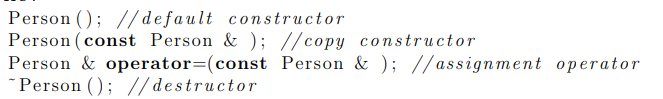
**Background**

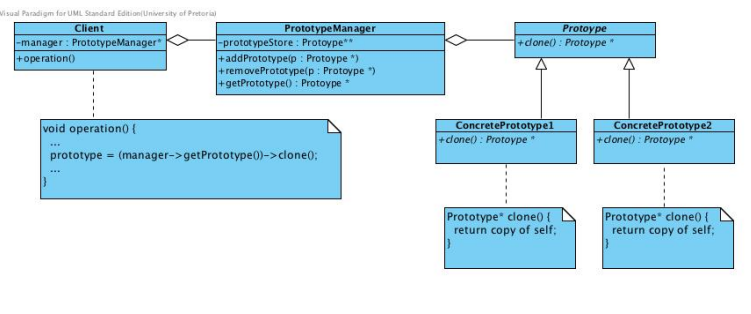
* The whole idea is caching
  + Tradeoff -> use of more memory
* Eg:
  + Game caching
  + Dynamic web-pages
  + Spreadsheets that use complex computations that are reused
  + Generating a GUI
  + Building a game
  + Applying a variety of analyses on the same result set from a database
* Copying
  + Shallow copy (only for primitives)
  + Deep copy (for primitives and pointers)
  + Copy constructor
    - One parameter (object of the class)
    - Takes a reference to a const parameter



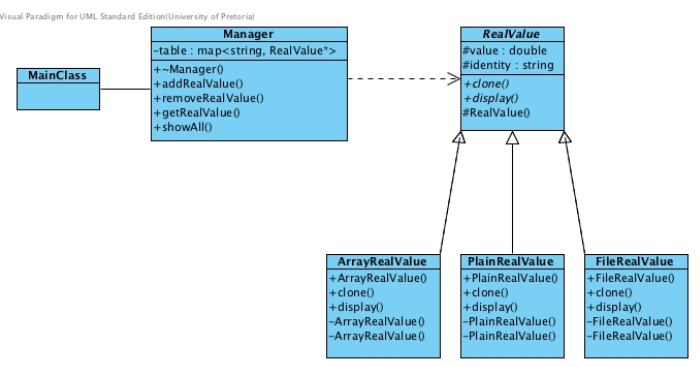
**Prototype**

**Diagram

Description automatically generated**

* Problem
  + Constructor is time consuming
  + Very hard implementing the clone()
* Participants
  + Prototype
  + ConcretePrototype
  + Client
* Alternate structure
* ****
* Participants
  + Prototype
  + ConcretePrototype
  + PrototypeManager
  + Client
* Where used?
  + When it is known that the constructor is time consuming
  + When only small variations will be needed between objects
  + When you’ll use copies
* Pros:
  + Client program will be more generic
  + System structure will be more streamlined
  + prototype manager is implemented, it is possible to instantiate new types at runtime simply by adding more prototypes to the prototype manager.
  + More convenient
* Misconceptions
  + Using a copyconstructor != prototype
  + Cloning objects != prototype
* Retlated patterns
  + Factory Method
  + Abstract Factory
  + Acstrand Factory and Builder
  + Composite and Decorator
  + Singleton and Abstrract Factory
  + Singleton, Memento and Flyweight

**Example**

****

**Graphical user interface, text, application

Description automatically generated**