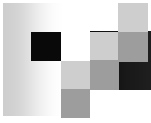


תוכנה 1 – תרגול 14

# The “Hyperbaric Chamber” Question





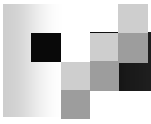
# האפליקציה: Hyperbaric Chamber

■ מטרה: שמירה על תנאים מסוימים בתא לחץ (hyperbaric chamber) ברמה פחות או יותר קבועה.

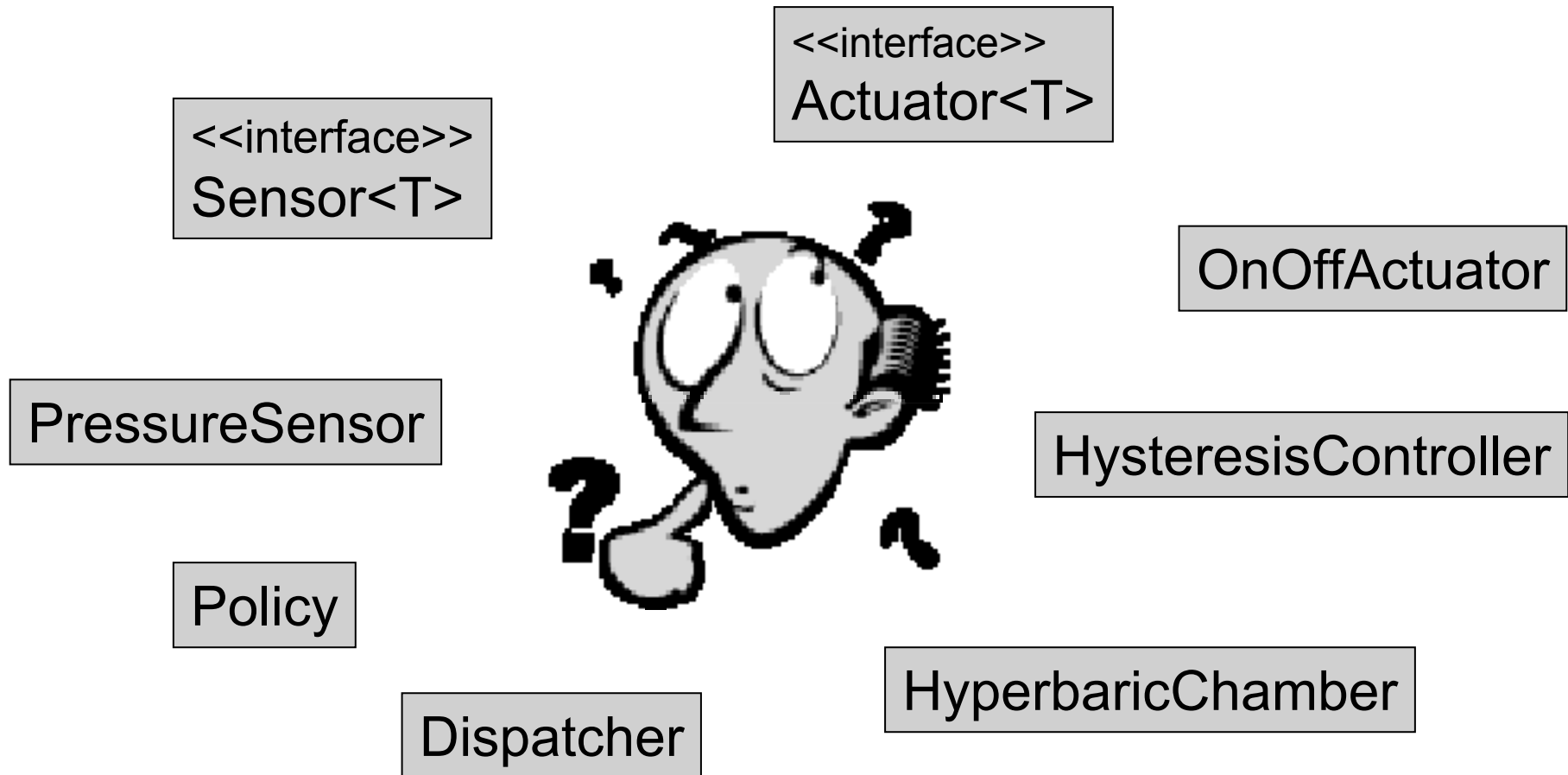
■ אמצעים:

□ ביצוע מדידות באמצעות חיישנים: לחץ, טמפרטורה, ...

□ הפעלת רכיבים - לדוגמא באמצעות מתגים.



# עיצוב (חלקי)




## מנשקי חיישנים ומפעילים

```
public interface Sensor <T> {  
    public T read();  
}
```



```
public interface Actuator <T> {  
    public void act(T command);  
}
```



# חיישנים ומפעילים קונקרטיים

```
public class PressureSensor implements Sensor<Double> {  
    public Double read() {...}  
}
```

```
public class OnOffActuator implements Actuator<Boolean> {  
    public void act(Boolean turn_it_on) {...}  
}
```



# Policy & Dispatcher

```
public abstract class Policy {  
    // missing declarations and/or code  
    abstract public void testAndRespond();  
}  
public class Dispatcher {  
    public static void dispatch() {  
        // missing implementation  
    }  
}
```

מחלקות שיורשות מ- Policy ומממשות את testAndRespond:  
- יעשו שימוש ב- Sensor ו- Acutator במימוש המתודה.  
- המתודה (הקונקרטית) testAndRespond תקרא באופן מחזורי ע"י Dispatcher



## Policy & Dispatcher (2)

- איך ה- Dispatcher ניגש לכל האובייקטים מסוג Policy ?
- פתרון: צריך לתחזק רשימה של כל האובייקטים מסוג Policy
  - הרשימה תוחזק ע"י שדה מחלקה (סטטי) ב- Policy.
  - ל- Policy יהיה בנאי שיוסיף את האובייקט הנוכחי (this) לרשימה.
  - הוספת שרות מחלקה ל- Policy שיחזיר את הרשימה או iterator שלה



# HysteresisController

```
public class HysteresisController {
    private double low_threshold, high_threshold;
    private Sensor<Double> sensor;
    private Actuator<Boolean> actuator;
    public HysteresisController(Sensor<Double> s,
        Actuator<Boolean> a, double l, double h) {
        low_threshold = l; high_threshold = h;
        sensor = s;    actuator = a;
        Policy too_low = new Policy () {
            public void testAndRespond() {
                double x = sensor.read();
                if (x < low_threshold) actuator.act(true);
            } };
    };
}
```

unboxing

autoboxing

# HysteresisController (cont.)

```
public class HysteresisController {  
    ..  
    public HysteresisController(Sensor<Double> s,  
        Actuator<Boolean> a, double l, double h) {  
        ...  
        Policy too_high = new Policy () {  
            public void testAndRespond() {  
                double x = sensor.read();  
                if (x > high_threshold) actuator.act(false);  
            }  
        };  
    }  
}
```



• HysteresisController אינו מכיל מתודות (רק בנאי). למה?  
• האם לאחר הפעלות חוזרות ונשנות של 2 מתודות ה-  
testAndRespond תהיה הפעלה של ה- garbage collector?

# HyperbaricChamber

```
public class HyperbaricChamber {  
    public static void main(String[] args) {  
        PressureSensor sensor = new PressureSensor();  
        OnOffActuator actuator = new OnOffActuator();  
        double low_thresh = Double.parseDouble(args[0]);  
        double high_thresh = Double.parseDouble(args[1]);  
        new HysteresisController(sensor, actuator,  
                                low_thresh, high_thresh);  
        Dispatcher.dispatch();  
    }  
}
```



מהם תנאי הקדם ל- main?  
כיצד ניתן לבטל אותם ומדוע כדאי לעשות  
זאת?

# "מתג חכם"

- כיצד ניתן לשנות את הממשק של OnOffActuator כך שניתן לדעת אם ההפעלה הצליחה?
- העלאת חריג  $\leq$  יש לעדכן את הממשק Actuator
- הוספת שאילתא
- הוספת מימוש ממשק `Sensor<Boolean>` (דומה להוספת שאילתא אבל אלגנטי יותר)



# HysteresisDownController

■ HysteresisController:

□ `actuator.act(false)`  $\Leftarrow$  קריאה גבוהה

□ `actuator.act(true)`  $\Leftarrow$  קריאה נמוכה

■ כיצד ניתן לממש, במינימום שכפול קוד, מחלקה

בשם `HysteresisDownController` שמבצעת:

□ `actuator.act(true)`  $\Leftarrow$  קריאה גבוהה

□ `actuator.act(false)`  $\Leftarrow$  קריאה נמוכה



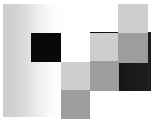
## HysteresisDownControler (2)

- פתרון: להשתמש ב- HysteresisController ול"טפל" ב- Actuator או ב- Sensor וערכי הסף:
  - Actuator "הפוך": עוטף Actuator תקין והופך את ערך הפרמטר ( $\text{true} \Leftrightarrow \text{false}$ )
  - פתרון דומה אך מסובך יותר: Sensor "הפוך" (עוטף Sensor ומחזיר את מינוס הערכים) + שינוי ערכי הסף ( $\text{new\_low} = -\text{high}$ ,  $\text{new\_high} = -\text{low}$ )



# HysteresisDownController (3)

```
class HystersisDownController{
    public class ReverseActuator implements
                                   Actuator<Boolean>{
        Actuator<Boolean> actuator;
        public ReverseActuator(Actuator<Boolean> a){
            actuator = a;
        }
        public void act(Boolean command) {
            act(!command);
        }
    }
    public HystersisDownController(Sensor<Double> s,
        Actuator<Boolean> a, double l, double h){
        new HysteresisController(s,
            new ReverseActuator(a),l,h);
    }
}
```



11.02.07

בהצלחה בבחינה!

