

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)

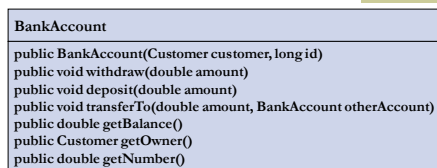
דיאגרמות

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.

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;
}
```

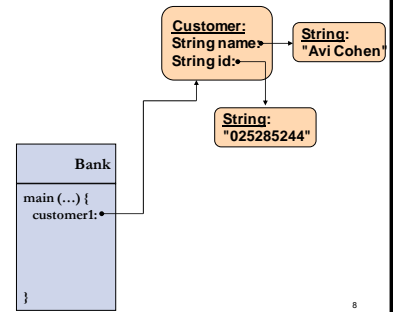
Toy Bank Program

```

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());
    }
}
    
```

7

Object Diagram



8

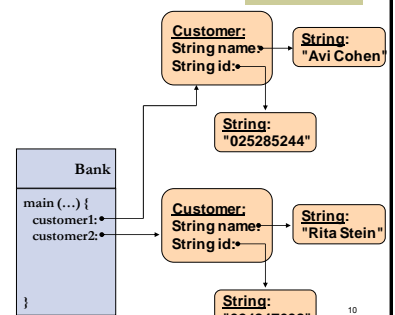
Toy Bank Program

```

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());
    }
}
    
```

9

Object Diagram



10

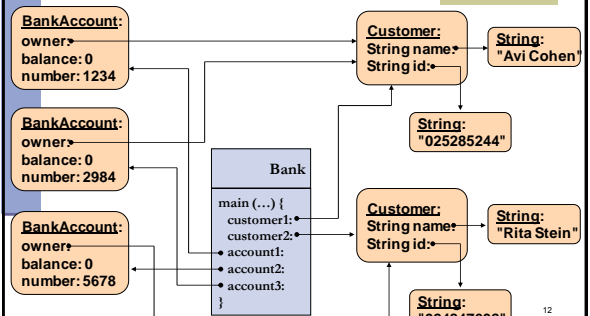
Toy Bank Program

```

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());
    }
}
    
```

11

Object Diagram



12

Message Sequence Chart

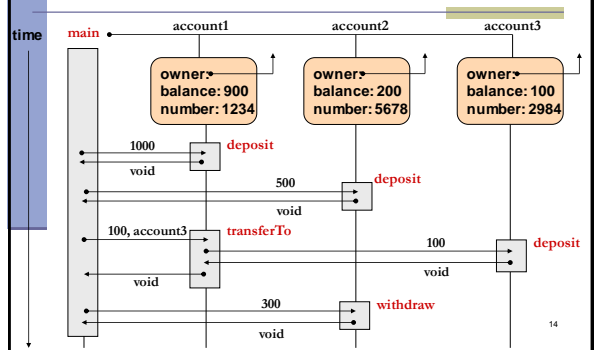
```
public class Bank {
    public static void main(String[] args) {
        Customer customer1 = new Customer("Avi Cohen", "025285244");
        Customer customer2 = new Customer("Rita Stein", "024847638");
        BankAccount account1 = new BankAccount(customer1, 1234);
        BankAccount account2 = 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());
    }
}
```

13

Message Sequence Chart



14

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

15

הפנאה של מחזורות

16

String Immutability

מחזורות הן מקובעות

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

אפשר לשנות את ערכו של מצביע למחזורות

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

String Interning

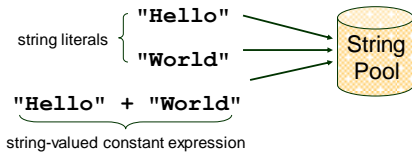
מונע שכפול של מחזורות זהות בזיכרון

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

המחזורות מקובעות ולכן ניתן לחלוק אותה

אגירת מחרזות

- כל המחרזות הליטרליות, וביטויים קבועים מטיפוס מחרזות נשמרים במאגר



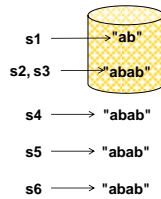
הפנמת מחרזות

- המחלקה String מחזיקה מאגר סטטי פרטי של מחרזות.
- המחרזות מוכנסות למאגר דרך המתודה intern
- myString.intern() :

```
if  $\exists s \in \text{pool} : \text{myString.equals}(s)$ 
    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 מחרזות בעזרת קריאה לבנאי
- String s1 = new String("Hello");
- גורם תמיד להקצאה של זכרון
- יצירת מחרזות ליטרלית
- String s1 = "Hello";
- נוצרת מחרזות חדשה רק אם היא עדיין 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: נגן מוזיקה אשר מותאם לעבוד עם קובצי מוזיקה (mp3) ועם קובצי וידאו (mp4)

מנשקים

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 למדג 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...

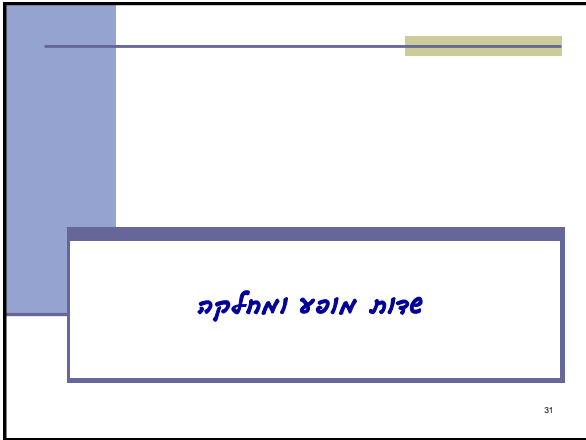
player.play(playables);

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



Instance vs. Class (static) Fields

Instance fields	Class (static) fields
<ul style="list-style-type: none"> ■ למה? ■ ייצוג פנימי של המופע 	<ul style="list-style-type: none"> ■ למה? ■ קבועים ■ ערכים המשותפים לכל מופעי המחלקה
<ul style="list-style-type: none"> ■ מתי? ■ מאותחלים עם יצירת האובייקט 	<ul style="list-style-type: none"> ■ מתי? ■ מאותחלים לפי הסדר עם טעינת המחלקה
<ul style="list-style-type: none"> ■ כמה? ■ אחד לכל מופע 	<ul style="list-style-type: none"> ■ כמה? ■ יש רק 1 בכל התוכנית! (0 לפני טעינת המחלקה)
<ul style="list-style-type: none"> ■ מאיפה? ■ נגישים אך ורק ממתודות מופע! (למה?) 	<ul style="list-style-type: none"> ■ מאיפה? ■ נגישים ממתודות סטטיות ומתודות מופע

32

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);
    }
}
    
```

33