Prototype

Linda Marshall

Department of Computer Science
University of Pretoria

27 August 2021

Name and Classification:

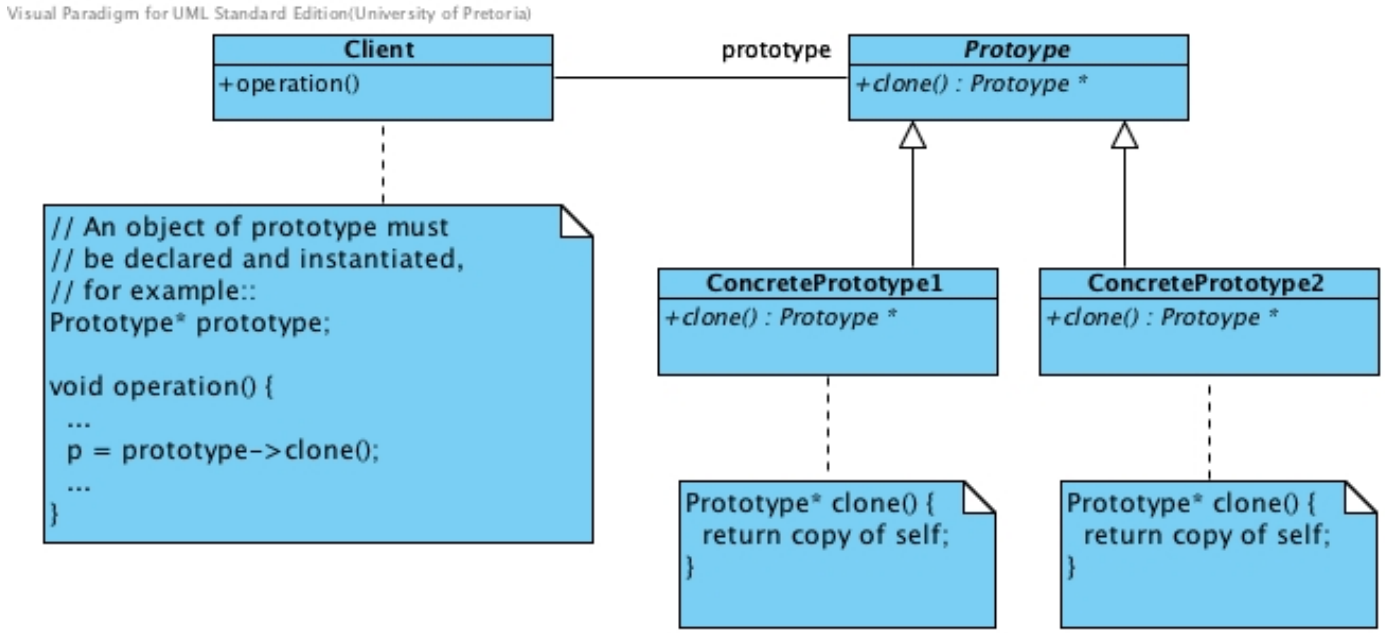
Prototype

(Object Creational)

Intent:

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

"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
- The state of this clone may be that of the current object, or that of the initial object.

Prototype

- defines an interface for cloning

ConcretePrototypeN

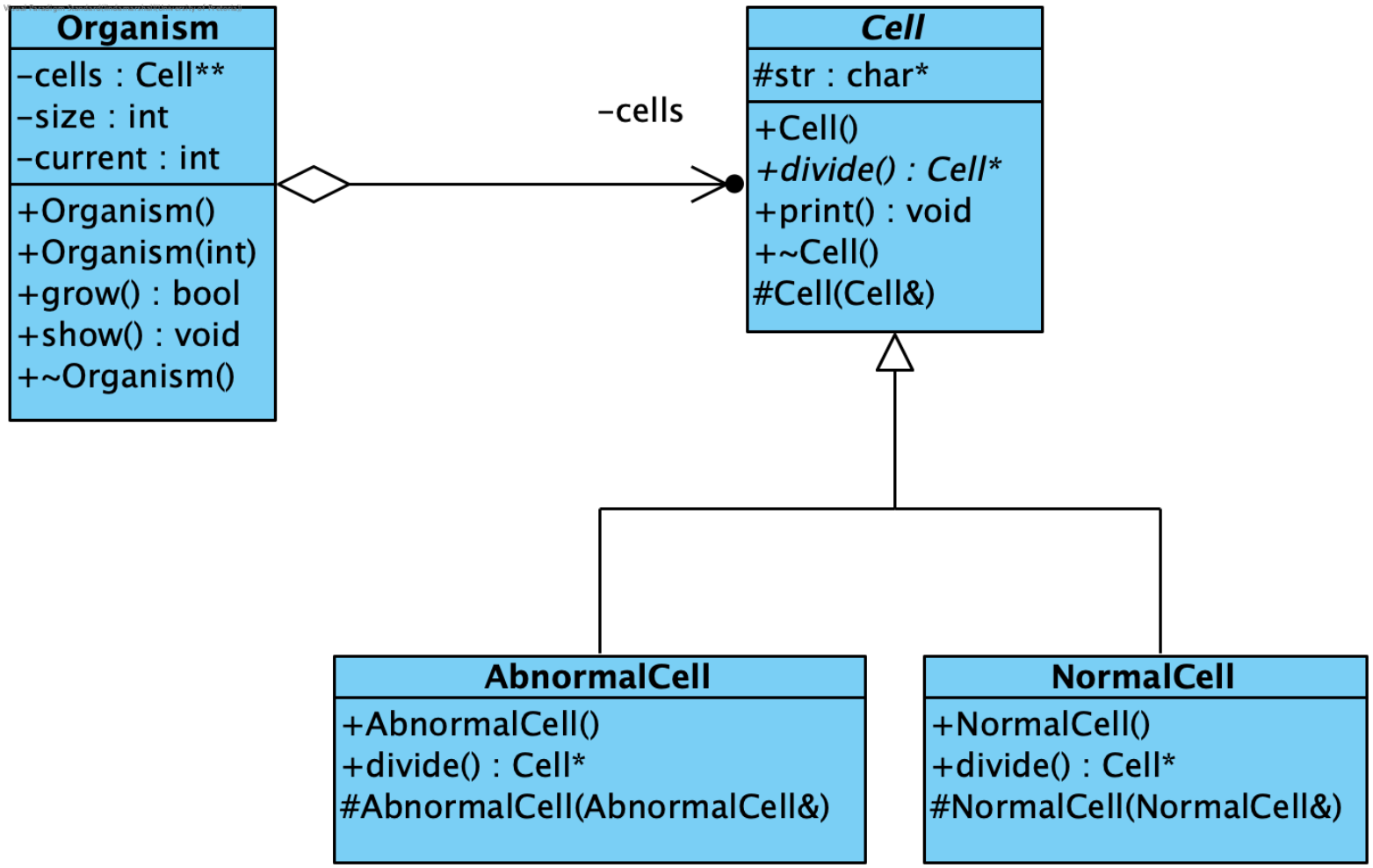
- implementation of operation for cloning

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 node 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&)

The company, COS214 coders, you have been working for has been using the Memo Sender code since 2011. At that time SMS's were too expensive to send, so the boss only sent out emails to the staff. With wifi being readily available and many of the staff using other Apps on their phones, it is time to re-evaluate the Memo Sender code and make it more adaptable.

Instead of maintaining a list of email addresses, the phone number and preferred method for the staff member if captured. When the boss wants to send a message to all staff, sendBatch is called with a list of staff objects. Memo's `send` operation needs to differentiate between the different types of Apps on which the memo can be sent.

Adapt the design of the memo sender to provide for the new functionality. Limit changes to the existing interfaces. Other than sending a list of staff objects through, the main program may not change.
Hint: The main change will be in `MemoSender::sendBatch` and in the Memo hierarchy.