Prototype

Linda Marshall and Vreda Pieterse

Department of Computer Science
University of Pretoria

4 and 5 August 2014

Overview

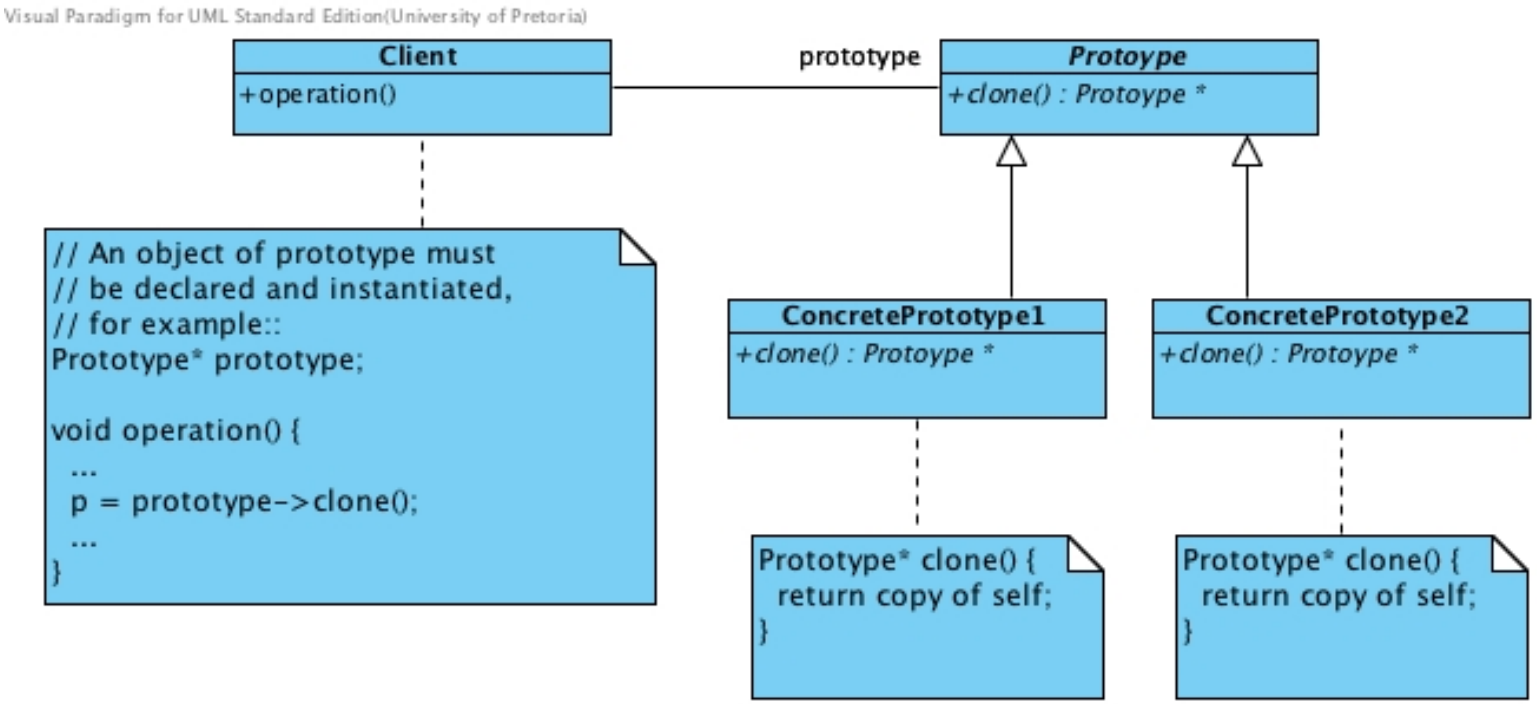
1. Identification
2. Structure
3. Participants
4. Related Patterns
5. Example

Name and Classification:

Prototype (Object Creational)

Intent:

"Specify the kinds of objects to create using a prototypical instance, and create new objects by copying this prototype" GoF(117)


Visual Paradigm for UML Standard Edition(University of Pretoria)
Client
+operation()
prototype
Protoype
+clone() : Protoype *
// An object of prototype must
// be declared and instantiated,
// for example::
Prototype* prototype;
void operation() {
...
p = prototype->clone();
...
}
ConcretePrototype1
+clone() : Protoype *
ConcretePrototype2
+clone() : Protoype *
Prototype* clone() {
return copy of self;
}
Prototype* clone() {
return copy of self;
}

- Gives flexible alternatives to inheritance
- The client creates a prototype and each time it requires a new object, the prototype is asked to clone itself

Prototype

- defines an interface for cloning

ConcretePrototypeN

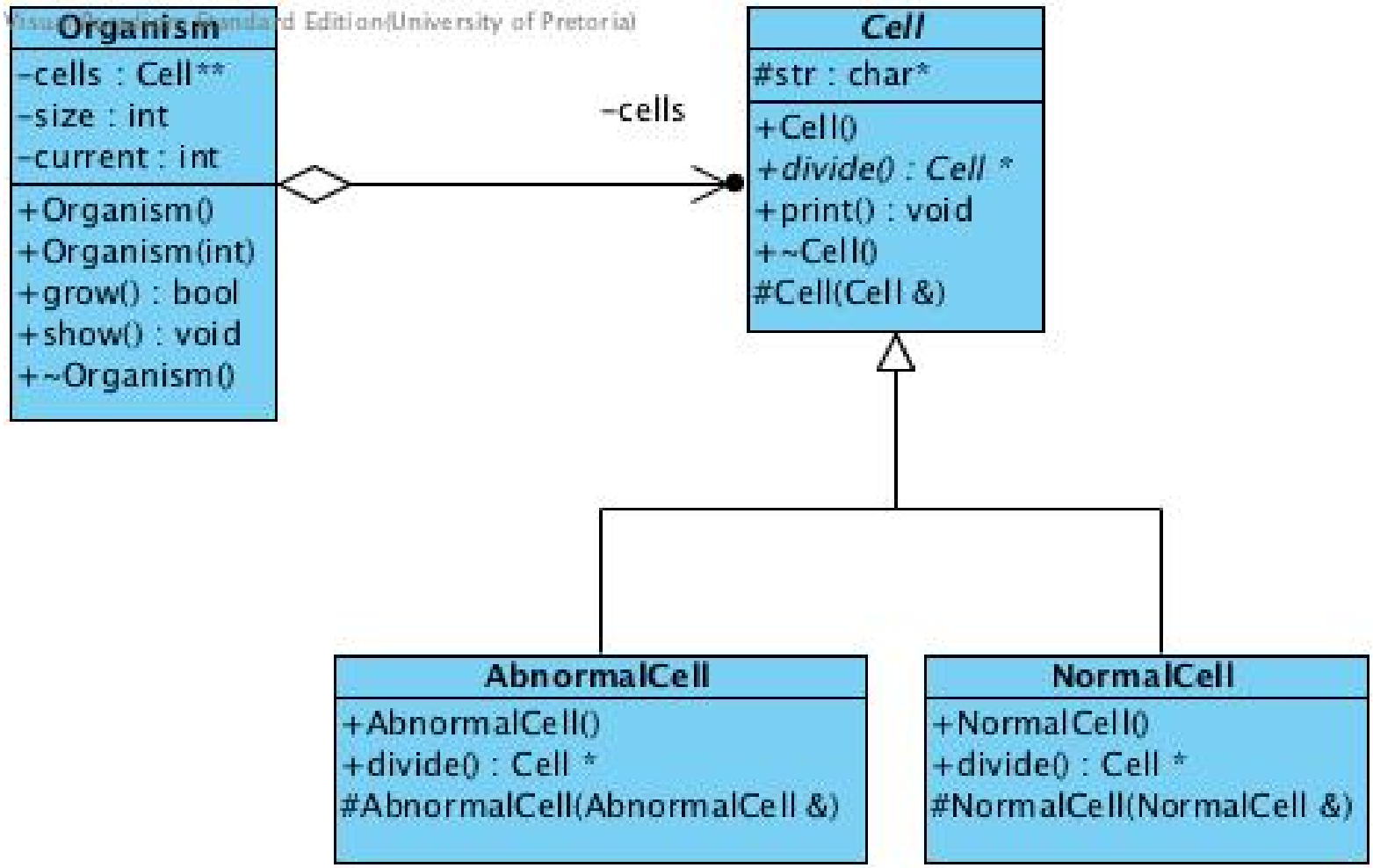
- implementation of operation for cloning

Client

- asks the prototype to clone so that a new object can be created

Related Patterns

- **Abstract Factory** (87): Competes in producing product, however an AF may use Prototype to clone product.
- **Composite and Decorator** (163 and 175): Make use of the prototype for nodes cloning.


Organism
-cells : Cell**
-size : int
-current : int
+Organism()
+Organism(int)
+grow() : bool
+show() : void
+~Organism()
-cells
Cell
#str : char*
+Cell()
+divide() : Cell *
+print() : void
+~Cell()
#Cell(Cell &)
AbnormalCell
+AbnormalCell()
+divide() : Cell *
#AbnormalCell(AbnormalCell &)
NormalCell
+NormalCell()
+divide() : Cell *
#NormalCell(NormalCell &)