

## התוכנית להיום

- דוגמה מעשית לתרגול מנשקים



## מעבדת מטריצות

2

## תוכנה 1 – מנשקים

תרגול 6  
מתי שמרת, ליאור שפירא

1

## מטריצה כמחלקה

- נראה שניתן לייצג מטריצה כמחלקה פשוטה

```
public class Matrix {  
    int getWidth(){...}  
    int getHeight(){...}  
    double getElement(int col, int row){...}  
    void setElement(int col, int row, double val){...}  
    Matrix multiply(Matrix m){...}  
    Matrix multiply(double scalar){...}  
}
```

מה יכולות להיות החסרונות של ייצוג כזה?

4

## המטרה

- עלינו לפתח תוכנית לביצוע חישובים באלגברה לינארית
- לב התוכנית יהיה ייצוג של מטריצה

- כיצד נייצג מטריצה?

5

5
1
4
2

51	4
1	7

3

## מטריצה כמנשק

- רעיון: נגדיר מטריצה כמנשק, וניצור מספר מחלקות, אחת לכל סוג של מטריצה (הן יממשו את המנשק)

- איזה סוגי מטריצות?

- מטריצות מלאות
- מטריצת היחידה
- מטריצה אלכסונית
- מטריצה דלילה
- מטריצה משולשית עליונה (תחתונה)

6

## לא כל המטריצות נולדו שוות

- ניקח לדוגמה כפל מטריצות
- כפל מטריצות כללי:

$$O(n^3) \leftarrow (AB)_{i,j} = \sum_{r=1}^n A_{i,r} B_{r,j}$$

- אבל מה אם שתי המטריצות אלכסוניות?

$$O(n) \leftarrow (AB)_{i,i} = A_{i,i} B_{i,i}, (AB)_{i \neq j} = 0$$

5

## מימוש ממשק מטריצה

### ■ נתחיל במטריצה רגילה (מלאה)

```
public class OrdinaryMatrix implements Matrix {
    public OrdinaryMatrix(int width, int height) {
        data = new double[width][height];
        this.width = width;
        this.height = height;
    }
    public OrdinaryMatrix(double[][] data) {
        ...
    }
    public double getElementAt(int col, int row) {
        return data[col][row];
    }
    ...
    private double data[][];
    private int width, height;
}
```

בנאים

שירותים שהוצהרו בממשק

"צוג פנימי"

8

## מטריצה כממשק

```
public interface Matrix {
    int getWidth();
    int getHeight();
    double getElementAt(int col, int row);
    void setElementAt(int col, int row, double newVal);
    Matrix multiply(Matrix m);
    Matrix multiply(double scalar);
}
```

שאלות

פקודות

שאלות מפקיות

7

## מימוש ממשק מטריצה

- מה נרויח ע"י מימוש נפרד של מטריצה אלכסונית?
- זיכרון (נאחסן רק את האלכסון הראשי)
- זמן ריצה (נוכל להשתמש באלגוריתמים יעילים)
- כפל מטריצות, להפוך מטריצות, חישוב נורמה, ...

10

## מימוש ממשק מטריצה

- נממש כפל בסקלר

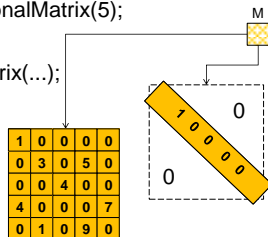
```
public Matrix multiply(double scalar) {
    OrdinaryMatrix result =
        new OrdinaryMatrix(width, height);
    for (int i = 0; i < width; i++) {
        for (int j = 0; j < height; j++) {
            result.data[i][j] = data[i][j] * scalar;
        }
    }
    return result;
}
```

הערך המוחזר יהיה תת-טיפוס של הממשק, כלומר מחלקה המממשת את הממשק.

9

## שימוש במטריצות

```
Matrix m = new DiagonalMatrix(5);
m.setElement(0,0,1);
m = new OrdinaryMatrix(...);
m.setElement(0,0,1);
```



12

## מימוש ממשק מטריצה

```
public class DiagonalMatrix implements Matrix {
    public DiagonalMatrix(int dsize) {...}
    public double getElement(int col, int row) {
        return (col == row ? data[col] : 0);
    }
    ...
    public DiagonalMatrix multiply(DiagonalMatrix other) {
        ...
    }
    private double diagonal[];
}
```

שירותים נוספים השייכים למחלקה זו בלבד.

"צוג פנימי"

11

## איזו פונקציה תתבצע?

```
public static void main(String[] args) {
    DiagonalMatrix d1 = new DiagonalMatrix(...);
    DiagonalMatrix d2 = new DiagonalMatrix(...);
    OrdinaryMatrix o = new OrdinaryMatrix(...);
    Matrix m = d2;

    printMatrix(d1.multiply(o));
    printMatrix(d1.multiply(d2));
    printMatrix(d1.multiply(m));
    printMatrix(o.multiply(d1));
    printMatrix(m.multiply(d1));
}
```

DiagonalMatrix:  
Matrix multiply(Matrix other);  
DiagonalMatrix multiply(DiagonalMatrix other);

OrdinaryMatrix:  
Matrix multiply(Matrix other);

14

## שימוש במטריצות

```
public class MatrixClient {
    public static void main(String[] args) {
        ...
    }

    private static void printMatrix(Matrix m) {
        for (int i = 0; i < m.getHeight(); i++) {
            for (int j = 0; j < m.getWidth(); j++) {
                System.out.format("%2f\t", m.getElementAt(i, j));
            }
            System.out.println();
        }
        System.out.println();
    }
}
```

מטפלת בכל סוגי המטריצות דרך הממשק

שימוש במתודות המוגדרות בממשק לאורך גישה לנתונים וביצוע פעולות.

13

## Arrays.sort

השירות sort דורש מימוש ממשק Comparable

```
public interface Comparable {
    int compareTo(Object o);
}
```

מחזיר ערך שלילי, 0 או חיובי אם האובייקט הנכתי קטן, שווה או גדול מהאובייקט הנספ.

נוסיף למטריצות שלנו את הממשק

```
public class DiagonalMatrix implements Matrix, Comparable {
    public int compareTo(Object o) {
        ...
    }
}
```

רשימה של ממשקים ממומשים מופרדים ב'

16

## מטריצה ברת השוואה

מה יקרה אם נפעיל את הקוד הבא?

```
Matrix[] arr = new Matrix[5];
for (int i=0; i<arr.length; i++) {
    ... // create new matrices and fill them with values
}

Arrays.sort(arr);
```

15



## בית חרושת למטריצות

- יש לנו כעת שני מימושים (לפחות) של מטריצה
- מי מחליט באיזה מימוש להשתמש? למי יש הידע הזה?
- הלקוח אשר מספק את המידע ומשתמש במחלקות?
- הספק אשר כתב את המחלקות ומכיר את פרטי המימוש?
- לעיתים שימושי להגדיר מחלקת Factory (בית חרושת) שתייצר מופעים של ממשק בהתאמה לצרכים

18

## מיון מטריצות

- מימוש הממשק Comparable מרמז שיש סדר טבעי לטיפוסים מסוג זה (הם ברי השוואה) ולכן ניתן למיין אותם
- במימוש compareTo אין אנו חייבים לממש השוואה לכל טיפוס אפשרי (נלמד על זאת יותר בהריגים)
- כל טיפוס בר-השוואה ניתן כעת למיין בעזרת אותו קטע קוד, בלי חשיבות למה מייצג הטיפוס

17

## סיכום

- מנשקים מנתקים את החוזה מהמימוש
- לעיתים נרצה מימושים שונים לאותו מנשק, במצבים שונים
- מנשקים יכולים לתפקד כ"שם תואר", מעידים על תכונה (serializable, iterable, comparable)

20



## בית חרושת למטריצות

- דוגמה לבית חרושת אפשרי:

```
public class MatrixFactory {
    private static boolean isDiagonal(double[][] data) {...}

    public static Matrix createMatrix(double[][] data) {
        Matrix m = isDiagonal(data) ?
            new DiagonalMatrix(data) :
            new OrdinaryMatrix(data);

        return m;
    }
}
```

19

הסוף

21