Builder Pattern

The Builder Pattern is a **creational design pattern** that lets you construct complex objects **step by step**. Instead of a massive constructor with many parameters, you set each field via named methods and finalize with build().

Problem Statement

You're building an HTTP Client Library. Users need to create HTTP requests with:

1. URL (required)
2. Method – GET, POST, etc. (default: GET)
3. Headers – Content-Type, Authorization (optional)
4. Body – JSON payload (optional)
5. Timeout – in ms (default: `30000`)

How do you let users create HttpRequest objects cleanly with multiple optional fields?

Approach 1 (Single Large Constructor)

```
public class HttpRequest {
    public HttpRequest(String url, String method, Map<String, String> headers,
                       String body, int timeout) { ... }
}

// Usage — forced to pass everything, even if you only need URL
HttpRequest req = new HttpRequest(
    "https://api.example.com",
    "GET",      // just want default
    null,       // no headers needed
    null,       // no body needed
    30000       // just want default
);
```

Must pass all params

Approach 2 (Telescoping Constructors)

```
public class HttpRequest {
    public HttpRequest(String url) { ... }
    public HttpRequest(String url, String method) { ... }
    public HttpRequest(String url, String method, String body) { ... }
    public HttpRequest(String url, String method, String body, int timeout) { ...
}
    // Need more combinations? Add more constructors...
}

// Usage
HttpRequest req = new HttpRequest("https://api.example.com", "POST");
```

Can't have all combinations:

```
public HttpRequest(String url, String method) { ... }
public HttpRequest(String url, String body) { ... }
```

Try adding a new param (need all the P&C)

Approach 3 (Setters)

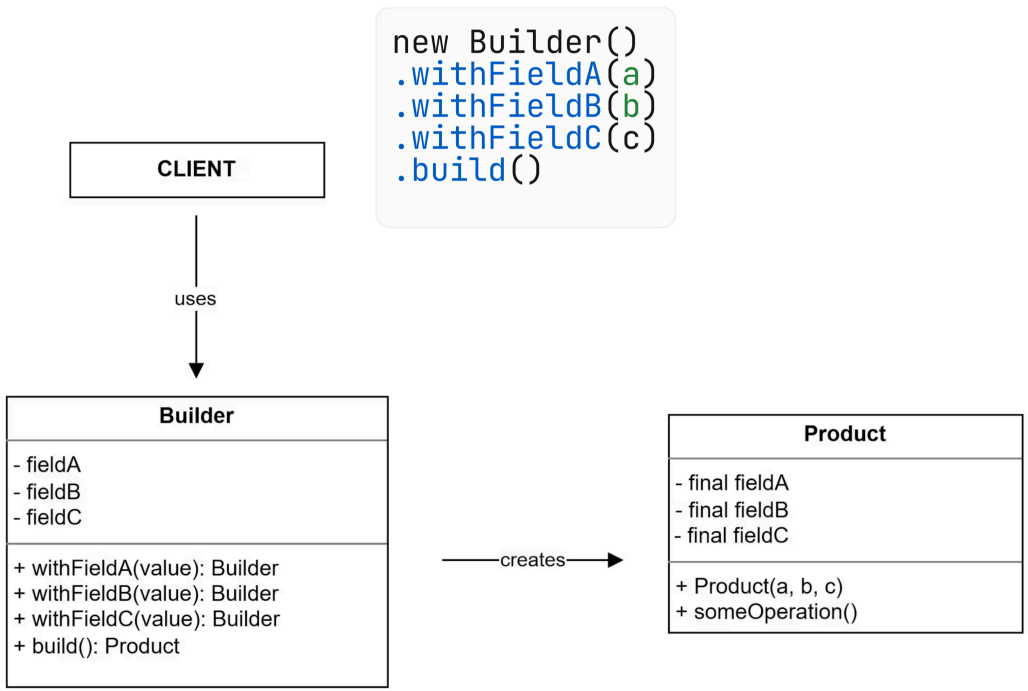
```java
public class HttpRequest {
    private String url;
    private String method;
    private String body;
    private int timeout;

    public void setUrl(String url) { this.url = url; }
    public void setMethod(String method) { this.method = method; }
    public void setBody(String body) { this.body = body; }
    public void setTimeout(int timeout) { this.timeout = timeout; }
}

// Usage
HttpRequest req = new HttpRequest();
req.setUrl("https://api.zerotechdebt.com");
req.setMethod("POST");
```

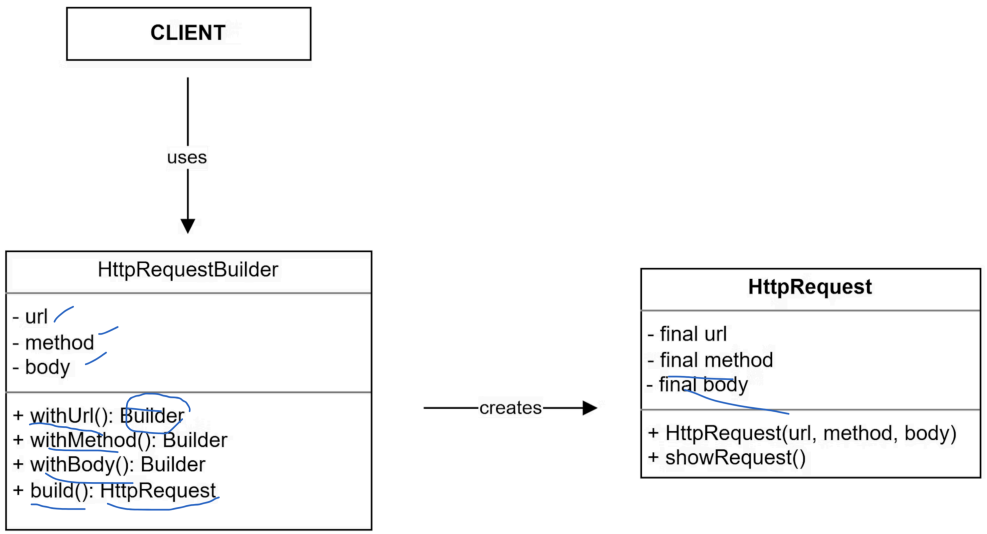
Object is mutable:

```java
// Later, someone else mutates it
req.setUrl("https://evil.com");
```

No creation time validation

Builder Pattern UML Diagram




CLIENT
uses
HttpRequestBuilder
- url
- method
- body
+ withUrl(): Builder
+ withMethod(): Builder
+ withBody(): Builder
+ build(): HttpRequest
creates
HttpRequest
- final url
- final method
- final body
+ HttpRequest(url, method, body)
+ showRequest()

```java
// PRODUCT — immutable, package-private constructor
class HttpRequest {
    private final String url;
    private final String method;
    private final Map<String, String> headers;
    private final String body;
    private final int timeout;

    // Package-private: accessible by HttpRequestBuilder (same package)
    HttpRequest(String url, String method, Map<String, String> headers, String body, int timeout) {
        this.url = url;
        this.method = method;
        this.headers = Collections.unmodifiableMap(headers);
        this.body = body;
        this.timeout = timeout;
    }

    public void showRequest() {
        System.out.println("HttpRequest[" + method + " " + url + ", headers=" + headers
                + ", body=" + (body != null ? body : "none") + ", timeout=" + timeout + "ms]");
    }
}
```

```java
// BUILDER — separate class, fluent (returns this)
class HttpRequestBuilder {
    private String url;
    private String method = "GET";
    private Map<String, String> headers = new HashMap<>();
    private String body;
    private int timeout = 30000;

    public HttpRequestBuilder withUrl(String url) {
      this.url = url;
      return this;
    }

    public HttpRequestBuilder withMethod(String method) {
      this.method = method;
      return this;
    }
```

```
    public HttpRequestBuilder withHeader(String k, String v) { this.headers.put(k, v); return this; }
    public HttpRequestBuilder withBody(String body) { this.body = body; return this; }
    public HttpRequestBuilder withTimeout(int ms) { this.timeout = ms; return this; }

    public HttpRequest build() {
        if (url == null || url.isEmpty()) {
            throw new IllegalStateException("URL is required");
        }
        return new HttpRequest(url, method, headers, body, timeout);
    }
}
```

```
HttpRequest req = new HttpRequestBuilder()
.withUrl("https://api.zerotechdebt.com/users")
.withMethod("POST")
.withHeader("Content-Type", "application/json")
.withBody("{\"name\": \"Akshit Singla\", \"age\": 25}")
.withTimeout(5000)
.build();
```