

תוכנה 1 בשפת Java  
שיעור מספר 14:  
"מה המצב, אחי?"

סיון טולדו  
אוהד ברזילי

בית הספר למדעי המחשב  
אוניברסיטת תל אביב

# דוגמא מודרכת

המצגת מכילה קטעים הספר:

Object-Oriented Software Construction, 2nd edition,  
by Bertrand Meyer (Prentice Hall) .

כל הזכויות שמורות למחברים

# דוגמא מודרכת

## עיצוב מערכת עם לוחות מרובים

### - Enquiry on Flights -

Flight sought from:  To:

Departure on or after:  On or before:

Preferred airline (s):

Special requirements: \_\_\_\_\_

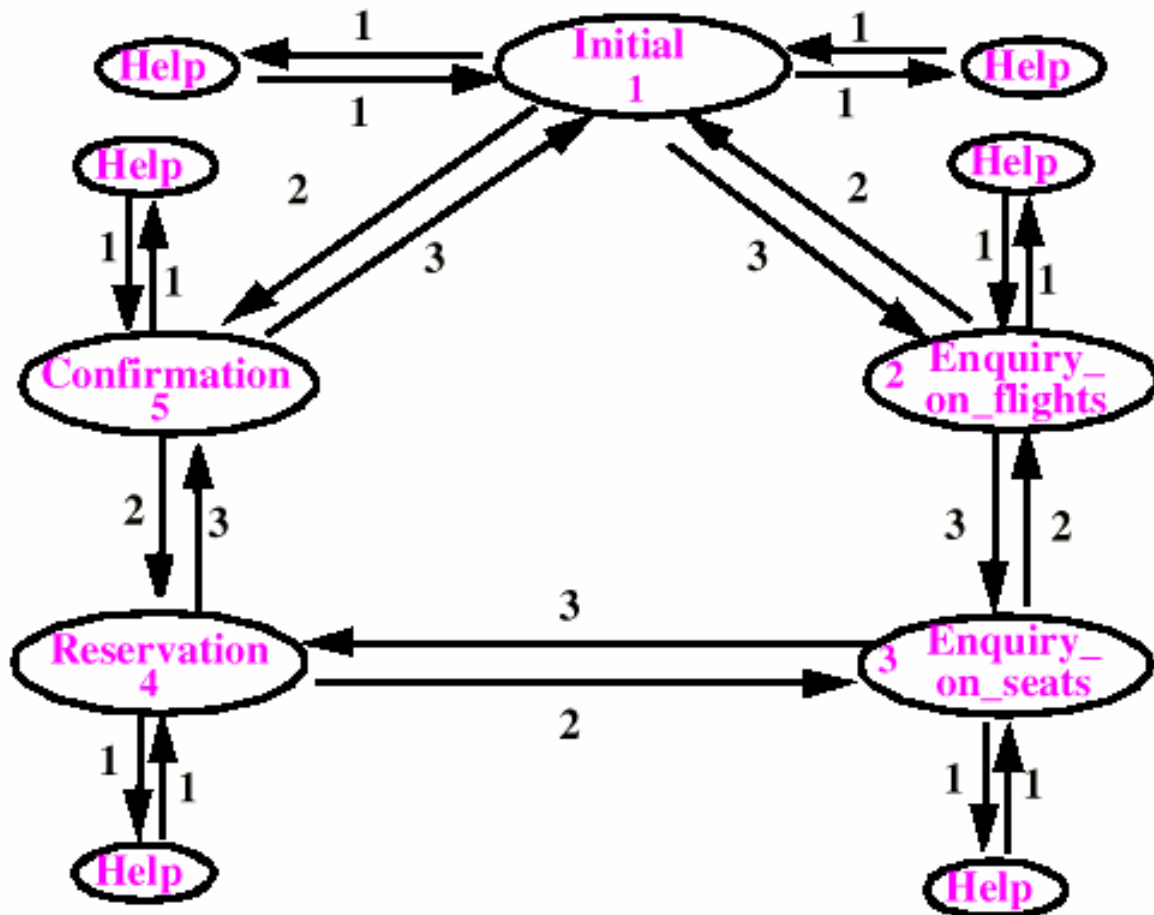
AVAILABLE FLIGHTS: 1

**Flt# AA 42      Dep 8:25      Arr 7:45      Thru: Chicago**

Choose next action:

- 0 — **Exit**
- 1 — **Help**
- 2 — **Further enquiry**
- 3 — **Reserve a seat**

# המערכת כאוטומט מצבים



תוכנה 1 בשפת Java  
אוניברסיטת תל אביב

# נסיון ראשון - פסאודוקוד

*Enquiry:*

“Display *Enquiry on flights* panel”

**repeat**

“Read user’s answers and choice *C* for the next step”

**if** “Error in answer”

**then** “Output appropriate message” **end**

**until not** “error in answer” **end**

“Process answer”

**switch** (*C*)

{

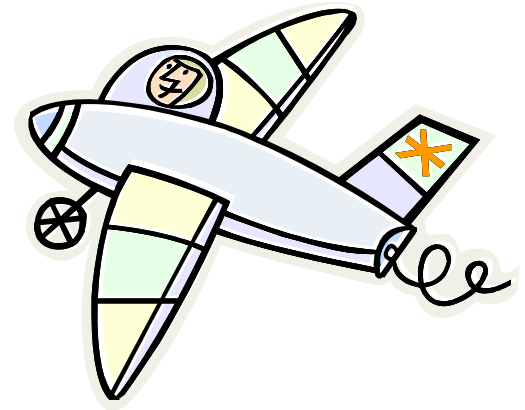
*case C*<sub>0</sub> : **goto** *Exit*

*case C*<sub>1</sub> : **goto** *Help*

*case C*<sub>2</sub> : **goto** *Reservation*

...

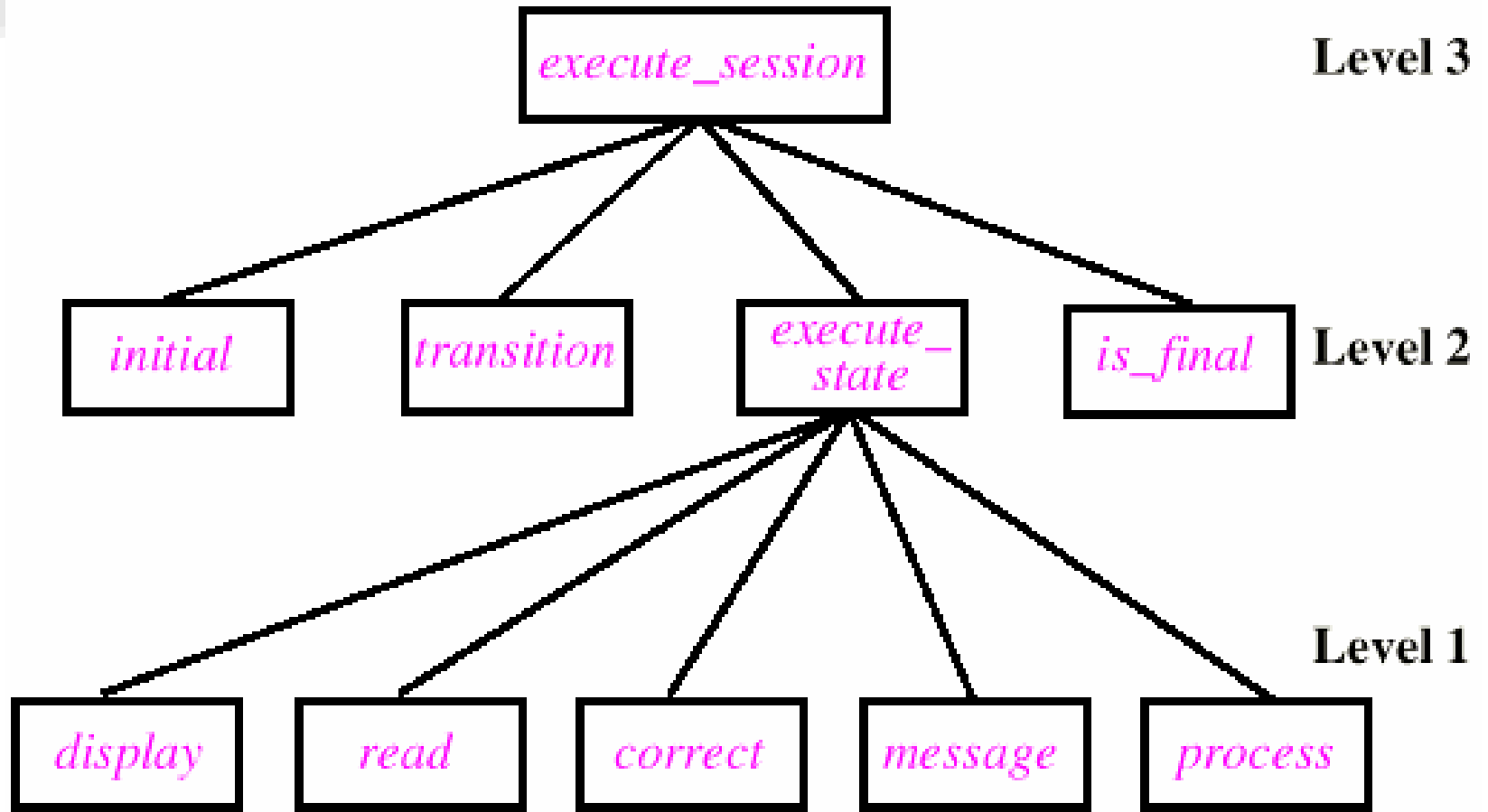
}



# טבלת מעבר מצבים

Choice → ↓ State	0	1	2	3
<b>1</b> ( <i>Initial</i> )	<b>-1</b>	<b>0</b>	<b>5</b>	<b>2</b>
<b>2</b> ( <i>Flights</i> )		<b>0</b>	<b>1</b>	<b>3</b>
<b>3</b> ( <i>Seats</i> )		<b>0</b>	<b>2</b>	<b>4</b>
<b>4</b> ( <i>Reserv.</i> )		<b>0</b>	<b>3</b>	<b>5</b>
<b>5</b> ( <i>Confirm</i> )		<b>0</b>	<b>4</b>	<b>1</b>
<b>0</b> ( <i>Help</i> )		<i>Return</i>		
<b>-1</b> ( <i>Final</i> )				

# פירוק פונקציונלי top-down



# execute\_session()

```
/** Execute a complete session of the interactive system */
public static void execute_session() {

    int state, choice;

    state = initial();

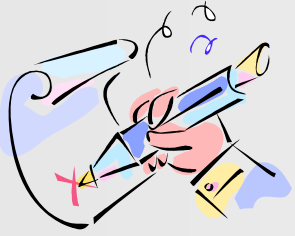
    while (!is_final(state)) {
        choice = execute_state(state);
        state = transition(state, choice);
    }
}
```



# execute\_state()

```
/**
 * Execute the actions associated with state s, returning the
 * user's choice for the next state
 */
private static int execute_state(int s) {
    int a = -2;
    boolean ok = false;

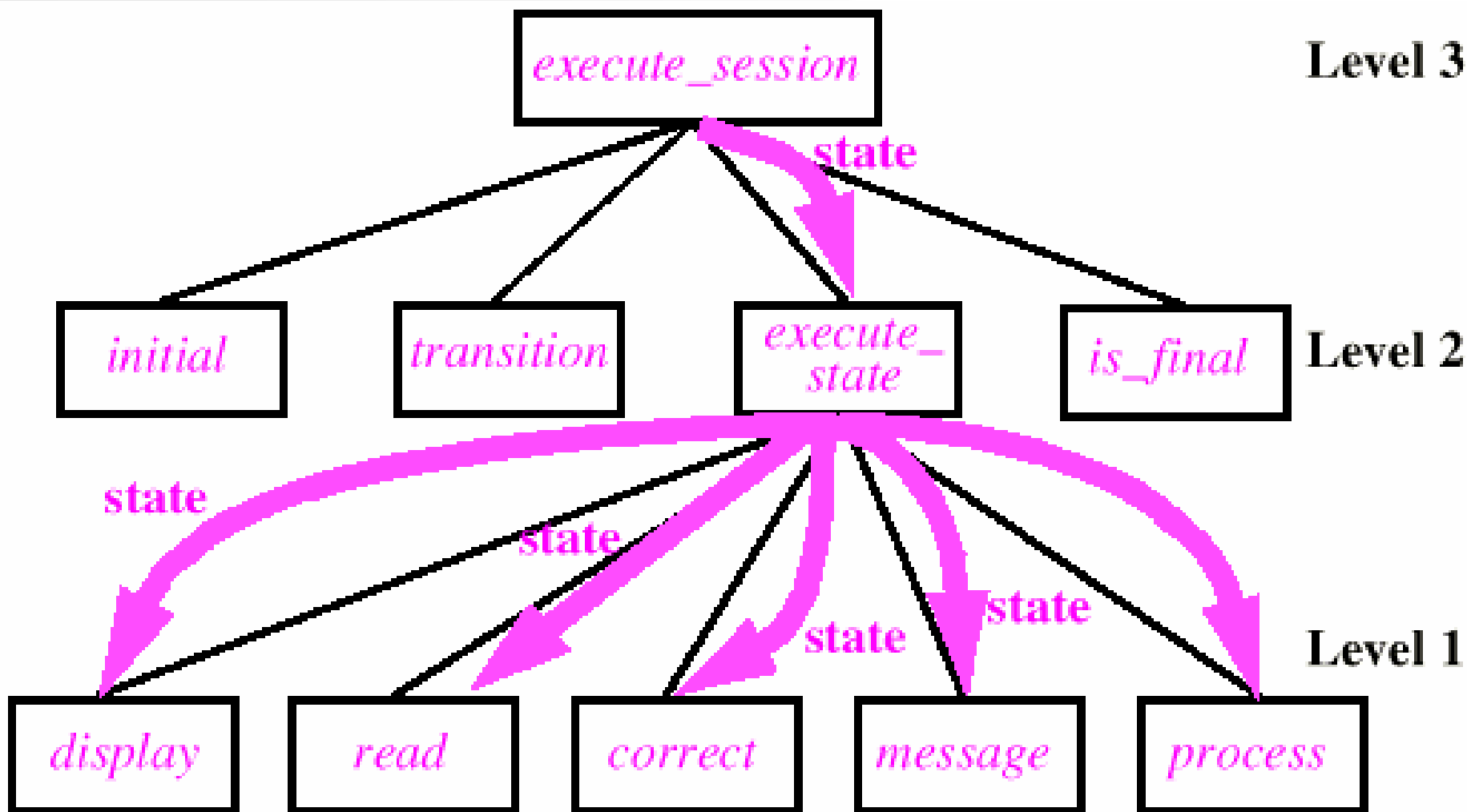
    while (!ok) {
        display(s);
        a = read(s);
        ok = correct(s, a);
        if (!ok)
            message(s, a);
    }
    process(s, a);
    return next_choice(a);
}
```



# חתימת המתודות

<i>int execute_state</i>	<i>(STATE s);</i>
<i>void display</i>	<i>(STATE s);</i>
<i>int read</i>	<i>(STATE s);</i>
<i>bool correct</i>	<i>(STATE s, int a);</i>
<i>void message</i>	<i>(STATE s, int a);</i>
<i>void process</i>	<i>(STATE s, int a);</i>

# זרימת המידע



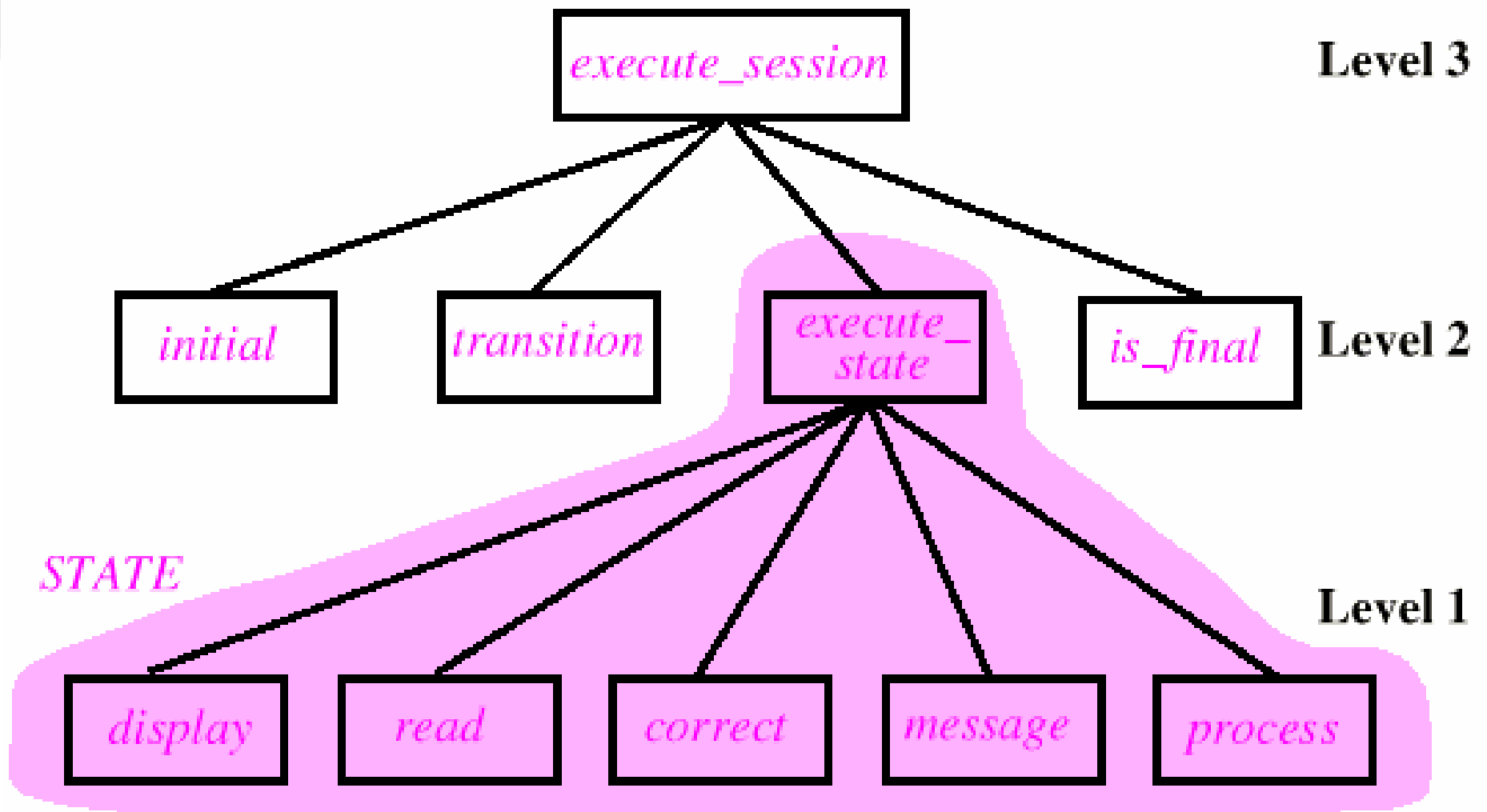
# נקמת הנתונים

display, read, correct, איך נראה מבנה השרותים ■  
? message, process

מהצורה: ■

```
inspect
  S
when Initial then
  ...
when Enquiry_on_flights then
  ...
...
end
```

# STATE תכונות



# STATE כמחלקה

```
public abstract class STATE {  
  
    /** User's answer to questions asked in this state */  
    protected int input;  
  
    /** User's choice for next step */  
    protected int choice;  
  
    /** Display panel associated with current state. */  
    abstract public void display();  
  
    /** get user's answer into input and choice into choice */  
    abstract public void read();  
  
    /** Is input a correct answer? */  
    abstract public boolean correct();  
}
```

# STATE כמחלקה

```
public abstract class STATE {  
    ...  
  
    /** Output error message corresponding to input  
     * @pre: !correct()  
     */  
    abstract public void message();  
  
    /** Process input  
     * @pre: correct()  
     */  
    abstract public void process();  
}
```

# STATE כמחלקה

```
public abstract class STATE {  
    ...  
  
    /** Execute actions associated with current state  
     * and set choice to denote user's choice for next state.  
     * @post: ok  
     */  
    public void execute() {  
        boolean ok = false;  
  
        while (!ok) {  
            display();  
            read();  
            ok = correct();  
  
            if (!ok)  
                message();  
  
            process();  
        }  
    }  
}
```



# מצבים ספציפיים יורשים מ STATE

```
public class ENQUIRY_ON_FLIGHTS extends STATE {
```

```
    @Override
```

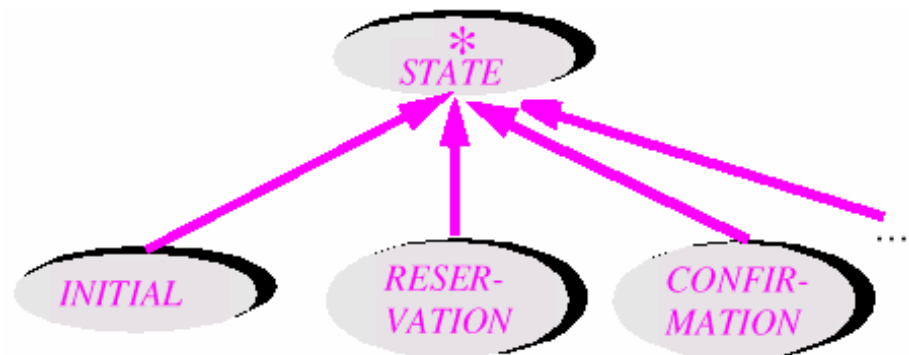
```
    public void display(){
```

```
        /* Specific display procedure */
```

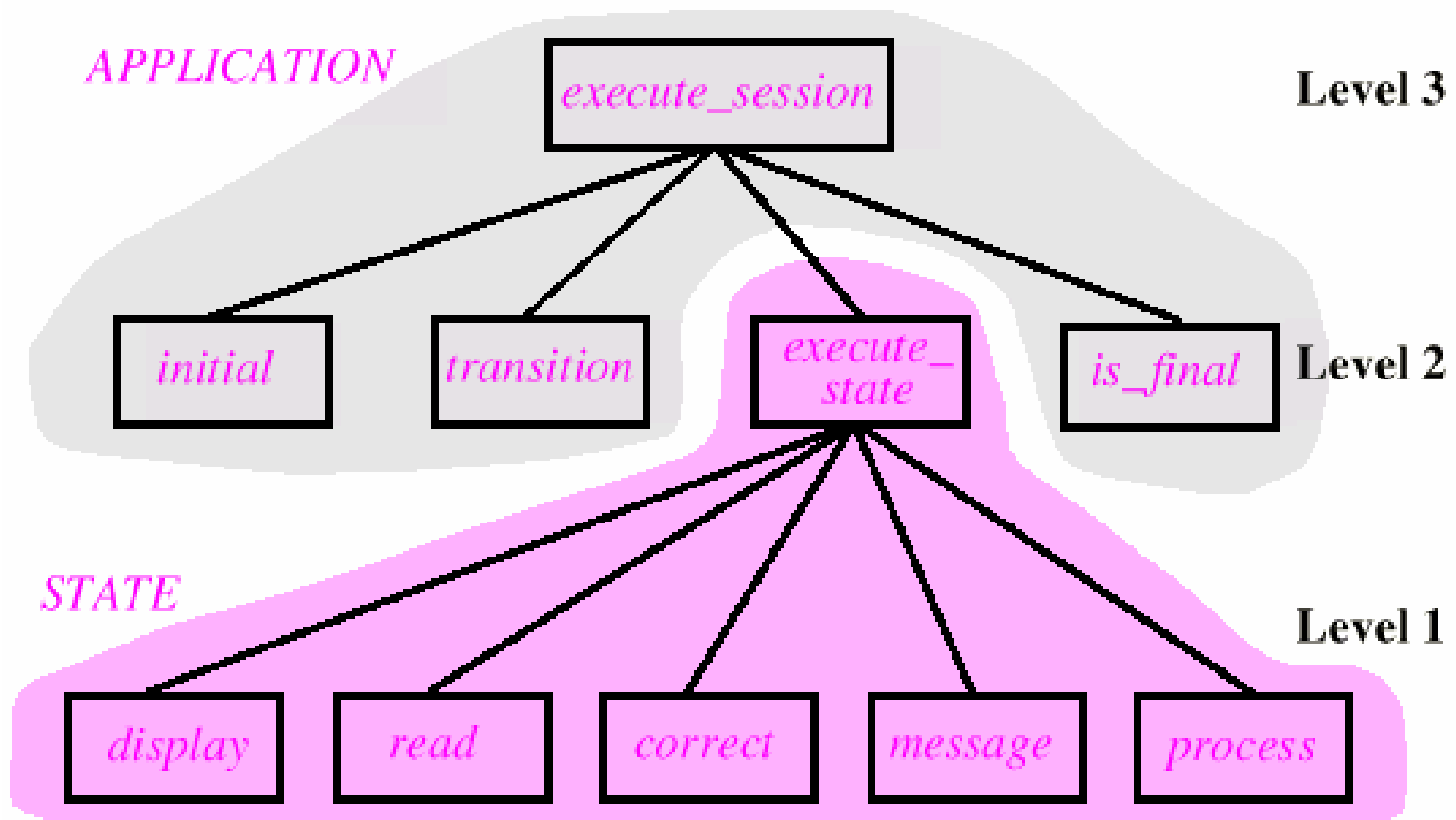
```
    }
```

```
... And similarly for read, correct, message and  
process...
```

```
}
```



# תכונות של STATE ושל APPLICATION



# המחלקה APPLICATION

## משתמשת ב STATE

```
/**
 * "Interactive panel-driven applications"
 * @inv: transition.length == associated_state.size()
 */
class APPLICATION {

    /**
     * Allocate application with n states and m possible
     * choices
     */
    public APPLICATION(int n, int m) {
        transition = new int[n + 1][m + 1];
        associated_state = new ArrayList<STATE>(n + 1);
    }
}
```

# המחלקה APPLICATION

```
/** Perform a user session */
public void execute() {
    STATE st;
    int st_number;

    st_number = initial;
    while (st_number != 0) {
        st = associated_state.get(st_number);

        // This refers to the execute procedure of STATE
        st.execute();

        st_number = transition[st_number][st.choice];
    }
}
```

# המחלקה APPLICATION

```
/** Element change Enter state st with index sn.
 * @pre: 0 < sn && sn <= associated_state.size()
 */
public void put_state(STATE st, int sn) {
    associated_state.set(sn, st);
}

/** Define state number sn as the initial state
 * @pre: 0 < sn && sn <= associated_state.size()
 */
public void choose_initial(int sn) {
    initial = sn;
}
```

# המחלקה APPLICATION

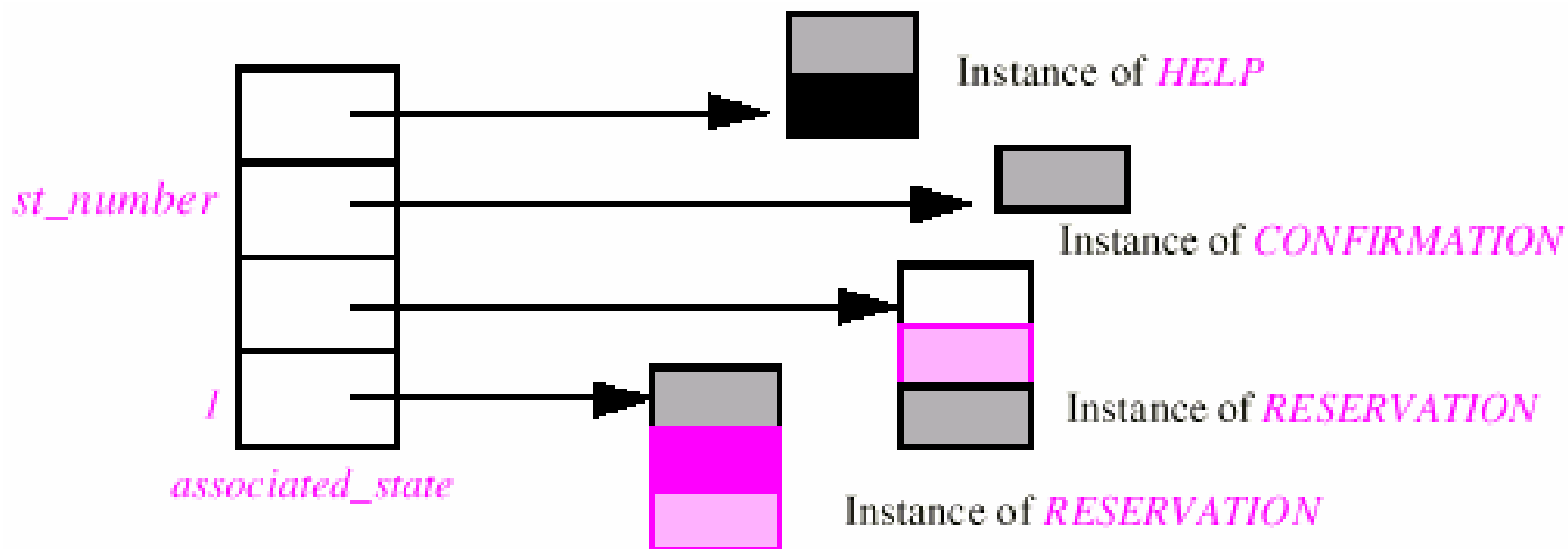
```
/**
 * Enter transition labeled label from state number source to state
 * number target
 *
 * @pre: 1 <= source
 * @pre: source <= associated_state.size()
 *
 * @pre: 0 <= target
 * @pre: target <= associated_state.size()
 *
 * @pre: 1 <= label && label <= transition[0].length
 */
private void enter_transition(int source, int target, int label) {
    transition[source][target] = label;
}
```

# המחלקה APPLICATION

```
/* Initial state's number * */  
private int initial;  
  
/* transition table */  
private int[][] transition;  
  
/* vector of states */  
private List<STATE> associated_state;  
  
...  
  
} // class APPLICATION
```

# המחלקה APPLICATION

■ היישום מחזיק מערך פולימורפי של STATES





# בניית יישום

■ אתחול עצם של APPLICATION:

```
APPLICATION air_reservation =  
    new APPLICATION(number_of_states, number_of_possible_choices);
```

■ הגדרים המצבים השונים ומספורם:

```
air_reservation.put_state(s, i);
```

■ בחירת המצב ההתחלתי:

```
air_reservation.choose_initial(i0);
```

■ הגדרת המעבר ממצב למצב:

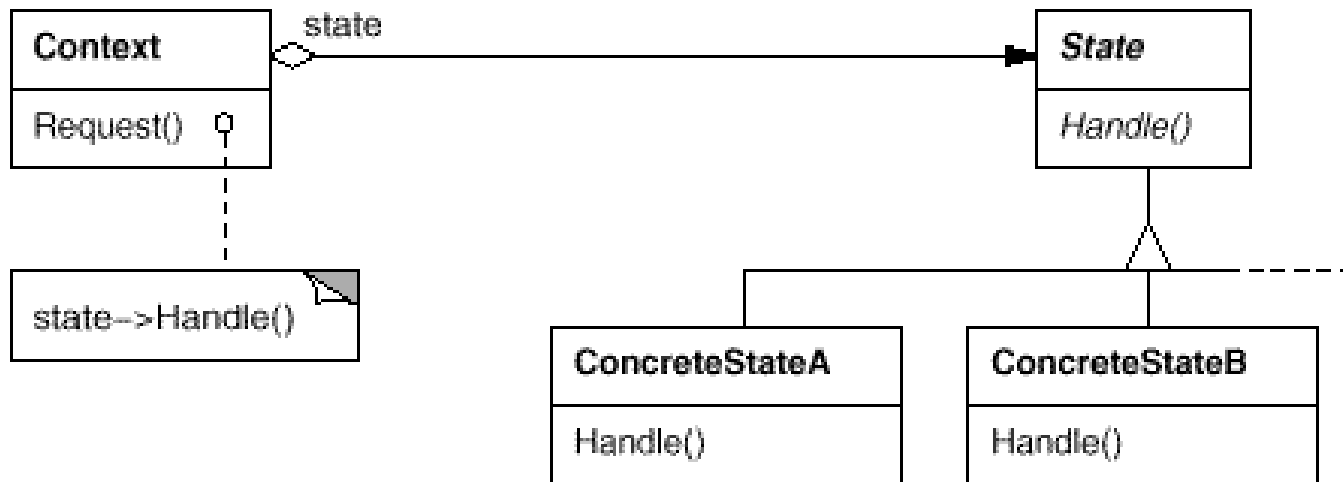
```
air_reservation.enter_transition(sn, tn, l);
```

■ הפעלת היישום:

```
air_reservation.execute();
```

# תבנית העיצוב STATE

■ המקרה הכללי:



■ כלפי חוץ נראה כאילו ה context משנה את טיפוסו בצורה דינאמית

# תבנית העיצוב STATE

- התבנית מרכזת התנהגות תלוית מצב למחלקה נפרדת
  - המצבים עצמם לא 'מודעים' לקיומם של מצבים אחרים ולא יושפעו מהוספה/הסרה/שינוי של מצבים אחרים
  - ניתן לחשוב על גישות אחרות
- המעבר בין המצבים השונים מופיע בקוד בצורה מפורשת
- אם מצבי הנגזרת לא מכילים שדות ניתן להשתמש באותם מצבים עבור כמה יישומים במקביל (כמה TCPConnection למשל)
  - מימוש אפשרי ע"י enum