

# חלק 11

## לחשוב עצמים

# דוגמא : אלגוריתם הנפה

- ראינו בתחילת הסמסטר מימוש של אלגוריתם הנפה של ארתוסתנס בעזרת מערך בוליאני.
- בנאי שיוצר את המספרים הראשוניים
- שרות שמדפיס את המספרים הראשוניים.

```
public class ArraySieve {  
    private boolean [] isPrime;  
    private int max;
```

```
public ArraySieve (int n) {  
    max = n;  
    isPrime = new boolean[max];  
    for (int i = 2; i < n; i++)  
        isPrime[i] = true;  
    for (int i = 2; i < max; i++) {  
        if (isPrime[i])  
            for (int j = i+i; j < max; j+=i)  
                isPrime[j] = false;  
    }  
}
```

```
public void printPrimes() {  
    for (int i = 2; i < max; i++)  
        if (isPrime[i])  
            System.out.print(i + " ");  
    }  
}
```

# מחלקה לבדיקה

```
public class Test {  
    public static void main(String[] args) {  
        ArraySieve a = new ArraySieve(100);  
        a.printPrimes(); System.out.println();  
    }  
}
```

● יודפס:

2 3 5 7 11 13 17 19 23 29 31 37 41 43 47

53 59 61 67 71 73 79 83 89 97

# אלגוריתם הנפה במימוש מונחה עצמים

- מחלקה לייצוג נפה - עצם לכל נפה (במקום עצם אחד לכולם)
- מהם השרותים של הנפה?
- איך נאכסן את המספרים עצמם?
- אפשרות פשוטה - מערך כמו קודם.
- אבל יש פתרון פשוט יותר, שאינו דורש מבנה נתונים נוסף.
- זאת לא דוגמא אופיינית ליישום, אבל היא מדגימה חשיבה בעצמים.

## הנפה כעצם

- נוצר עם מספר  $n$  . מדפיס את  $n$  וממתין להודעות.
- כאשר מגיעה הודעה  $put(m)$
- אם  $m$  מתחלק ב  $n$  התעלם
- אם  $m$  לא מתחלק ב  $n$
- אם הנפה הבאה עדיין לא קיימת, יש ליצור נפה חדשה עם המספר  $n$  ולקשרה לשדה  $next$
- אם הנפה הבאה קיימת, להעביר אליה את ההודעה
- עצם הנפה צריך לזכור
- את המספר איתו נוצר  $first$
- את הנפה הבאה  $next$

# המחלקה Sieve

```
public class Sieve {  
    private int first;  
    private Sieve next;  
    public Sieve(int n) {  
        first = n;  
        System.out.print(n + " ");  
    }  
}
```



```
public void put(int m) {  
    if ((m % first) != 0) {  
        if (next == null)  
            next = new Sieve(m);  
        else next.put(m);  
    }  
}  
}
```

# איך הכל יופעל?

- ה main ייצור את הנפה הראשונה עם המספר 2, ויישלה אליה הודעות put(3), put(4), ..... put(Max)

```
public class Primes {  
    public static void main(String[] args) {  
        Sieve first = new Sieve(2);  
        for (int i = 3; i < 100; i++)  
            first.put(i);  
    }  
}
```

# דוגמא undo

- בתוכנות עריכה ויישומים אחרים יש צורך לאפשר למשתמש לבטל את הפקודה האחרונה שבוצעה.
- דרישות מהיישום:
  - שלא יהיה צורך לתכנן מחדש בכל פעם שנוסיף למערכת פקודה חדשה. מזה נובע שלא נוכל להתייחס ל undo ו redo כאל פקודות רגילות
  - שצריכת הזיכרון תהיה סבירה. מזה נובע שלא נוכל לשמור את כל המצב לפני ביצוע כל פקודה
  - שהפתרון יאפשר ביצוע undo למספר רמות כלשהו.

# פתרון

- פקודה תהיה עצם.
- נגדיר מנשק Command

```
interface Command {  
    abstract public void execute();  
    abstract public void undo();  
}
```

- כל סוג של פקודה יהיה מחלקה שממשת את Command עם שדות להחזיק את המידע הדרוש כדי לבטל את הביצוע.
- כדי להפעיל פקודה, ניצור עצם, נקרא ל `execute()` ונשמור את העצם.

# פקודה לדוגמא

- עבור מחיקת שורה יש לזכור את מספר השורה ואת תוכנה. כאן מספר השורה נקבע בזמן יצירת העצם, ותוכן השורה נשמר בזמן ביצוע המחיקה.

```
public class LineDeletion
    implements Command {
    private int deletedLineIndex;
    private String deletedLine;
    public LineDeletion(int n) {
        deletedLineIndex = n;
    }
}
```

# המשך מחיקת שורה

- מימוש השרותים:

```
public void execute() {  
    deletedLine = line at position deletedLineIndex  
    Delete the line in position deletedLineIndex  
}
```

```
public void undo () {  
    Put back deletedLine at position deletedLineIndex  
}
```

```
}
```

- באופן דומה נממש פקודות אחרות

# קוד הלקוח

- במערכת עם GUI קוד הלקוח יופיע קרוב לוודאי בתוך listener מתאים, שיופעל כאשר ארוע מתאים יתבצע.
- למשל, האירוע שיגרום לביצוע הפקודה `LineDeletion` הוא בחירה של הפריט המתאים `delete line` מתפריט.
- קוד הלקוח ייראה כך

```
Command com = new LineDeletion(cursor);  
com.execute();  
lastCommand = com;
```

- כאשר `cursor` הוא משתנה שעוקב אחר השורה הנוכחית בטקסט, ו `lastCommand` אמור לזכור את הפקודה האחרונה שבוצעה, כדי לאפשר ביצוע `undo`

# קוד הלקוח ל undo

- קוד הלקוח ל undo (שיימצא בתוך listener מתאים) יראה בערך כך:

```
if (lastCommand == null)
    write a message "nothing to undo"
else lastCommand.undo();
```



# מימוש undo במספר רמות

- כדי לממש undo במספר רמות צריך לשמור רשימה history של פקודות, ומצביע current למקום הנוכחי ברשימה.
- כאשר נתבקש לבצע undo , יבוצע (אם current מצביע לאבר קיים)

```
history.current.undo();
```

*move current back one step*

- בהקשר זה ניתן לראות כיצד יבוצע redo (באותה הנחה):

*move current forward one step*

```
history.current.redo();
```

# הערות כלליות

- איך למצוא את העצמים (והמחלקות?) - להציע מועמדים, לפסול את חלקם (אם כל תכונותיהם כבר כוסו).
- שם של מחלקה יהיה בדרך כלל שם עצם. לעיתים תהיה מחלקה מופשטת או מנשק ששמה הוא שם תואר - למשל  
Comparable
- השגיאה הרווחת - להגדיר כמחלקה משהו שאינו כזה. אם שאומרים "המחלקה XXX מבצעת ..." כנראה זה שרות.
- מחלקה עם שרות יחיד בדרך כלל לא צריכה להיות מחלקה
- ירושה - לעיתים יש נטייה להתחיל הקלסיפיקציה מוקדם מדי
- הפשטה מעורבת - כל השירותים של מחלקה צריכים להתייחס אל הפשטה יחידה, שברור מהי.