







A / Design Patterns / Creational patterns / Prototype

Prototype in C++



Back to **Prototype** description

Prototype design pattern demo

Discussion. Image base class provides the mechanism for storing, finding, and cloning the prototype for all derived classes. Each derived class specifies a private static data member whose initialization "registers" a prototype of itself with the base class. When the client asks for a "clone" of a certain type, the base class finds the prototype and calls clone() on the correct derived class.

```
#include <iostream>
enum imageType
  LSAT, SPOT
};
class Image
{
  public:
    virtual void draw() = 0;
    static Image *findAndClone(imageType);
  protected:
    virtual imageType returnType() = 0;
    virtual Image *clone() = 0;
    // As each subclass of Image is declared, it registers its prototype
    static void addPrototype(Image *image)
    {
        _prototypes[_nextSlot++] = image;
    }
  private:
    // addPrototype() saves each registered prototype here
    static Image *_prototypes[10];
    static int _nextSlot;
};
Image *Image::_prototypes[];
int Image::_nextSlot;
// Client calls this public static member function when it needs an instance
// of an Image subclass
Image *Image::findAndClone(imageType type)
  for (int i = 0; i < _nextSlot; i++)</pre>
    if (_prototypes[i]->returnType() == type)
      return _prototypes[i]->clone();
  return NULL;
}
class LandSatImage: public Image
  public:
    imageType returnType()
        return LSAT;
    void draw()
```

```
std::cout << "LandSatImage::draw " << _id << std::endl;</pre>
    }
    // When clone() is called, call the one-a ment ctor with a dummy arg
    Image *clone()
        return new LandSatImage(1);
    }
  protected:
    // This is only called from clone()
    LandSatImage(int dummy)
        _id = _count++;
    }
  private:
    // Mechanism for initializing an Image subclass - this causes the
    // default ctor to be called, which registers the subclass's prototype
    static LandSatImage _landSatImage;
    // This is only called when the private static data member is initiated igcup
    LandSatImage()
    {
        addPrototype(this);
    // Nominal "state" per instance mechanism
    int _id;
    static int _count;
};
// Register the subclass's prototype
LandSatImage LandSatImage;
// Initialize the "state" per instance mechanism
int LandSatImage::_count = 1;
class SpotImage: public Image
  public:
    imageType returnType()
        return SPOT;
    void draw()
        std::cout << "SpotImage::draw " << _id << std::endl;</pre>
    Image *clone()
        return new SpotImage(1);
    }
  protected:
    SpotImage(int dummy)
```

```
{
        _id = _count++;
    }
  private:
    SpotImage()
        addPrototype(this);
    static SpotImage _spotImage;
    int _id;
    static int _count;
};
SpotImage SpotImage::_spotImage;
int SpotImage::_count = 1;
// Simulated stream of creation requests
const int NUM_IMAGES = 8;
imageType input[NUM_IMAGES] =
  LSAT, LSAT, LSAT, SPOT, LSAT, SPOT, SPOT, LSAT
};
int main()
  Image *images[NUM_IMAGES];
  // Given an image type, find the right prototype, and return a clone
  for (int i = 0; i < NUM_IMAGES; i++)</pre>
    images[i] = Image::findAndClone(input[i]);
  // Demonstrate that correct image objects have been cloned
  for (int i = 0; i < NUM_IMAGES; i++)</pre>
    images[i]->draw();
  // Free the dynamic memory
  for (int i = 0; i < NUM_IMAGES; i++)</pre>
    delete images[i];
}
```

Output

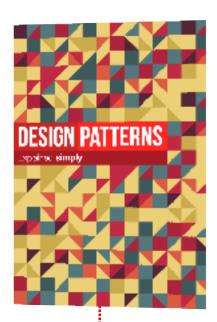
LandSatImage::draw 1
LandSatImage::draw 2
LandSatImage::draw 3
SpotImage::draw 1
LandSatImage::draw 4
SpotImage::draw 2
SpotImage::draw 3
LandSatImage::draw 5

Read next

This article is taken from our book **Design Patterns Explained Simply**.

All of the design patterns are compiled there. The book is written in clear, simple language that makes it easy to read and understand (just like this article).

We distribute it in PDF & EPUB formats so you can get it onto your iPad, Kindle, or other portable device immediately after your purchase.





Learn more

Code examples

Java	Prototype in Java	Prototype in Java
C++	Prototype in C++: Before and after	Prototype in C++
PHP	Prototype in PHP	Prototype in PHP

Design Patterns My account
AntiPatterns Forum
Refactoring Contact us

UML About us

© 2007-2018 SourceMaking.com All rights reserved.

Terms / Privacy policy