Design patterns c++

Prototype (clone): es clonar la clase pero solo Podemos modificar los valores pero no los metodos porque cada copia tiene su heap y stack (memoria para variables y apuntadores) pero los metodos se guardan compartidos

Downcasting:

Suponiendo que tienes una clase móviles, entonces tienes varios hijos, Tablet, smartphone y smartwatch, lo que se hace es que por medio de clases abstractas en el padre, solo se declaran métodos y ya en los hijos se implementan los métodos (entonces vas de arriba para abajo)

Upcasting:

Es hacer un cast desde los hijos hasta el padre

Interfaces:

* No tienen implementaciones
* Obligo a que los hijos implementen los métodos
* Se definen cosas constantes y estáticos
* Todo es publico

Las variables que están en los métodos se conocen como variables temporales porque saliendo del método se borra la pila **(dangling pointer)**

*Int do(){*

*Int c= 5; (esta es la variable que se borra con el stack)*

*Return &c;*

*}*

static\_cast: permite lanzar excepciones pero en tiempo de compilacion

Dynamic\_cast: permite lanzar excepciones en tiempo de ejecucion

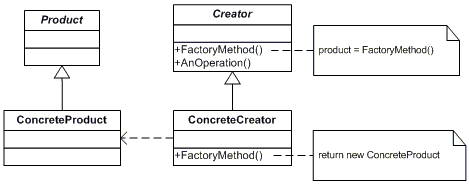
Uso:

dynamic\_cast<ref.>(algo)

Para poder usar el constructor del padre, se utiliza using namespace padre

Problema: solo detecta al padre (existe la gerarquia, pero solo detecta al padre directo, no al abuelo)

Factory method: basicamente son dos clases, una clase que crea y otra abstracta que la implementa



Tenemos la clase Creator que llama a concrete creator para los metodos y hace una instancia de concrete product