד

```
public void playSongs(MP3Song[] songs) {
    do {
        if (shuffle)
            Collections.shuffle(Arrays.asList(songs));

        for (MP3Song song : songs)
            song.play();

    } while (repeat);
}

public void playVideos(VideoClip[] clips) {
    do {
        if (shuffle)
            Collections.shuffle(Arrays.asList(clips));

        for (VideoClip videoClip : clips)
            videoClip.play();

    } while (repeat);
}
```

למרות ששני השרותים נקראים play() אלו פונקציות שונות!

נרצה למזג את שני קטעי הקוד

שימוש בממשק

```
public void play (Playable[] items) {
    do {
        if (shuffle)
            Collections.shuffle(Arrays.asList(items));

        for (Playable item : items)
            item.play();
    } while (repeat);
}
```

```
public interface Playable {
    public void play();
}
```

מימוש הממשק ע"י הספקים

```
public class VideoClip implements Playable {

    @Override
    public void play() {
        // render video, play the clip on screen...
    }

    // does complicated stuff related to video formats...
}
```

```
public class MP3Song implements Playable {

    @Override
    public void play(){
        // audio codec calculations, play the song...
    }

    // does complicated stuff related to MP3 format...
}
```

מערכים פולימורפים

```
Playable[] playables = new Playable[3];

playables[0] = new MP3Song();
playables[1] = new VideoClip();
playables[2] = new MP4Song(); // new Playable class

Player player = new Player();
// init player...
player.play(playables);
```

```
public void play (Playable [] items) {
    do {
        if (shuffle)
            Collections.shuffle(Arrays.asList(items));

        for (Playable item : items)
            item.play();
    } while (repeat);
}
```

עבור כל איבר במערך
יקרא ה `play()` המתאים

פעולות על סיביות

■ אופרטורים לביצוע פעולות על ביטים
■ רק על טיפוסים איטגרליים (`int`, `short`, `byte`, `char`)

<code>~</code>	Unary bitwise complement
<code><<</code>	Signed left shift
<code>>></code>	Signed right shift
<code>>>></code>	Unsigned right shift
<code>&</code>	Bitwise AND
<code>^</code>	Bitwise XOR
<code> </code>	Bitwise OR

28

פעולות על סיביות - דוגמאות

■ 32 ביטים `int`

ייצוג בינארי

```
3      0000000000000000000000000000000011
-3     11111111111111111111111111111100
-3     11111111111111111111111111111101
3 << 2 00000000000000000000000000000000100
-3 >> 1 1111111111111111111111111111110
-3 >>> 1 0111111111111111111111111111110
```

■ מה נקבל מ `3 & i`?
■ שני הביטים הימניים של `i`
■ ומה נקבל מ `0xF0 & i`?

29