

# תוכנה 1

תרגול 6: מנשקים, פולימורפיזם ועוד

*נדפד*

# מנשקים - תזכורת

- מנשק (interface) הוא מבנה תחבירי ב-Java המאפשר לחסוך בקוד לקוח.
- מנשק מכיל כותרות של מתודות (חתימות). יתכן גם מימוש דיפולטיבי שלהן (החל מג'אווה 8).
- קוד אשר משתמש במנשק יוכל בזמן ריצה לעבוד עם מגוון מחלקות הממשות את המנשק הזה (ללא צורך בשכפול הקוד עבור כל מחלקה).

# הגדרת ממשק - תזכורת

```
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

# דוגמא 1: Shape - מנשק המייצג צורה

- נגדיר מנשק בשם **Shape** המייצג צורה גיאומטרית.
- המנשק 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 המנשק

מימוש של מתודות המנשק

# 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;  
    }  
}
```

# 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() );
```



# כללי השמה נוספים

■ ראינו השמה של עצם למשתנה מטיפוס ממשק (שהוא מממש).

```
Square mySquare = new Square(100);  
Shape myShape = mySquare;
```

■ אי אפשר לעשות השמה בכיוון ההפוך, או בין שני טיפוסים שמממשים את אותו ממשק

■ שוב, אפשר להיעזר ב-down-casing

✗ `Square mySquare2 = myShape;`

✗ `Circle myCircle2 = mySquare;`

✓ `Square mySquare2 = (Square) myShape;`

■ down-casting מאפשר "להתחכם" ולבצע השמה מוזרה

```
Circle myCircle2 = (Circle) myShape;
```

■ במקרה כזה, השגיאה תתגלה רק בזמן ריצה (כשיתברר ש-myShape אינו עיגול)

# גישה אחידה לאובייקטים ע"י שימוש במנשק Shape

- השימוש במנשקים מאפשר לנו לעבוד באופן אחיד עם אובייקטים של מחלקות שונות המממשות את המנשק.
- 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() );
```

# דוגמא 2: נגן מוזיקה

---

■ דוגמא:

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

# 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

# עוד על מנשקים

- לא ניתן ליצור מופע של מנשק בעזרת הפקודה `new`.
- מנשק יכול להכיל מתודות וגם קבועים אך לא שדות.
- מחלקה יכולה לממש יותר ממנשק אחד בג'אווה (תחליף לירושה מרובה).

```
public class Circle implements Shape, Drawable {...}
```

- מנשק יכול להרחיב מנשק אחר (ואז יכלול גם את המתודות המוגדרות במנשק זה).

```
public interface Shape extends Drawable {...}
```

פצולות על סיביות

# פעולות על סיביות

אופרטורים לביצוע פעולות על ביטים ■

רק על טיפוסים שלמים (int, short, byte, char) ■

3 00000000000000000000000000000011

~ Unary bitwise complement

<< Signed left shift

>> Signed right shift

>>> Unsigned right shift

& Bitwise AND

^ Bitwise XOR

| Bitwise OR

# פעולות על סיביות

■ אופרטורים לביצוע פעולות על ביטים

■ רק על טיפוסים שלמים (int, short, byte, char)

```
3      00000000000000000000000000000011
~3     11111111111111111111111111111100
```

- `int x = 3;`
- `int y = ~x;`

# פעולות על סיביות - דוגמאות

int 32 ביטים. ■

ייצוג בינארי

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

מה נקבל מ  $i \& 3$  ? ■

# פעולות על סיביות - דוגמאות

int 32 ביטים. ■

ייצוג בינארי

```
3      00000000000000000000000000000011
&
i      00000000000000000000000000000101
=
      00000000000000000000000000000001
```

מה נקבל מ  $3 \& i$ ? ■

שני הביטים הימניים של  $i$  ■

ומה נקבל מ  $0xF0 \& i$ ? ■<sup>23</sup>

פאנווה פו דזסט שײאָה  
(Stack Trace)



# Interpreting a Stack Trace of an Exception

כשנתקלים בחריגה במהלך ריצת התוכנית, ניתן להשתמש במידע שניתן לנו כדי לזהות את **סוג החריגה ואת המיקום בתוכנית שבו היא ארעה**.

## Console:

```
Exception in thread "main" java.lang.NullPointerException at
  com.example.myproject.Book.getTitle(Book.java:16) at
  com.example.myproject.Author.getBookTitles(Author.java:25) at
  com.example.myproject.Bootstrap.main(Bootstrap.java:14)
```

## Book.java:

```
public String getTitle() {
    System.out.println(title.toString()); <-- line 16
    return title;
}
```

# Interpreting a Stack Trace of an Exception

דוגמא נוספת: ■

```
Exception in thread "main" java.lang.OutOfMemoryError: Java heap space
at java.util.Arrays.copyOf(Unknown Source)
at java.lang.AbstractStringBuilder.expandCapacity(Unknown Source)
at java.lang.AbstractStringBuilder.ensureCapacityInternal(Unknown Source)
at java.lang.AbstractStringBuilder.append(Unknown Source)
at java.lang.StringBuilder.append(Unknown Source)
at SmallTestMultiCollections.testOrder(SmallTestMultiCollections.java:56)
at SmallTestMultiCollections.main(SmallTestMultiCollections.java:34)
```

הסוף...