תוכנה 1

סמסטר א' תשס"ט

תרגול מס' 6

מנשקים, דיאגרמות וביטים*

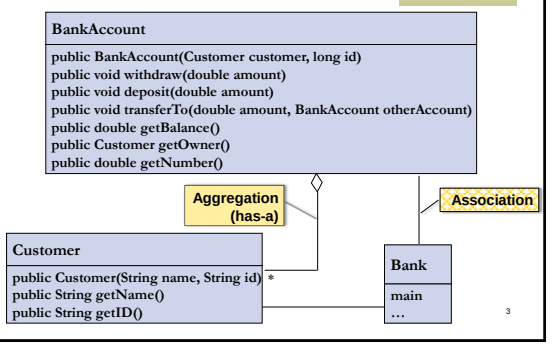
רובי בוים ומתי שמרת

* לא בהכרח בסדר הזה

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 האוביקטים
    - תיאור של תרחיש

Class Diagram



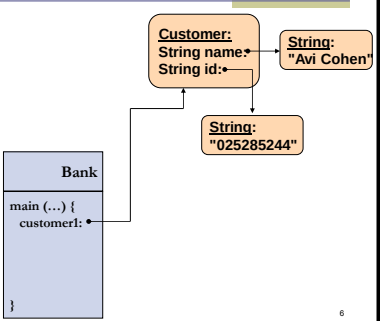
המחלקה Customer

```
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(customer2, 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

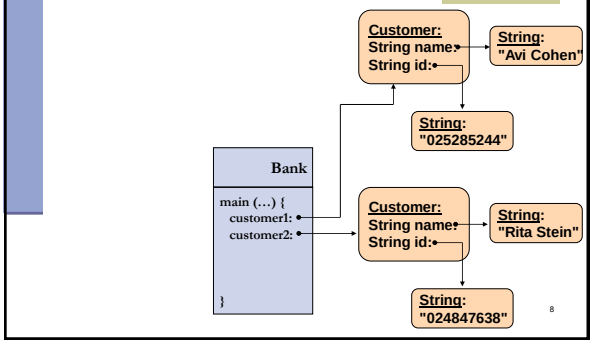


Toy Bank Program

```
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(customer2, 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



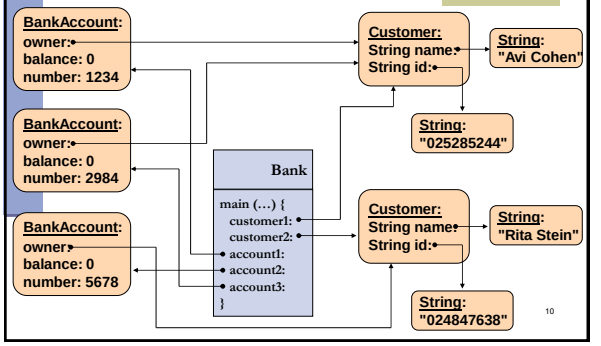
8

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(customer2, 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



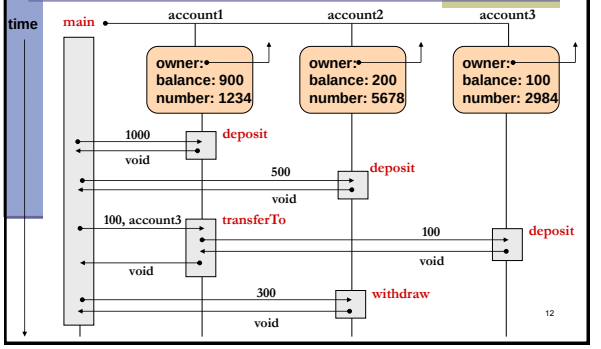
10

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(customer2, 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

Message Sequence Chart



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(customer2, 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

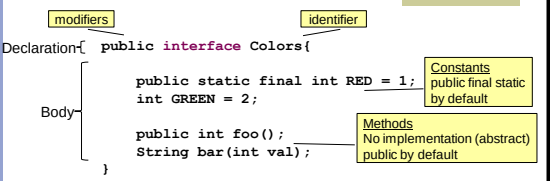
מנשקים - תזכורת

- כדי לתקשר בין הספק והלקוח עליהם להגדיר **מנשק** (interface, ממשק) ביניהם
- בתהליך פיתוח תוכנה תקין, כתיבת המנשק תעשה בתחילת תהליך הפיתוח
- כל מודול מגדיר מהם השירותים שלהם הוא זקוק ממודולים אחרים ע"י ניסוח מנשק רצוי
- מנשק זה מהווה בסיס לכתיבת הקוד הן בצד הספק, שיממש את הפונקציות הדרושות והן בצד הלקוח, שמשתמש בפונקציות (קורא להן) ללא תלות במימוש שלהן

מנשקים interface

- 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)

תחביר



- מנשק מגדיר טיפוס הפניה חדש (reference type)
- בזמן ריצה האובייקט המוצבע הוא ממחלקה המממשת את המנשק

תחביר ספק/לקוח

- ספק

```
public class MyClass implements Colors{
    public int foo() {...}
    public String bar(int val) {...}

    public void moreFunc() {...}

}
```

הספק מחוייב לממש את כל הפונקציות שהוגדרו במנשק. הוא יכול להגדיר מתודות נוספות.

- לקוח

```
public static void main(String[] args) {
   MyClass cls = new MyClass();
   Colors col = cls;
}
```

הלקוח יכול לפנות לאובייקטים בעזרת טיפוס המחלקה או בעזרת טיפוס המנשק.

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 (Player)

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

        for (Playable item : items)
          item.play();

    } while (repeat);

}
```

הגדרת המנשק

- שני קטעי הקוד עושים שימוש ב "משהו שאפשר לנגן אותו"
- נגדיר מנשק Playable
- כל מחלקה שממממשת טיפוס שאפשר לנגן אותו תממש את המנשק

דומה להגדרת מחלקה

מגדיר מתודה יחידה

```
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(playabl
```

```
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 איטנגרליים (int, short, byte, char)

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

פעולות על סיביות - דוגמאות

- 32 int ביטים

```
                ייצוג בינארי
3       00000000000000000000000000000011
~3      11111111111111111111111111111100
-3      11111111111111111111111111111101
3 << 2  00000000000000000000000000001100
-3 >> 1 11111111111111111111111111111110
-3 >>> 1 01111111111111111111111111111110
```

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

שימוש

- נתחיל במספר (או מספרים) שעבור אנו יודעים את הייצוג הבינארי
  - `00000000000000000000000000000001` – 1
  - `00000000000000000000000000001111` – 15
- בעזרת הפעולות שתוארו נוכל להגיע לכל סידור ביטים רצוי