COS 214 Project

sudo apt-get marks

Members:

u22497082 - Ethan Vletter

u22652974 - Carinda Smith

u23632730 - Janri du Toit

u23577674 - Ethan Wilke

u23656175 - James Neale

u23708833 - Megan Pretorius

u23590883 - Marco Paximadis

Task 2.1 Identify the Functional Requirements

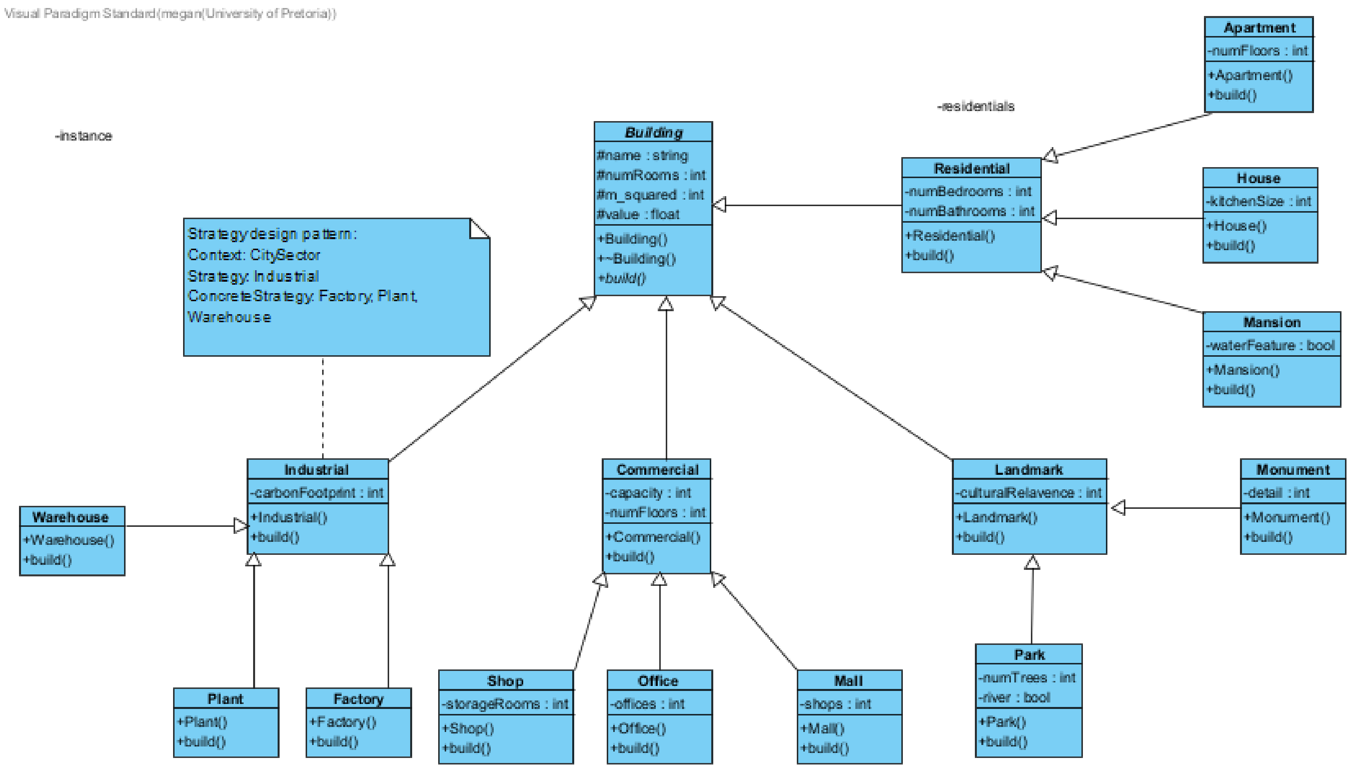
City Management:

- Create and manage different types of buildings.
  Building
  - Residential
    - Apartment
    - House
    - Mansion
  - Landmark
    - Monument
    - Park
  - Commercial
    - Mall
    - Office
    - Shop
  - Industrial
    - Warehouse
    - Plant
    - Factory

Design Pattern for Industrial

Strategy design pattern

- Context : CitySector
- Strategy : Industrial
- ConcretStrategy : Factory, Plant and Warehouse

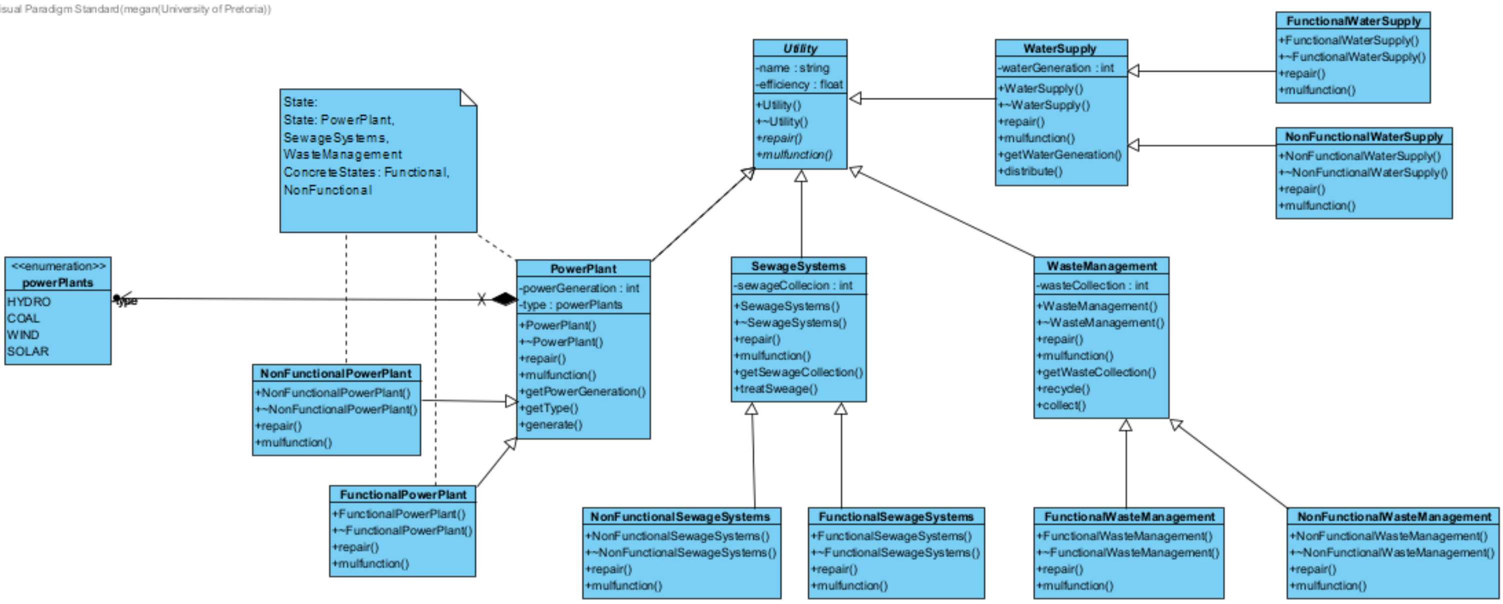


- <u>Manage Utilities</u>
    - Powerplants : generate electricity to supply to city.
        - Functional and Non-functional
    - Water Supply : ensure steady distribution of water.
        - Functional and Non-functional
    - Waste Management : handle waste removal and recycling.
        - Functional and Non-functional
    - Sewage Systems : manage sewage disposal and treatments.
        - Functional and Non-functional

<u>Design Pattern for Utilities</u>

State design pattern

- State : Powerplant, SewageSystems, WasteManagement, WaterSupply.
- Concrete State : Functional and Non-functional

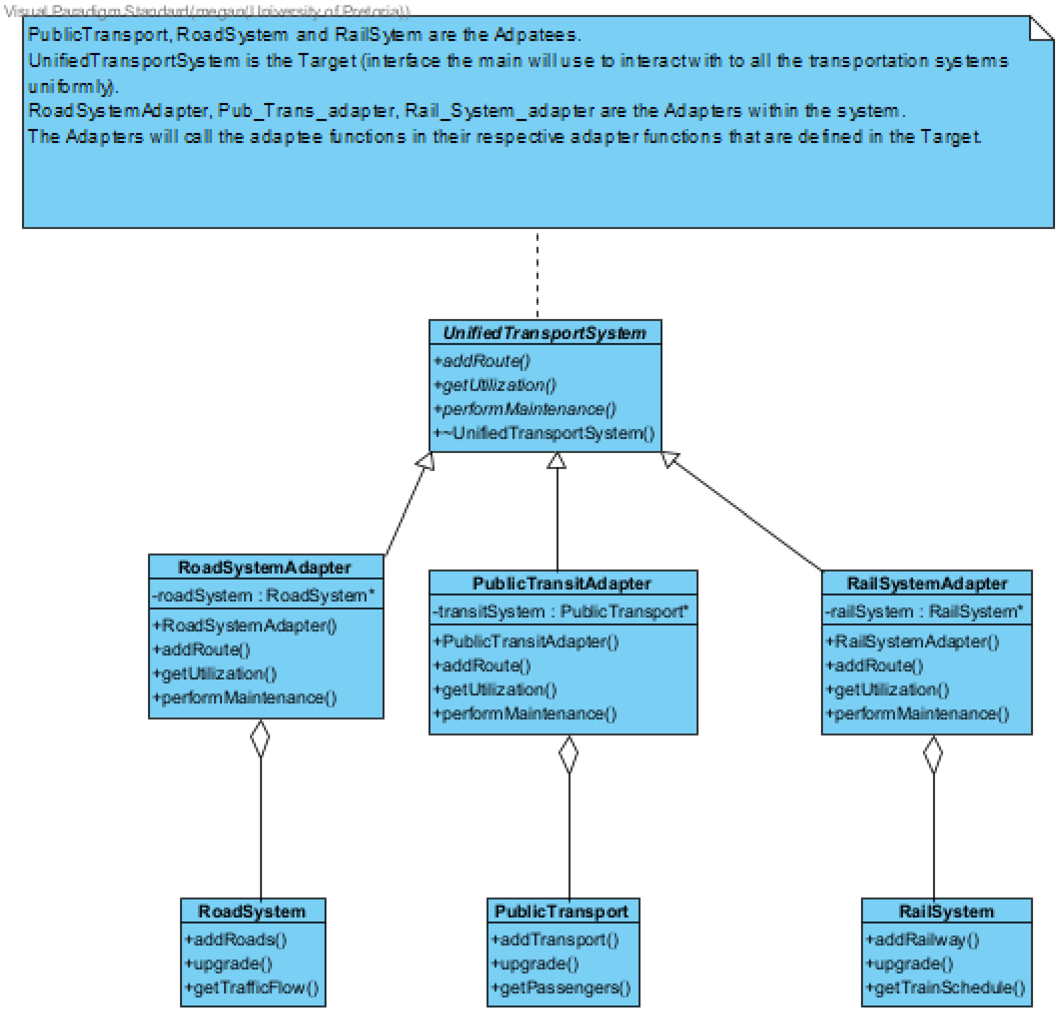


- <u>Handle Transportation system.</u>
  - Roads: Basic infrastructure for vehicle movement.
  - Public Transit: Buses, taxis, etc.
  - Trains: Freight and passenger trains.
  - Airports: Facilitate air travel and cargo transport.

<u>Design Pattern for Transport System</u>

Adapter design pattern

- Adapters : RoadSystemAdapter, PublicTransitAdapter and RailSystemAdapter
- Target : UnifiedTransportSystem
  The interface the main will use to interact with to all the transportation systems uniformally.
- Adaptees : RoadSystem, PublicTransit and RailSystem



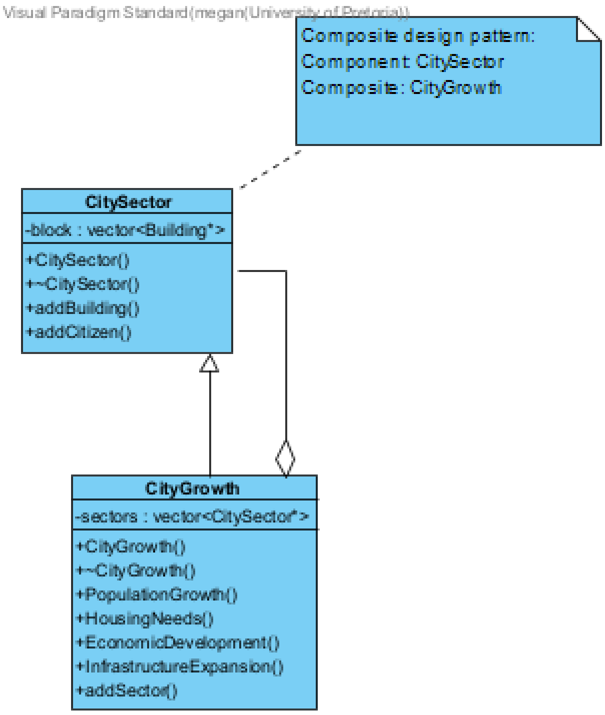
PublicTransport, RoadSystem and RailSytem are the Adpatees.
UnifiedTransportSystem is the Target (interface the main will use to interact with to all the transportation systems uniformly).
RoadSystemAdapter, Pub_Trans_adapter, Rail_System_adapter are the Adapters within the system.
The Adapters will call the adaptee functions in their respective adapter functions that are defined in the Target.

- City creation and growth

Composite Design Pattern for Infrastructure

Allows for treating individual objects and compositions uniformly and enables creation of complex structures, like city district, composed of multiple infrastructure elements.

- Component : CitySector
- Composite : CityGrowth
  - Population growth
  - Housing needs
  - Economic development
  - Infrastructure expansion



Citizen Management:

- Handle population growth. (CityGrowth from Composite)
- Manage citizen satisfaction levels.
- Handle employment and services.

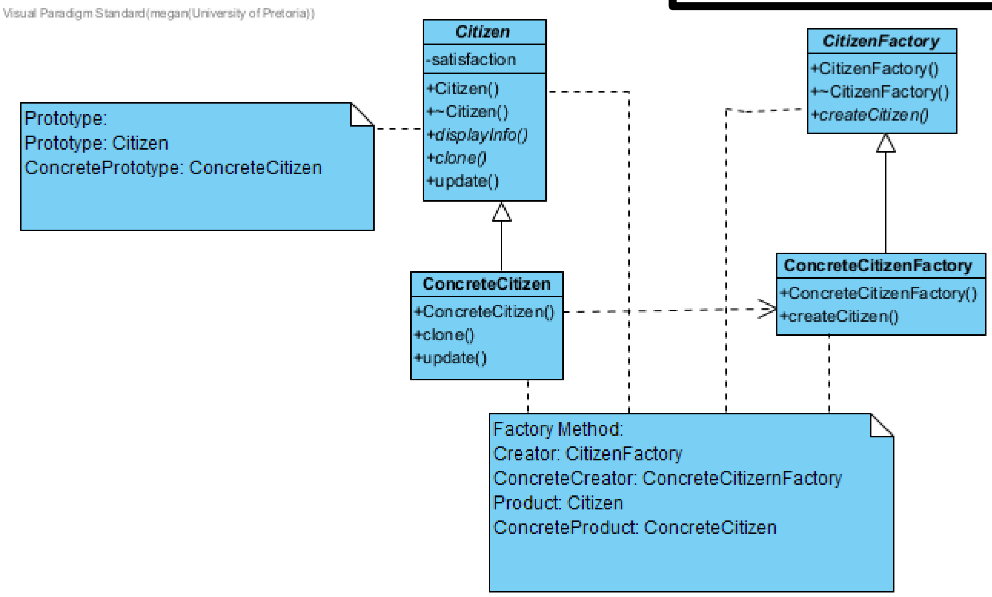
Design Patterns for Citizen

Factory Method

- Creator : CitizenFactory
- ConcreteCreator : ConcreteCitizenFactory (clone(), update())
- Product : Citizen (information())
- ConcreteProduct : ConcreteCitizen (createCitizen ())

Prototype

- Prototype : Citizen (has satisfaction attribute)
- ConcretePrototype : ConcreteCitizen.



Government: "owns everything"

- Taxation: Setting and collecting taxes from citizens and businesses.
- City Budget: Budget allocation for various city services and projects.
  - Budget is a generalisation of Resources.
- Policies: Implementing laws and regulations that impact city dynamics.
- Public Services: Managing healthcare, education, law enforcement, and more.

Design Patterns for Government

Observer design pattern

- Subject : Government
- Observer : Citizen
- ConcreteObserver : ConcreteCitizen

Tax system design Pattern:

Setting and collecting taxes from citizen and business.

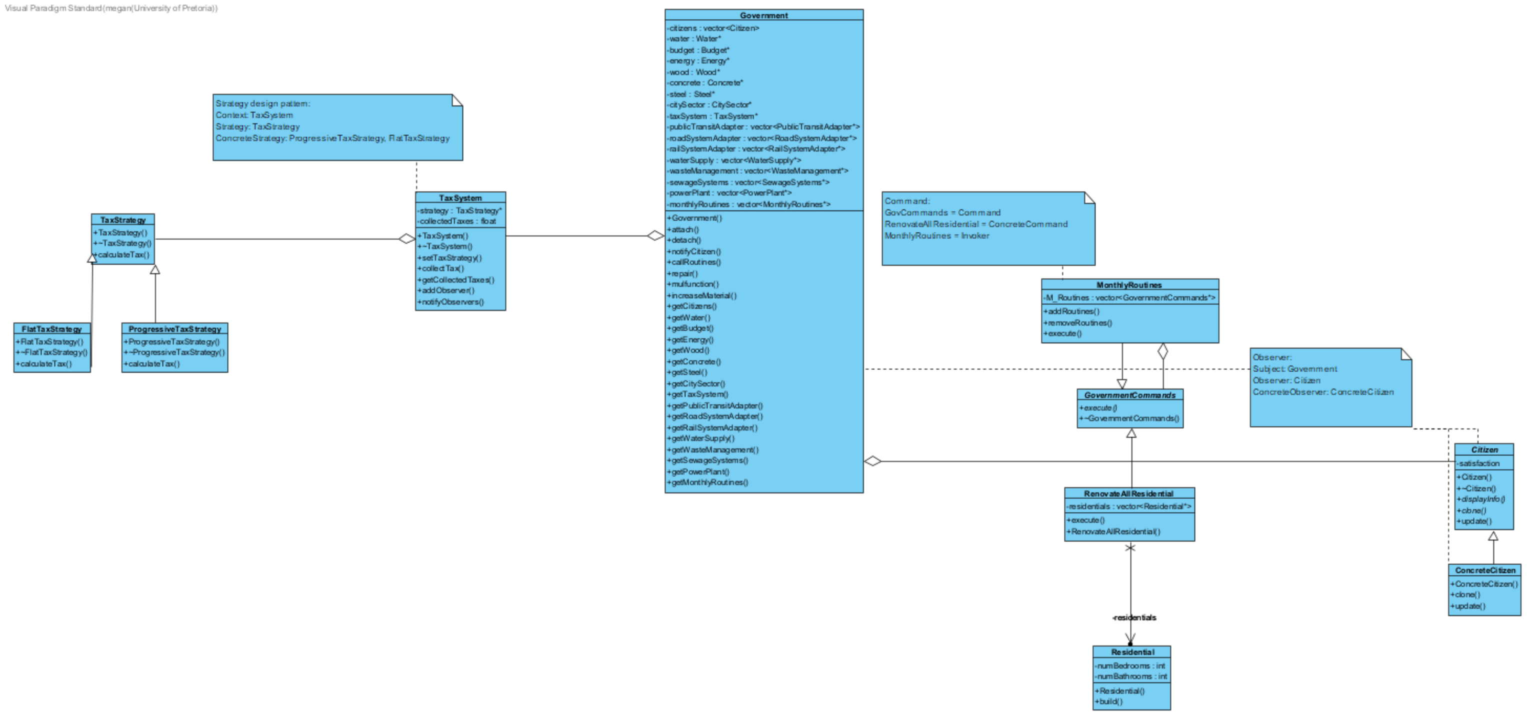
Strategy

- Context : TaxSystem
- Strategy : TaxStrategy
- ConcreteStrategy : ProgressTaxStrategy, FlatTaxStrategy

Design Patterns for Residential Renovation:

Command

- Command : GovernmentCommand (execute())
- ConcreteCommand : RenovateAllResidential (execute())
- Invoker : MonthlyRoutine (add(), remove(), execute())



Resources: Managing resources

- Materials

<u>Design Patterns for Materials:</u>

Template Method

- AbstarctClass : Materials
- ConcreteClass : Concrete, Wood, Steel

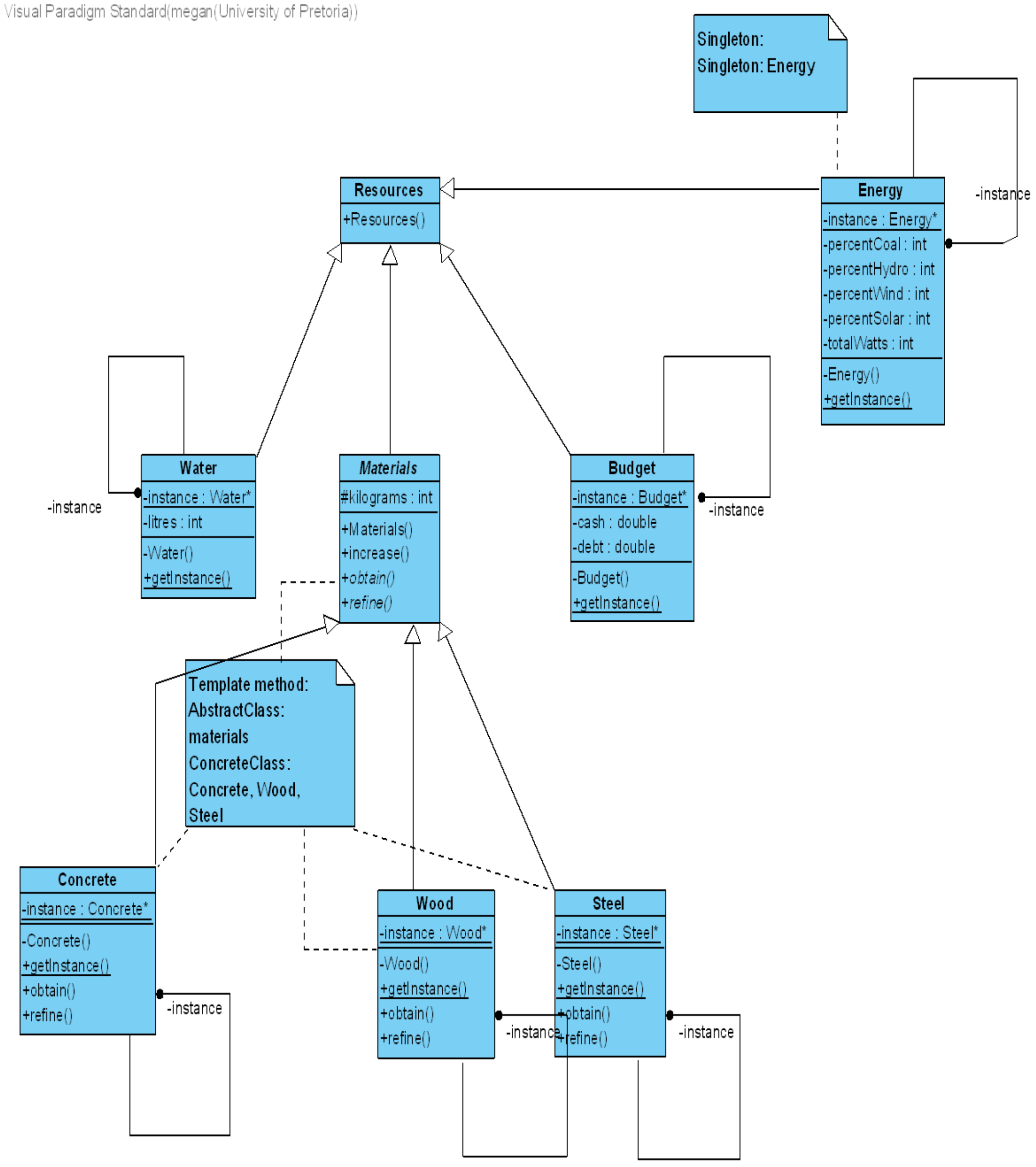
- Energy

<u>Design Patterns for Energy:</u>

Singleton

- Singleton : Energy

- Water
- Budget



Visual Paradigm Standard(megan(University of Pretoria))
Singleton:
Singleton: Energy
Resources
+Resources()
Energy
-instance : Energy*
-percentCoal : int
-percentHydro : int
-percentWind : int
-percentSolar : int
-totalWatts : int
-Energy()
+getInstance()
-instance
Water
-instance : Water*
-litres : int
-Water()
+getInstance()
-instance
Materials
#kilograms : int
+Materials()
+increase()
+obtain()
+refine()
Budget
-instance : Budget*
-cash : double
-debt : double
-Budget()
+getInstance()
-instance
Template method:
AbstractClass:
materials
ConcreteClass:
Concrete, Wood,
Steel
Concrete
-instance : Concrete*
-Concrete()
+getInstance()
+obtain()
+refine()
-instance
Wood
-instance : Wood*
-Wood()
+getInstance()
+obtain()
+refine()
-instance
Steel
-instance : Steel*
-Steel()
+getInstance()
+obtain()
+refine()
-instance