Prototype Pattern

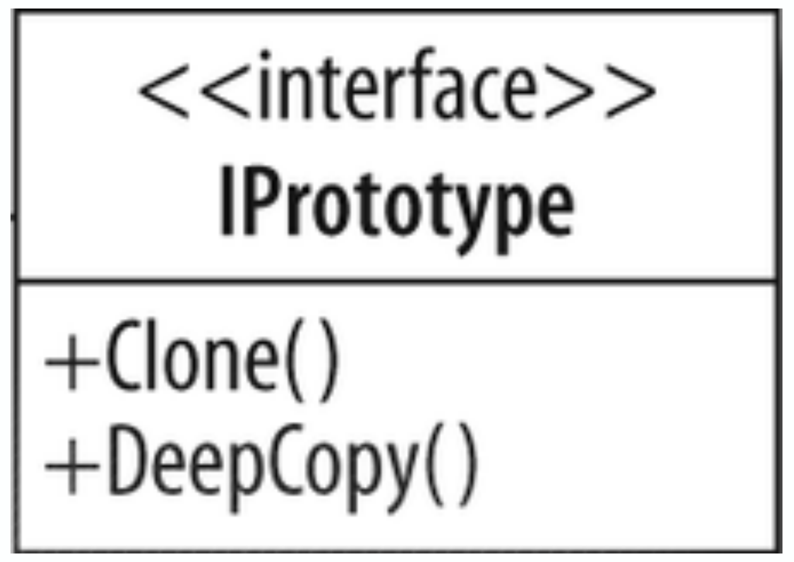
Bryan Hansen

twitter: bh5k | http://www.linkedin.com/in/hansenbryan

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 parameters for construction

Shallow VS Deep Copy

Everyday Example - Object Clone

```
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;
  }
}
```

Exercise Prototype

Create Prototype

Demonstrate shallow copy

Create with a Registry

Pitfalls

- Sometimes not clear when to use
- Used with other patterns
  - Registry
- Shallow VS Deep Copy

Contrast

Prototype

- Lighter weight construction
  - Copy Constructor or Clone
- Shallow or Deep
- Copy of itself

Factory

- Flexible Objects
  - Multiple constructors
- Concrete Instance
- Fresh Instance

Prototype Summary

- Guarantee unique instance
- Often refactored in
- Can help with performance issues
- Don’t always jump to a Factory