

תוכנה 1

תרגול 7: ממשקים, פולימורפיזם ועוד

ממשקים

ממשקים

- ממשק (interface) הוא מבנה תחבירי ב-Java המאפשר לחסוך בקוד לקוח.
- ממשק מכיל כותרות של מתודות (חתימות) ללא המימוש שלהן.
- קוד אשר משתמש בממשק יוכל בזמן ריצה לעבוד עם מגוון מחלקות המממשות את הממשק הזה (ללא צורך בשכפול הקוד עבור כל מחלקה).

הגדרת ממשק

```
public interface InterfaceName {  
    public String someMethod();  
    public void anotherMethod(int param);  
}  
public class Concrete implements InterfaceName {  
    --  
    @Override  
    public String someMethod() {...}  
    @Override  
    public void anotherMethod(int param) {...}  
}
```

שם הממשק

מחלקה המממשת את הממשק

דוגמה 1: Shape - ממשק המייצג צורה

- נגדיר ממשק בשם **Shape** המייצג צורה גיאומטרית.
- הממשק Shape מחייב את כל המחלקות שמממשות אותו, לכלול מימוש עבור 2 מתודות:
 - `getArea()` – מחשבת את שטח הצורה
 - `getDetails()` – מחזירה מחרוזת המייצגת את הצורה.

```
public interface Shape {  
    public float getArea();  
    public String getDetails();  
}
```

המחלקה Square

```
public class Square implements Shape {  
    float side;  
    public Square(float side) {  
        this.side=side;  
    }  
    public float getArea() {  
        return (side*side);  
    }  
    public String getDetails(){  
        return "Square: side=" + this.side;  
    }  
}
```

המחלקה מצהירה שהיא מממשת את הממשק

מימוש של מתודות הממשק

המחלקה Circle

```
public class Circle implements Shape {
    float radius;

    public Circle(float radius) { //Constructor
        this.radius=radius;
    }

    public float getArea() { //Implementing Shape.getArea()
        return (float) (radius*radius*Math.PI);
    }

    public String getDetails() { //Implementing Shape.getDetails()
        return "Circle: radius=" + this.radius;
    }

    public float getRadius() { //Circle specific method
        return this.radius;
    }
}
```

7

טיפוס הפניה מסוג Shape

טיפוס הפניה מסוג Shape יכול להצביע אל כל אובייקט המממש את הממשק Shape.

```
Shape shape1 = new Square(100);
Shape shape2 = new Circle(50);
```

ניתן לקרוא באמצעותו רק למתודות הכלולות בהגדרת הממשק. לדוג': shape1.getArea().
כדי לקרוא למתודה הספציפית ל-Circle, יש לבצע הצרה באמצעות casting:

```
Circle circle = (Circle) shape2; // Down-casting
System.out.println( circle.getRadius() );
```

8

גישה אחידה לאובייקטים ע"י שימוש בממשק Shape

השימוש בממשקים מאפשר לנו לעבוד באופן אחיד עם אובייקטים של מחלקות שונות המממשות את הממשק. מערך פולימורפי יכיל אובייקטים מסוגים שונים.

```
Shape[] shapes = new Shape[]{
    new Square(10),
    new Circle(20),
    new Square(100)
};
```

```
for (Shape shape : shapes)
    System.out.println( shape.getDetails() + "\t area=" +
        shape.getArea() );
```

9

דוגמא 2: נגן מוזיקה

דוגמא:

נגן מוזיקה אשר מותאם לעבוד עם קבצי מוזיקה (mp3) ועם קבצי וידאו

10

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 MP3 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

עוד על ממשקים

- מחלקה יכולה לממש יותר מממשק אחד בג'אווה (תחליף לירושה מרובה).
- ממשק יכול להכיל מתודות וגם קבועים אך לא שדות.
- ממשק הוא טיפוס אבסטרקטי לחלוטין (ללא מימוש כלל). לא ניתן ליצור מופע של ממשק בעזרת הפקודה .new

דיאגרמות

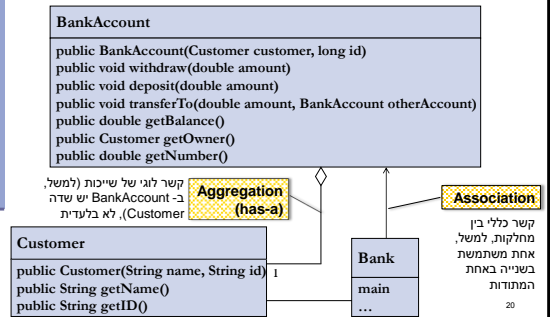
המערכת הבנקאית

- 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



19

Class Diagram



20

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;
}
```

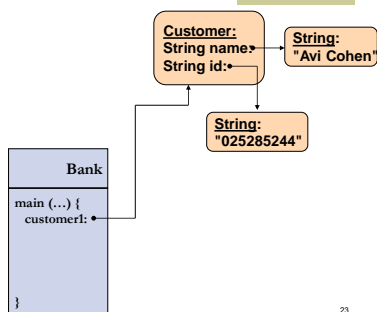
21

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());
    }
}
```

22

Object Diagram



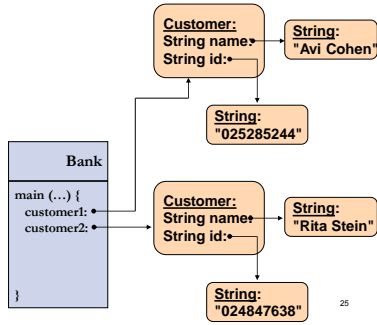
23

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());
    }
}
```

24

Object Diagram



Toy Bank Program

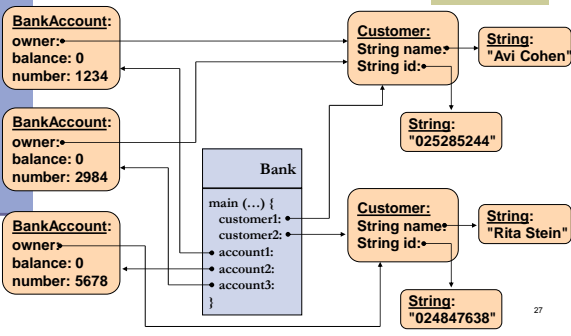
```

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

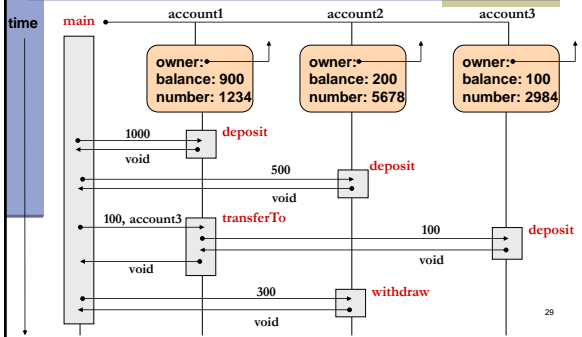
```

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