תוכנה 1

תרגול 6: מנשקים, פולימורפיזם ועוד

* לא בהכרח בסדר הזה

מנשקים

מנשקים - תזכורת

- מנשק (interface) הוא מבנה תחבירי ב-Java המאפשר לחסוך בקוד לקוח.
- המנשק מכיל את **שירותי המופע הציבוריים** (public methods) שתספק המחלקה המבוקשת
  - חלק מהשירותים יהיה **נטולי מימוש** – abstract
  - חלק מהשירותים **יכילו מימוש** – default (החל מ java 8)
- שירותי מופע **ממומשים** בניראות **private**
- שירותים **סטטיים** עבורם **קיים מימוש** – בניראות public או private (בגרסאות אחרי java 8)
- **שדות** שהם **public static final**

- קוד אשר משתמש במנשק יוכל בזמן ריצה לעבוד עם מגוון מחלקות המממשות את המנשק הזה (ללא צורך בשכפול הקוד עבור כל מחלקה).

הגדרת מנשק - 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 המנשק

אבסטרקטיות

abstract

מחלקה המממשת את המנשק

המימוש

דוגמא 1: Shape - מנשק המייצג צורה

- נגדיר מנשק בשם **Shape** המייצג צורה גיאומטרית.
- המנשק Shape מחייב את כל המחלקות שממשות 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;
  }

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

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

- ניתן לקרוא באמצעותו רק למתודות הכלולות בהגדרת המנשק. לדוג': shape1.getArea()
- כדי לקרוא למתודה הספציפית ל-Circle?

```
Circle circle = shape2;

System.out.println( circle.getRadius() );
```

טיפוס הפניה מסוג Shape

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

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

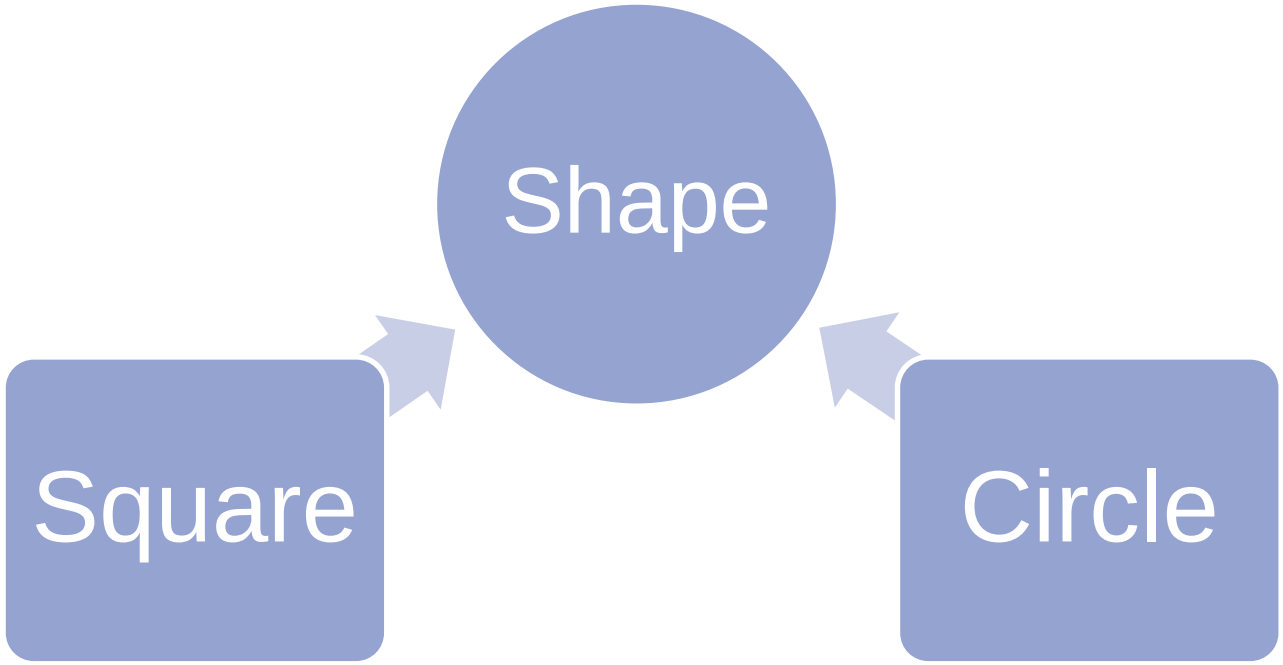
- ניתן לקרוא באמצעותו רק למתודות הכלולות בהגדרת המנשק. לדוג': shape1.getArea()
- כדי לקרוא למתודה הספציפית ל-Circle, יש לבצע הצרה באמצעות casting:

```
Circle circle = (Circle) shape2;  // Down-casting

System.out.println( circle.getRadius() );
```

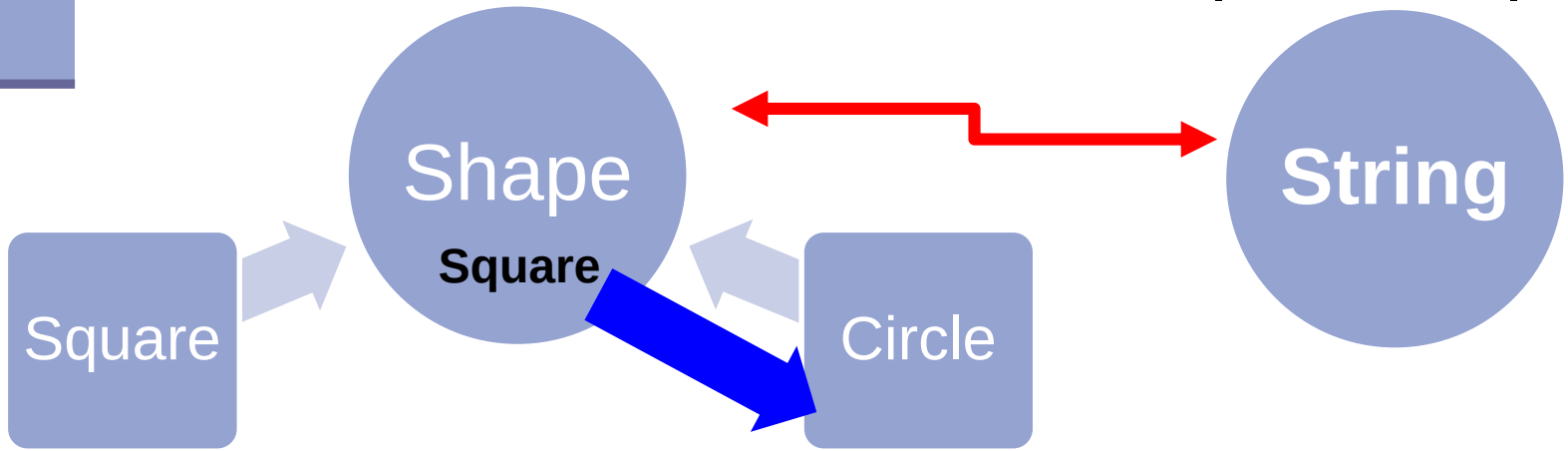
Cast

- יש שני סוגים של cast:
  - **Upcast**: אפשר לעשות במרומז (implicit).
  - **Downcast**: חייבים לעשות בבירור (explicit).



Cast

- יש שני סוגים של cast:
  - **Upcast**: אפשר לעשות במרומז (implicit).
  - **Downcast**: חייבים לעשות בבירור (explicit).
- בשימוש שגוי יתכנו שגיאות הידור או שגיאות זמן ריצה
  - הידור: לא מתחת ולא מעל ההיררכיה.
  - זמן ריצה: יתכן שתחתיו, אבל בפועל לא.



Cast

- יש שני סוגים של cast:
  - **Upcast**: אפשר לעשות במרומז (implicit).
  - **Downcast**: חייבים לעשות בבירור (explicit).
- בשימוש שגוי יתכנו שגיאות הידור או שגיאות זמן ריצה
  - הידור: לא מתחת ולא מעל ההיררכיה.
  - זמן ריצה: יתכן שתחתיו, אבל בפועל לא.
- Cast משנה יחס, **לא משנה את האובייקט עצמו**
- **בדרך כלל נרצה להימנע מcast**
  - קוד לא קריא, ו"מזמין" שגיאות
  - לצערנו לפעמים אי אפשר להימנע מהשימוש

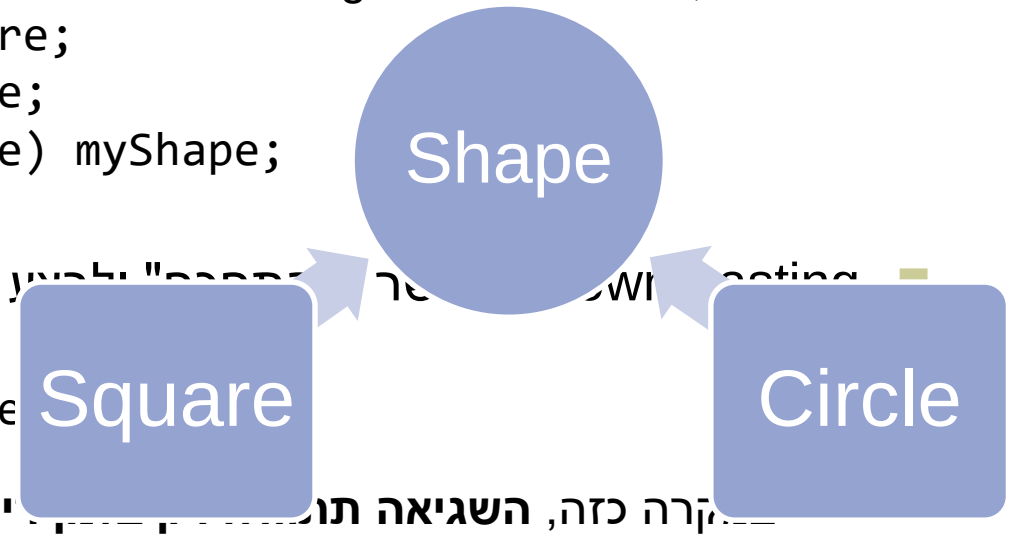
כללי השמה נוספים

- ראינו השמה של עצם למשתנה מטיפוס מנשק (שהוא מממש).

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

- אי אפשר לעשות השמה בכיוון ההפוך, או בין שני טיפוסים שממשים את אותו מנשק
  - שוב, אפשר להיעזר ב-down-casing

```
Circle myCircle2 = mySquare;
Square mySquare2 = myShape;
Square mySquare2 = (Square) myShape;
```



- [illegible] השמה מוזרה

```
Circle myCircle2 = (Circle
```

[illegible] **השגיאה תו[illegible] יצה** (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

Player MP3

```
public void playSongs(MP3Song[] songs) {
      do {
            if (shuffle)
              Collections.shuffle(Arrays.asList(songs));

            for (MP3Song song : songs)
              song.play();

      } while (repeat);

}
```

Player video

```
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 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(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

עוד על מנשקים

- לא ניתן ליצור מופע של מנשק בעזרת הפקודה new.
- מנשק יכול להכיל מתודות וגם קבועים
- 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 {...}
```

פעולות על סיביות

פעולות על סיביות

- אופרטורים לביצוע פעולות על ביטים
  - **<u>רק על טיפוסים שלמים</u>** (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 - דוגמאות

- int 32 ביטים

	ייצוג בינארי
3	00000000000000000000000000000011
~3	11111111111111111111111111111100

- int x = 3;
- int y= ~x // Bitwise not

- System.out.println(Integer.toBinaryString(3));

פעולות על סיביות - דוגמאות

- 32 ביטים int

	ייצוג בינארי	
3	00000000000000000000000000000011	
~3	11111111111111111111111111111100	not
-3	11111111111111111111111111111101	
3 << 2	00000000000000000000000000001100	Shift left
-3 >> 1	11111111111111111111111111111110	Shift right
-3 >>> 1	01111111111111111111111111111110	

- מה נקבל מ `3 & i`?

פעולות על סיביות - דוגמאות

- int 32 ביטים

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

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

פענוח של הדפסת שגיאה
(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
```

The end