Prototype Pattern

Bryan Hansen
Director of Software Development

@bh5k

Concepts

Avoids costly creation

Avoids subclassing

Typically doesn’t use “new”

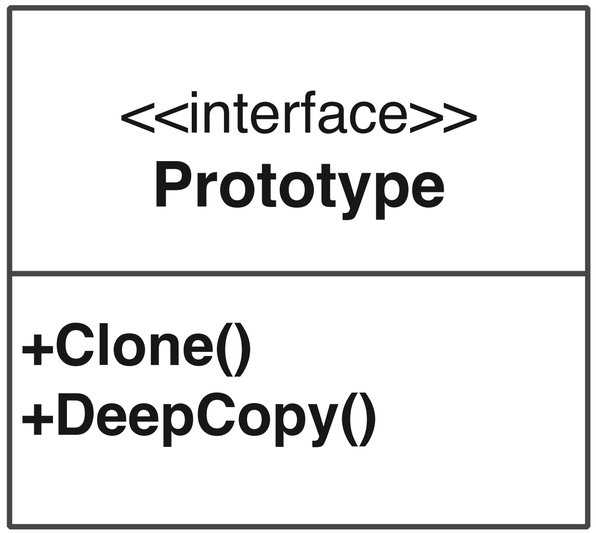
Often utilizes an Interface

Usually implemented with a Registry

Example:

- java.lang.Object#clone()

Design



- Clone / Cloneable
- Avoids keyword “new”
- Although a copy, each instance unique
- Costly construction not handled by client
- Can still utilize constructor parameters
- Shallow VS Deep Copy

Everyday Example - Statement

```
public class Statement implements Cloneable {

	public Statement(String sql, List<String> parameters, Record record) {
		this.sql = sql;
		this.parameters = parameters;
		this.record = record;
	}

	public Statement clone() {
		try {
			return (Statement) super.clone();
		} catch (CloneNotSupportedException e) {}
		return null;
	}
}
```

Demo

Create Prototype

Demonstrate shallow copy

Create with a Registry

Pitfalls

Not always clear when to use

Used with other patterns

- **Registry**

Shallow VS Deep Copy

Contrast

Prototype	Factory
Lighter weight construction	Flexible Objects
Copy Constructor or Clone	Multiple constructors
Shallow or Deep	Concrete Instance
Copy of itself	Fresh Instance

Summary

Guarantee unique instance

Often refactored in

Can help with performance issues

Don’t always jump to a Factory