

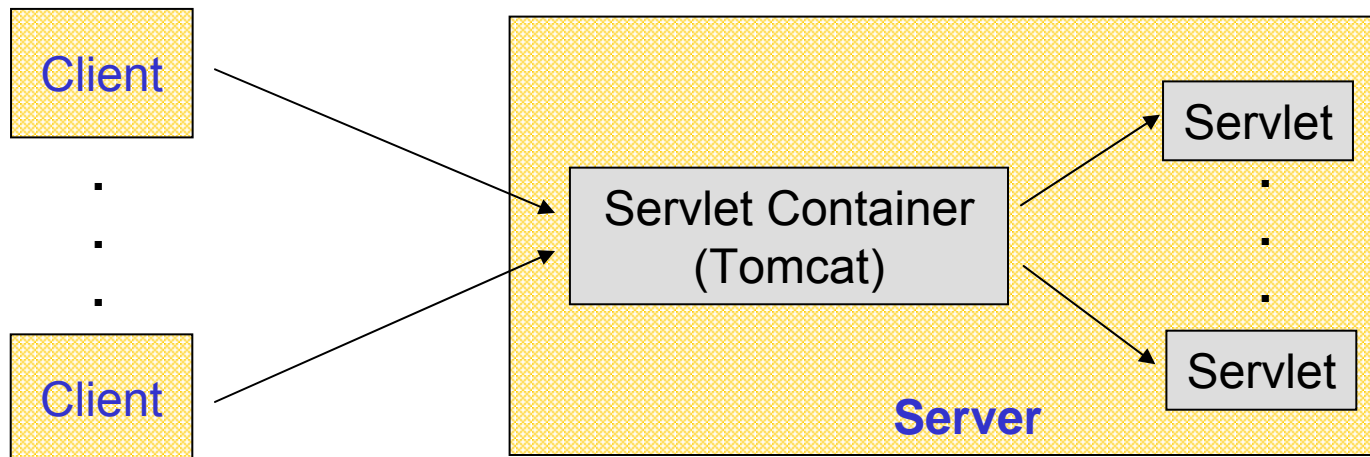


Software 1 with Java

Recitation No. 7
(Servlets, Inheritance)

Servlets

- Java modules that run on a Web server to answer client requests
- For example:
 - Processing data submitted by a browser
 - Providing dynamic content



The Servlet Interface

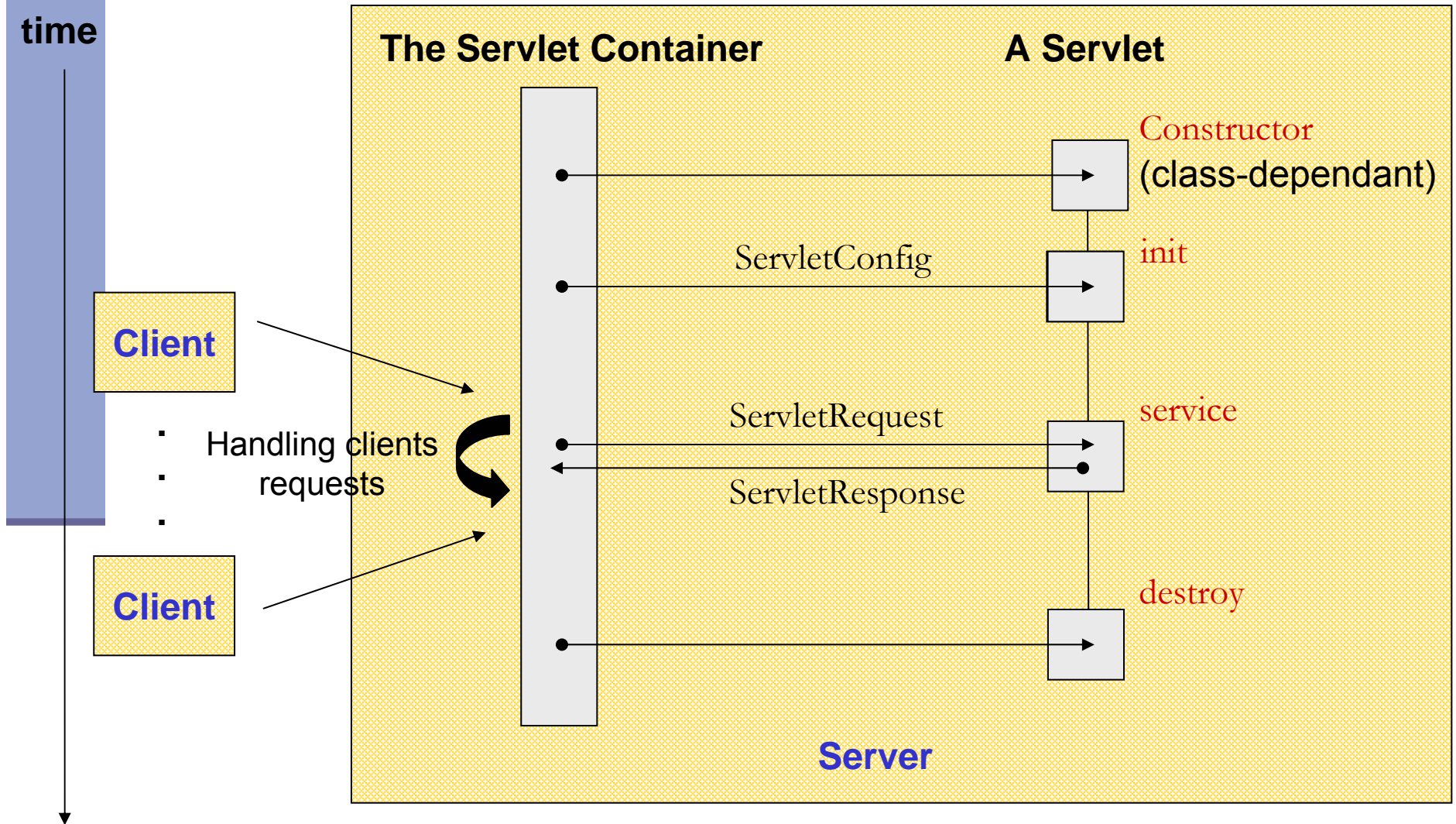
- An interface in the `javax.servlet` package

**Life-cycle
methods**

```
<interface>
Servlet

public void init(ServletConfig config);
public void destroy();
public void service(ServletRequest req, ServletResponse res);
...
```

The Servlet Life Cycle

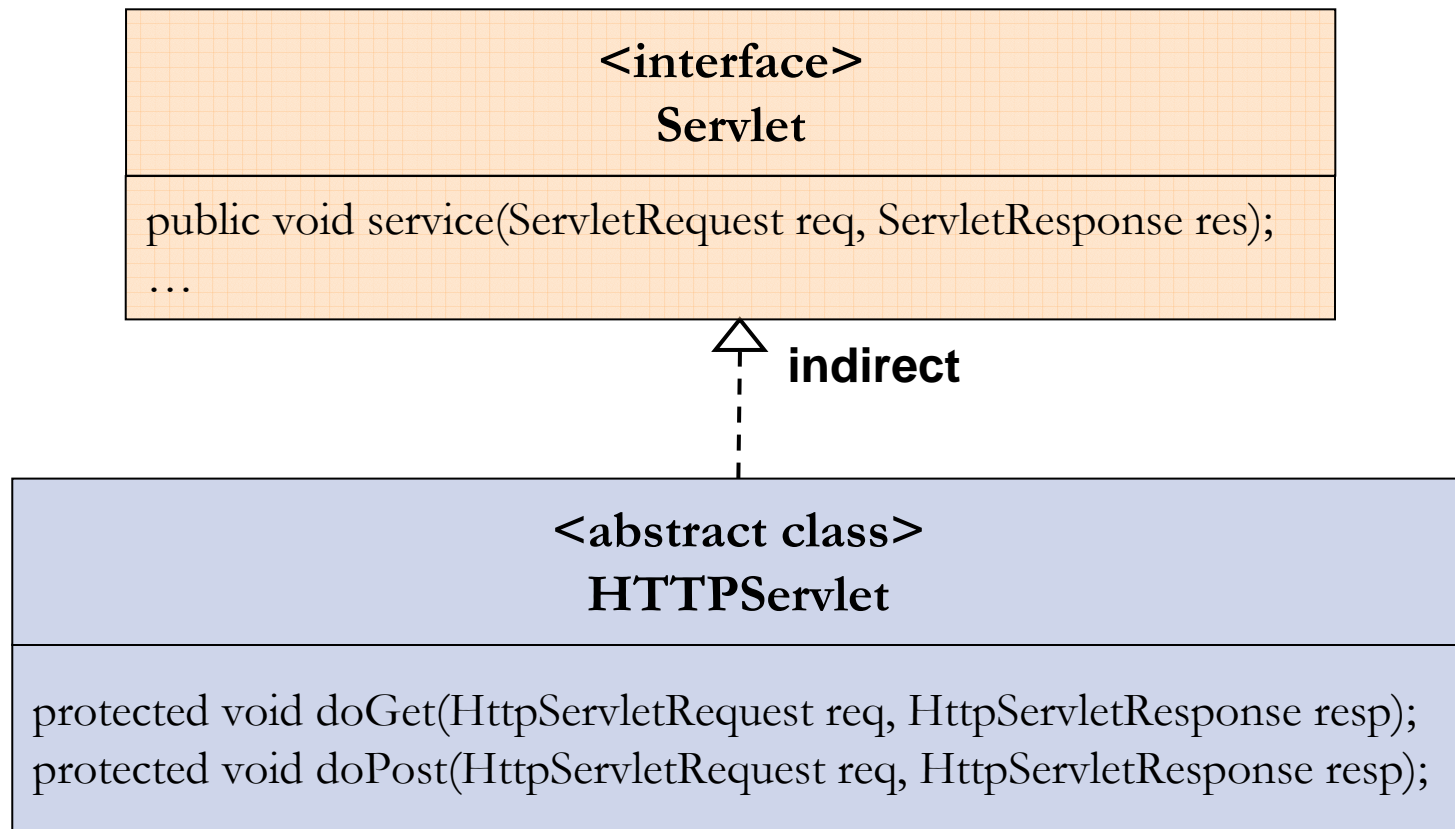


HTTP Servlets

- HTTP (**H**ypertext **T**ransfer **P**rotocol):
 - A protocol used by a WWW client like a browser to send requests to a Web Server
 - A request-response oriented protocol
 - Most common HTTP requests:
 - GET – the parameters are passed in the URL
 - POST – for posting unlimited data to server
- Extend the abstract class `HTTPServlet` in the `javax.servlet.http` package

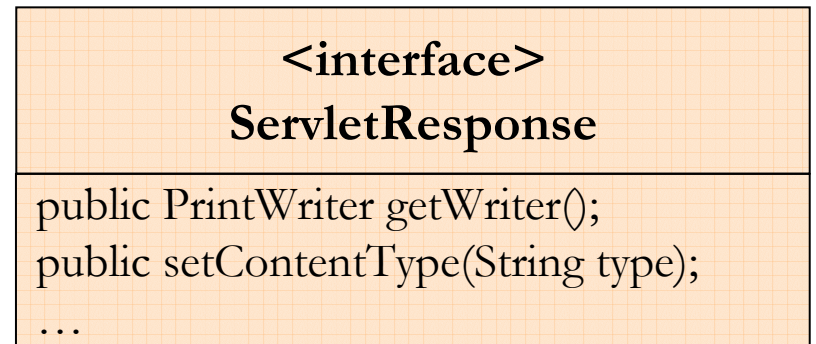
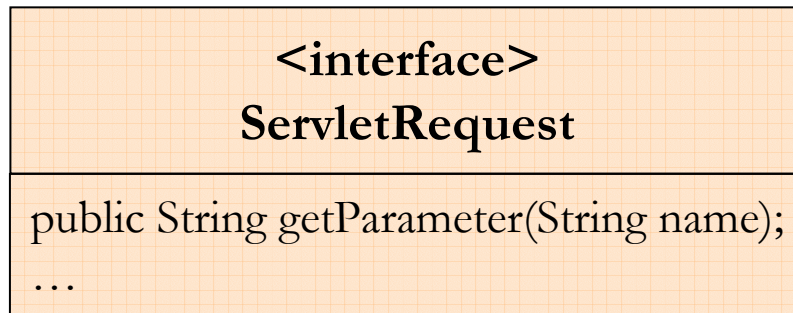
HTTP Servlets

- Extend the `HTTPServlet` abstract class in the `javax.servlet.http` package

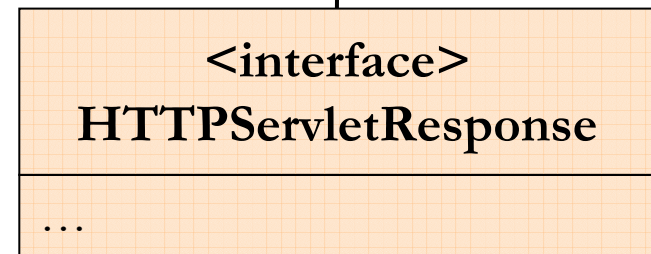
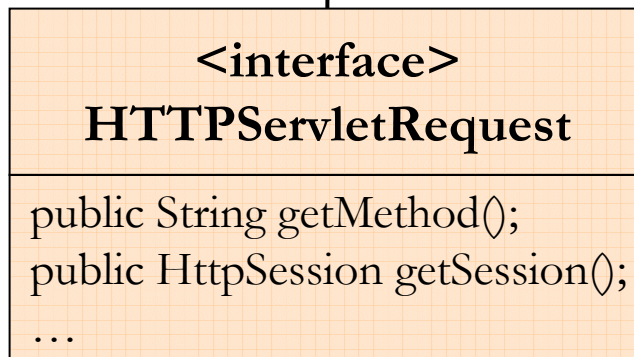


Requests and Responses

javax.servlet
package



javax.servlet.http
package



HTTP Sessions

- Session: A server-side context associated with a user when visiting a Web site.
- The session is kept for a specified period of time across more than one page request or visit of the user.
- Most common ways for servers to maintain sessions:
 - using cookies
 - rewriting URLs

HTTP Sessions

- The servlet container tracks sessions
- A combined mechanism for tracking a session:

if the client supports and enables cookies

use them

otherwise

automatically revert to URL-rewriting

The HttpSession Interface

- Holds information about a session, e.g.
 - the session identifier
 - creation time
 - last accessed time
- Binds objects to a session:
 - Object **getAttribute**(String name)
 - void **setAttribute**(String name, Object value)

Online Resources

- Sun tutorial on servlets:

http://java.sun.com/j2ee/tutorial/1_3-fcs/doc/Servlets.html

- <http://www.novocode.com/doc/servlet-essentials/>

- Tomcat and Servlet API:

<http://tomcat.apache.org>



Software 1 with Java

Inheritance
(Example: Geometric Shapes)

Class Polygon

Polygon

```
public Polygon(List vertices)
public double perimeter()
public void display()
public void rotate(Point center, double angle)
public void translate(Point p)
public int count()
...
```

Class Polygon

```
/**
 * @inv count() >=3,
 */
public class Polygon {
    /** @pre vertices.size() >=3,
        @pre vertices is a list of Point Objects
    */
    public Polygon(List vertices) {
        this.vertices = vertices;
    }
    ...
    /**
     * Successive points making up the polygon
     */
    private List vertices;
}
```

Class Polygon

```
public class Polygon {
    ...

    /** Length of the perimeter */
    public double perimeter() { ... }

    /** Display polygon on screen */
    public void display() { ... }

    /** Rotate by angle around center */
    public void rotate(Point center, double angle) { ... }

    /**
     * Move by p.x() horizontally and p.y() vertically
     */
    public void translate(Point p) { ... }

    /** Returns the number of sides */
    public int count() { return vertices.size(); }
}
```

Method Implementation

```
/**
 * Rotate by angle around center
 */
public void rotate(Point center, double angle) {
    for (Object p : vertices) {
        ((Point) p).rotate(center, angle);
    }
}
```


Method Implementation

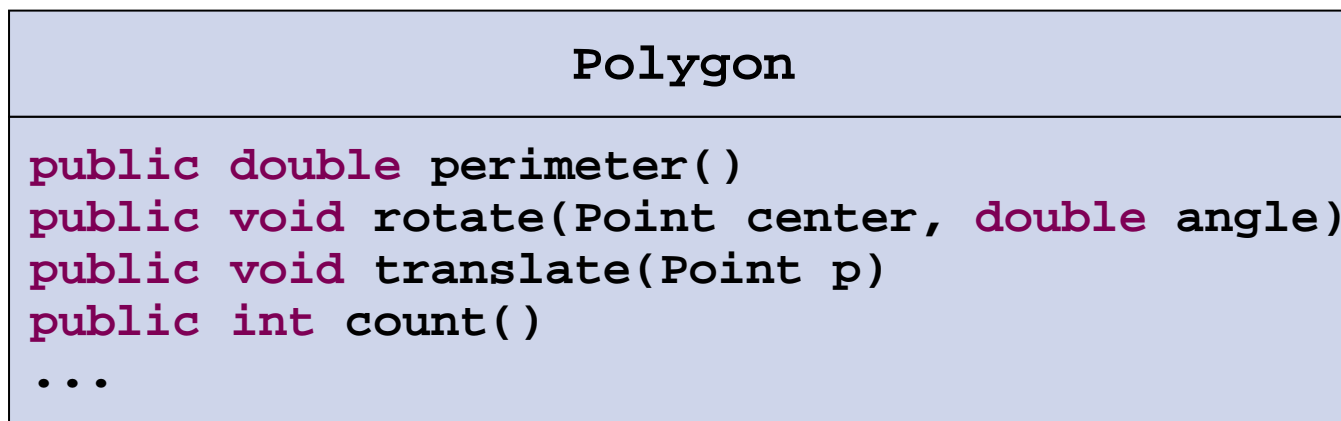
```
/** Returns the polygon's perimeter */
public double perimeter() {
    double result = 0.0;

    for (int i = 0 ; i < count() ; i++) {
        Point curr = (Point) verices.get(i);
        Point next = (Point) vertices.get((i+1)%count());
        result += curr.distance(next);
    }

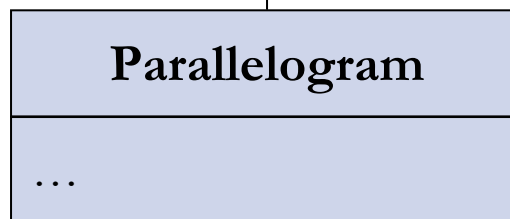
    return result;
}
```

Class Parallelogram

- מקבילית היא סוג של פוליגון
- Parallelogram היא מקרה פרטי של Polygon



קשר ירשה
ב-JAVA הרחבה (extension)

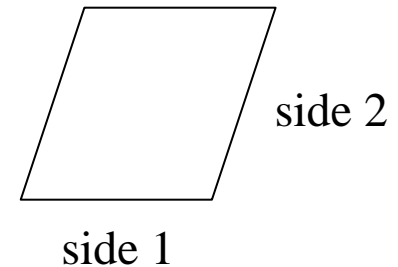


כל תכונות המצולע (המתודות)
תקפות למקבילית

Class Parallelogram

```
/**
 * @inv count() == 4
 * @inv vertices.get(0).distance(vertices.get(1)) ==
 *       side1
 * Similar invariants for the other 3 sides.
 */
public class Parallelogram extends Polygon {
    ...

    /** The two side lengths */
    private double side1, side2;
}
```



The Constructor

```
public class Parallelogram extends Polygon {
    /**
     * @pre vertices.size() == 4
     * @pre vertices.get(0).distance(vertices.get(1))
     * == vertices.get(2).distance(vertices.get(3))
     * @pre vertices.get(1).distance(vertices.get(2))
     * == vertices.get(0).distance(vertices.get(3))
     */
    public Parallelogram(List vertices) {
        super(vertices);
        // Setting side1, side2
        ...
    }
    ...
}
```