

# תוכנה 1

תרגול 7: מנשקים, דיאגרמות, ועוד \*

# == 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)

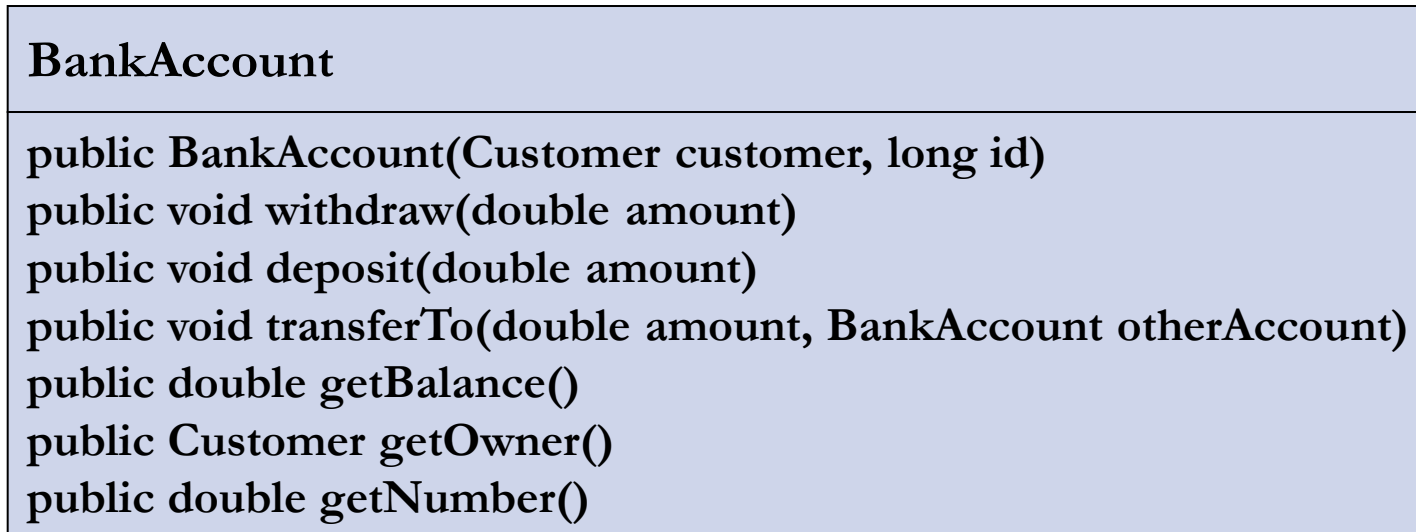
ז'אכרנות

# 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

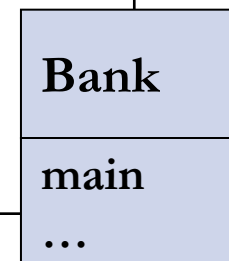
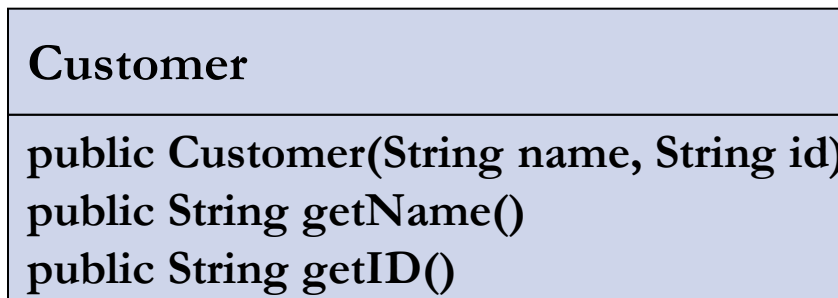


# Class Diagram



קשר לוגי של שייכות (למשל,  
ב- BankAccount יש שדה  
Customer), לא בלעדית

**Aggregation  
(has-a)**



**Association**

קשר כללי בין  
מחלקות, למשל,  
אחת משתמשת  
בשנייה באחת  
המתודות

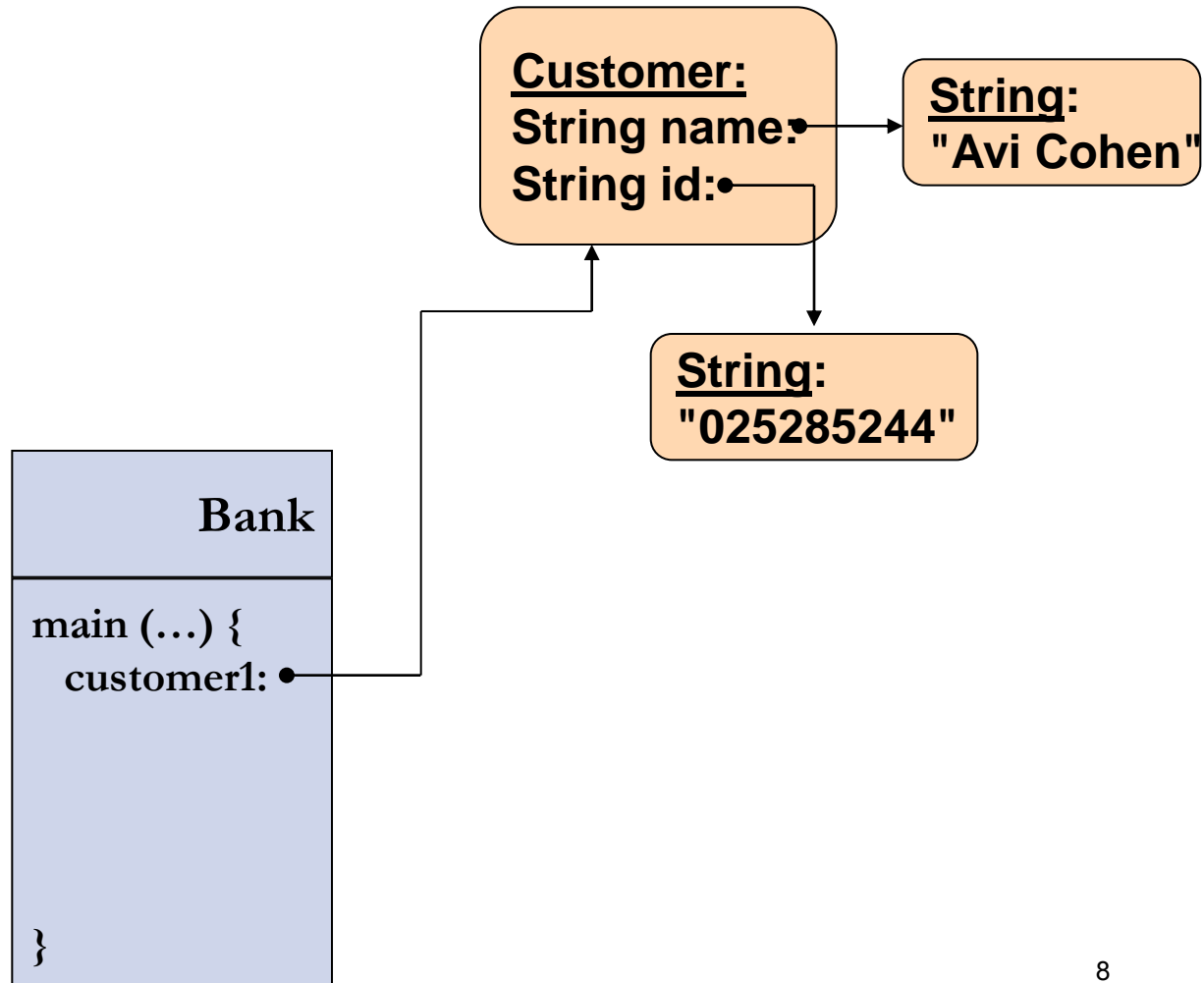
# המחלקה Customer

```
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

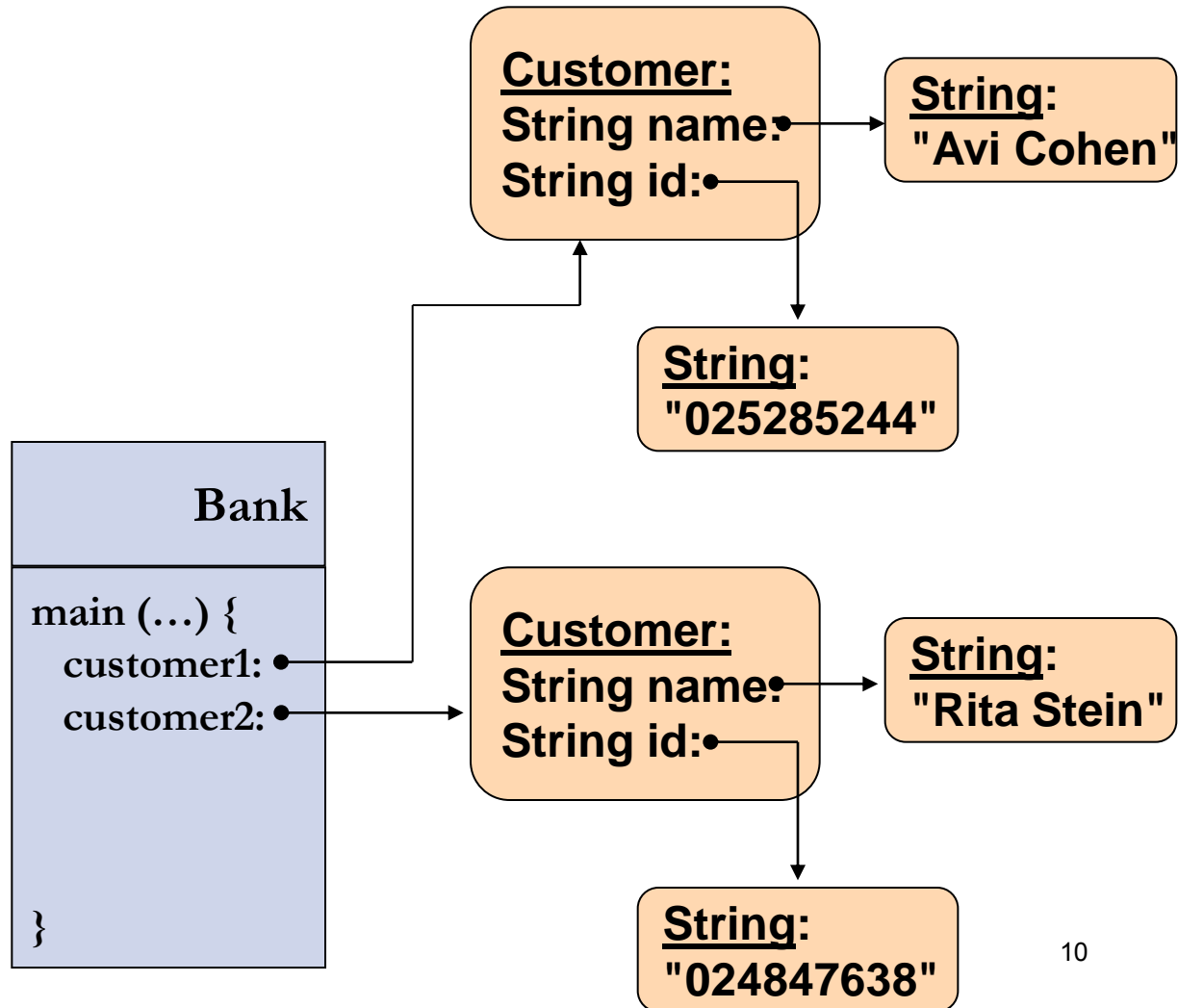




# Toy Bank Program

```
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());  
    }  
}
```

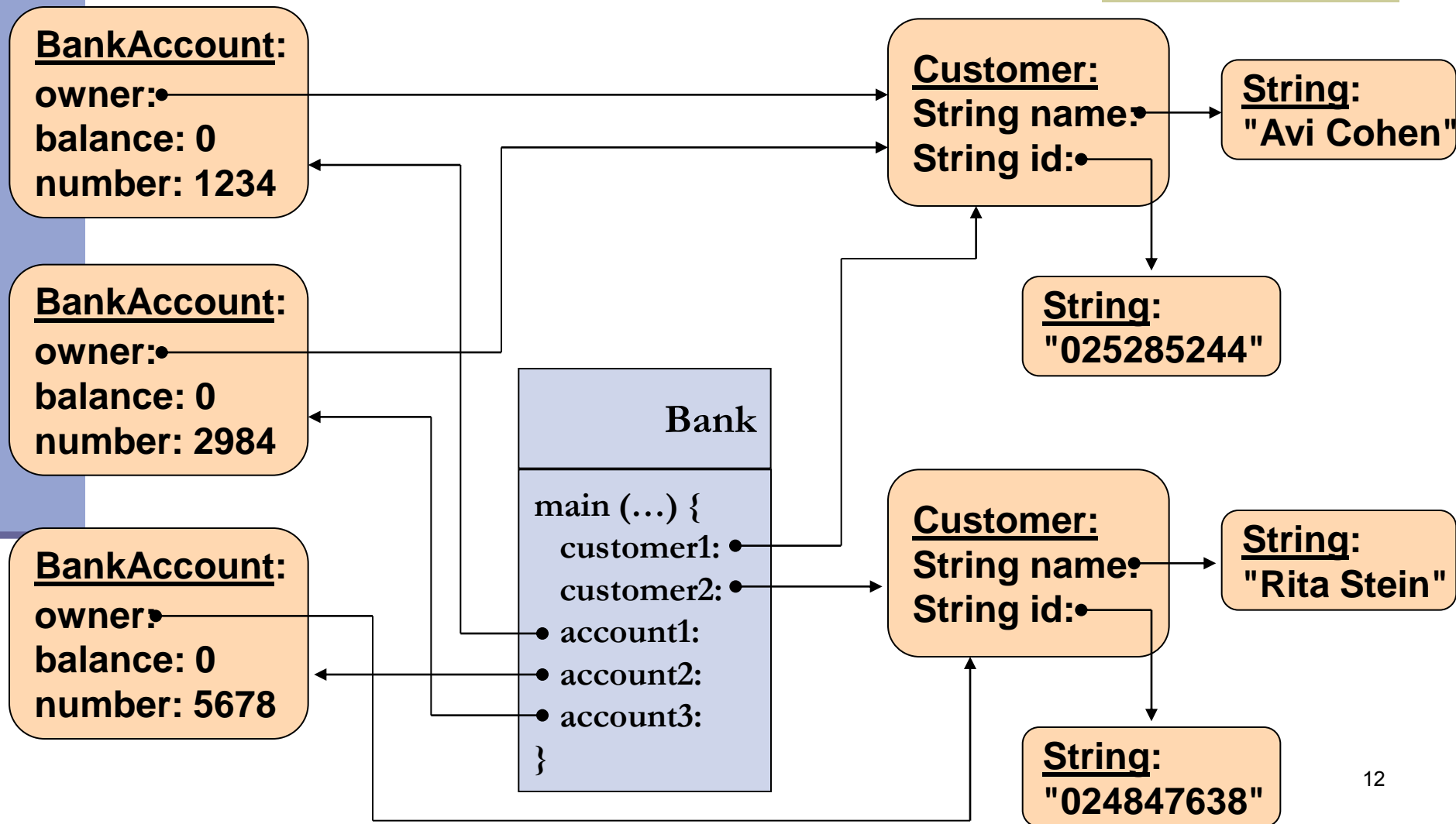
# Object Diagram



# 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());  
    }  
}
```

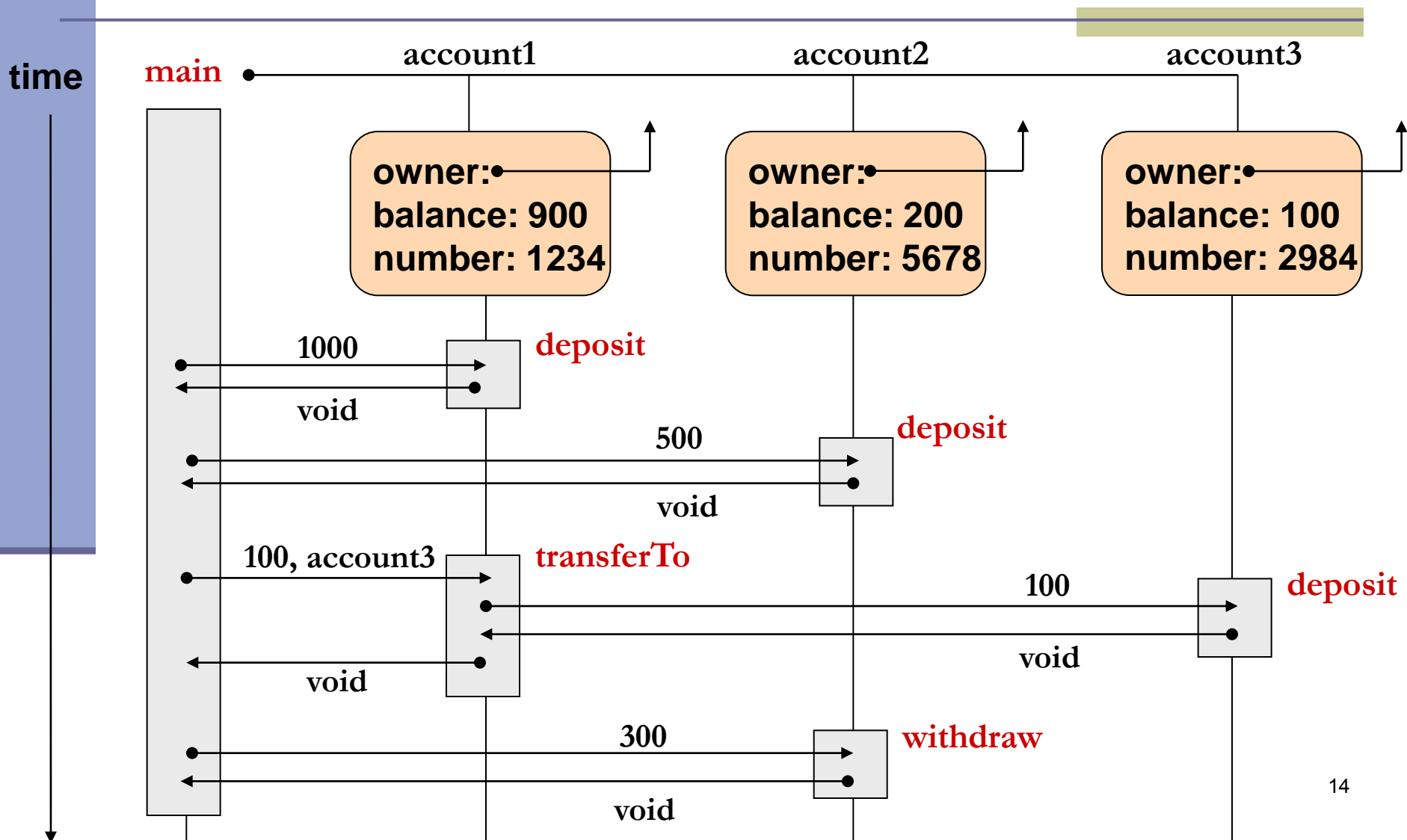
# Object Diagram



# 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());  
    }  
}
```

# Message Sequence Chart



# 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

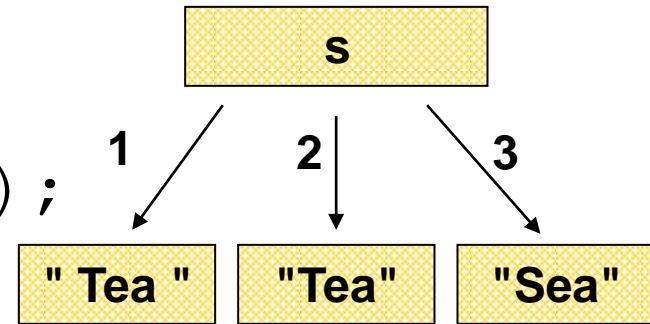
הפנמה של מחרזות



# String Immutability

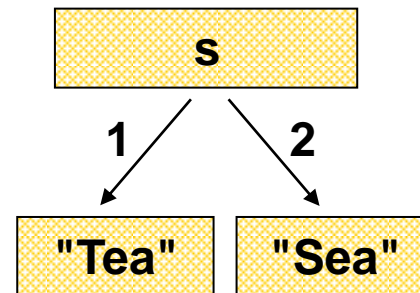
## מחרוזת הן מקובעות

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



## אפשר לשנות את ערכו של מצביע למחרוזת

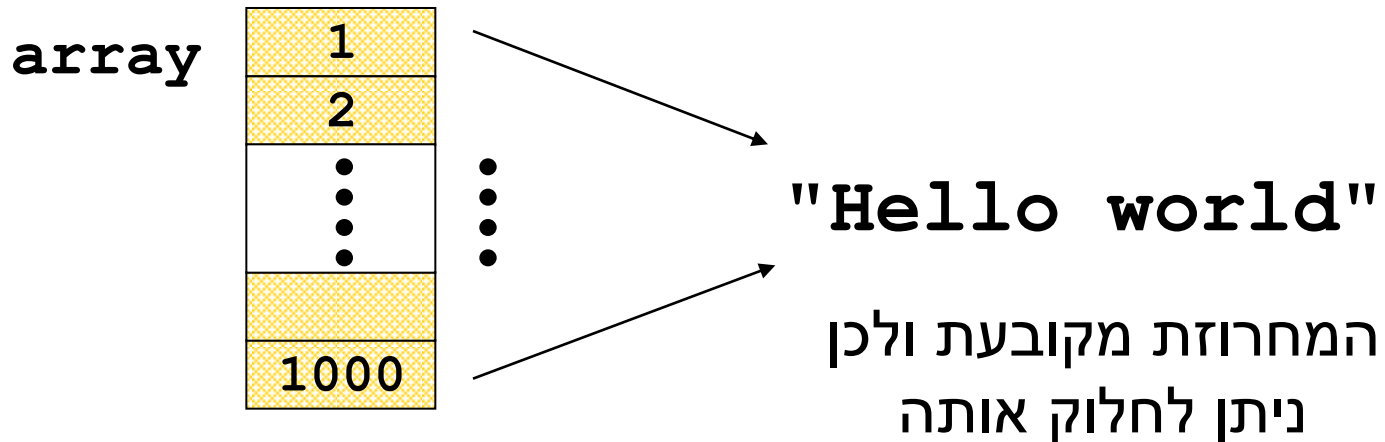
```
String s = "Tea";  
s = "Sea";
```



# String Interning

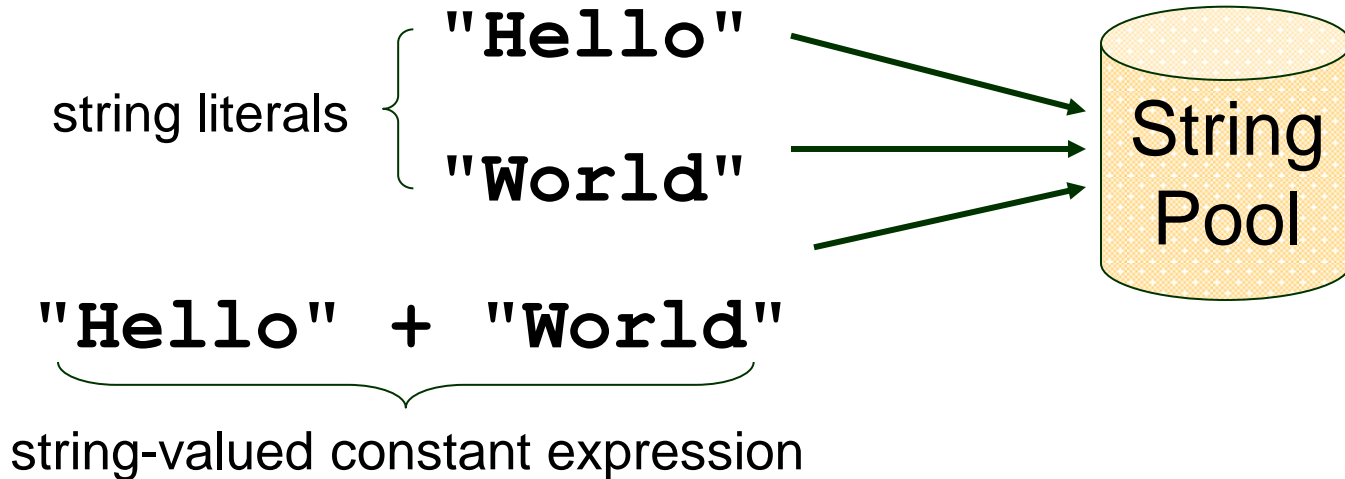
מונע שכפול של מחרוזות זהות בזיכרון ■

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



# אגירת מחרוזות

- כל המחרוזות הליטרליות, וביטויים קבועים מטיפוס מחרוזת נשמרים במאגר



# הפנמת מחרוזות

■ המחלקה String מחזיקה מאגר סטטי פרטי של מחרוזות.

■ המחוזות מוכנסות למאגר דרך המתודה intern

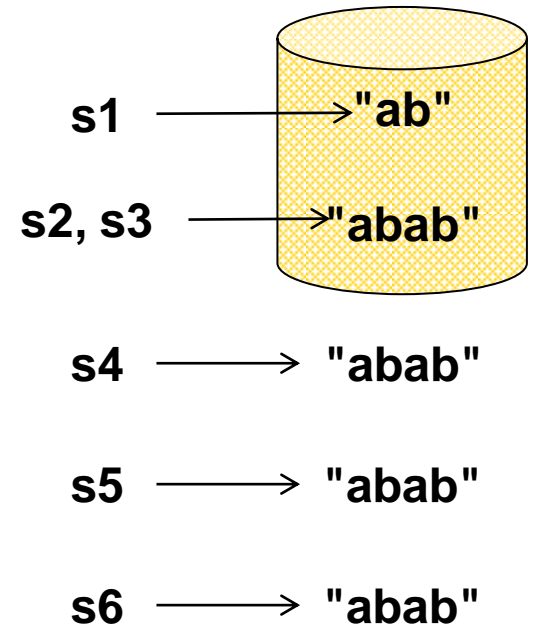
■ `myString.intern()` :

```
If  $\exists s \in pool : myString.equals(s)$   
    return s;  
Else  
    add myString to the pool  
    return myString;
```

# הפנמת מחרוזות - 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

*נדפד*

# 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: נגן 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);  
}
```

למרות ששני השרותים נקראים `play()`  
אלו פונקציות שונות!

```
public void playVideos(VideoClip[] clips) {  
    do {  
        if (shuffle)  
            Collections.shuffle(Arrays.asList(clips));  
  
        for (VideoClip videoClip : clips)  
            videoClip.play();  
  
    } while (repeat);  
}
```

נרצה למזג את שני קטעי הקוד

# שימוש במנשק

```
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 המנשק 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

שירות מופע ומחלקה

# Instance vs. Class (static) Fields

## Instance fields

למה? ■

■ ייצוג פנימי של המופע

מתי? ■

■ מאותחלים עם יצירת האובייקט

כמה? ■

■ אחד לכל מופע

מאיפה? ■

■ נגישים אך ורק ממתודות מופע!  
(למה?)

## Class (static) fields

למה? ■

■ קבועים

■ ערכים המשותפים לכל מופעי המחלקה

מתי? ■

■ מאותחלים לפי הסדר עם טעינת המחלקה

כמה? ■

■ יש רק 1 בכל התוכנית! (0 לפני טעינת המחלקה)

מאיפה? ■


■ נגישים ממתודות סטטיות ומתודות מופע



# דוגמא

```
public class BankAccount {
    public static final String BANK_NAME = "BNP"; //static constant
    private static int lastAccountId = 0; //static field
    private int id;

    public BankAccount() {
        id = ++lastAccountId; // unique ID for every account
    }

    /* static method */
    public static void main(String[] args) {
        System.out.println(lastAccountId);
         System.out.println(id);
        BankAccount account = new BankAccount();
        System.out.println(account.id);
    }

    /* instance method */
    public void printStuff() {
        System.out.println(lastAccountId);
        System.out.println(id);
    }
}
```