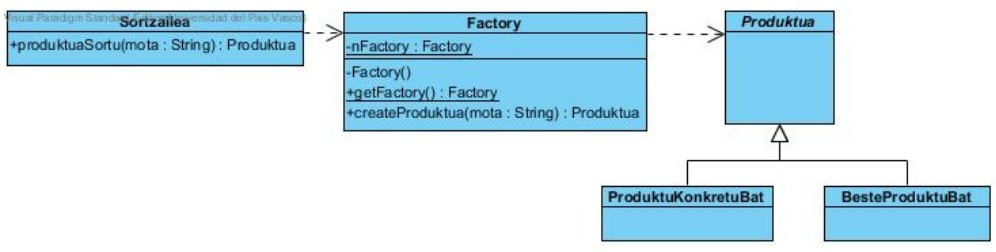
# **SORTZAILEAK**

**Factory**: objektuak sortzeko interfazea definitu, baina, azpiklaseen esku klaseen instantziazioaren kudeaketa.

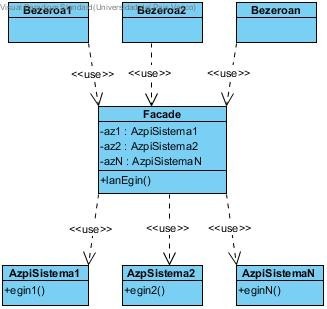


**Ezaugarriak**

* Objektuen sorrera faktorian enkapsulatu
* Objektuen sorrera (faktoria) eta objektuekin kudeaketa (pizzeria) banatu
* Pizza mota berria sortzeko
  + Klase abstraktua hedatzeko klasea sortu
  + Faktorian bi lerro gehitu
* Objektuen sorrera kontrolatu
* Mantenketa eta hedatzea erraztu

# **EGITURAZKOAK**

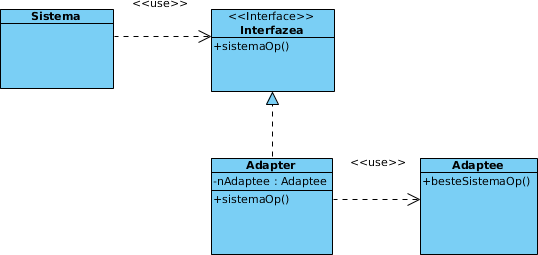
**Facade**: azpisistema bateko interfaze-multzo bati interfaze bateratua eman; hots, maila altuko interfazea, azpisistema erabilterrazagoa izan dadin.



**Ezaugarriak**

* Bezeroak azpisistematik isolatu
* Bezero/azpisistema akoplamendu ahula.
* Azpisistemak bezeroarentzat eskuragarri, hala nahi izanez gero
* Sistema geruzatan banatu
* Kontuan izan, bezeroek azpisistema desberdinak erabiliz gero, facade desberdinak.

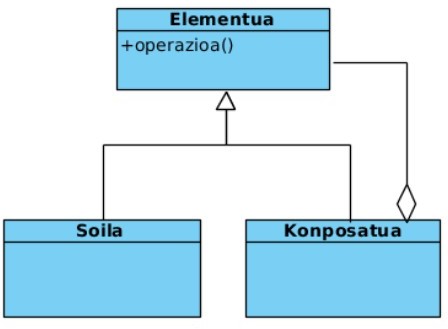
**Adapter**: interfaze bateraezinei elkarrekin lan egiteko aukera eman. Gure sistemako funtzionalitate baliokidea dauka bezeroaren interfazeak, baina interfaze desberdina; bitartekari lana egiten du.



**Ezaugarriak**

* Berrerabilgarritasuna hobetu
* Hedapena erraztu
* Adapterrak interfazea enkapsulatu
  + Bezeroa interfazetik desakoplatu
  + Interfazea aldatuta, bezeroak ez du ikusten
* Adaptee-a ez da ukitzen

**Composite**: aplikazioan “banakako” objektuak eta “konposatuak” modu berean erabiltzea ahalbidetzen du. Zuhaitz motako hierarkietan txertatzen ditu objektu horiek.

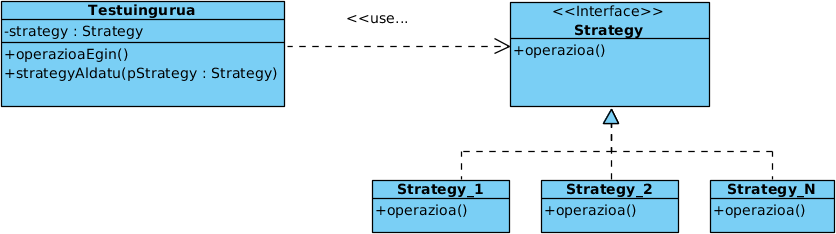


**Ezaugarriak**

* Banakako elementuak (hostoak) eta konposatuak (nodoak) zuhaitz egitura berean txertatu.
  + “Part-whole” hierarkiak
  + Konposatuek elementu soilez zein konposatuez osatuta egon daitezke
* Objektu guztiek interfaze bera
  + Nodo zein hosto, era berean tratatu.

# **PORTAERAZKOAK**

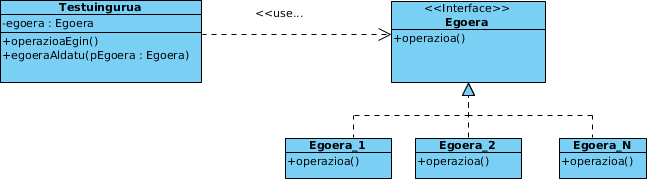
**Strategy**: funtzionalitate beraren portaera (estrategia) desberdinak definitzen ditu, eta testuinguruaren arabera estrategia hautatzea ahalbidetzen du. Gainera, estrategia “run-time”-ean aldatzeko aukera ematen du.



**Ezaugarriak**

* Testuingurua eta portaerak banatu
* Berrerabilpena hobetu
* Algoritmo familiak definitu
* Inplementazio aukera desberdinak
* Bezeroak Strategy desberdinak ezagutu eta aukeratzen ditu

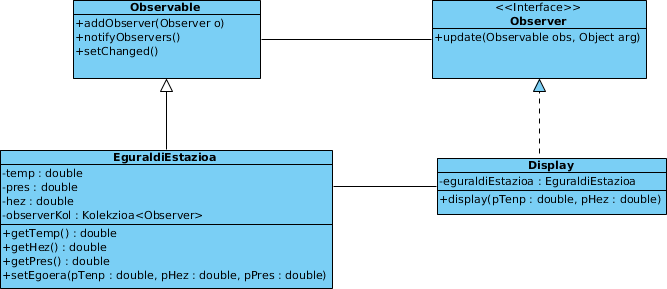
**State**: objektu baten barne egoera aldatzean, bere portaera aldatzea ahalbidetzen dio.



**Ezaugarriak**

* Egoeraren arabera, portaera desberdina
* Egoera bakoitzeko portaera enkapsulatu
* Egoeren arteko transizioak esplizituak
* Hedagarria
* Bezeroak egoeren inguruko informazio gutxi edo ezer

**Observer**: objektuen arteko “one-to-many” dependentziak definitu; objetu batek bere egoera aldatzen duenean, bere menpeko guztiei jakinaraziko die.



**Ezaugarriak**

* Akoplamendu soltea (loose-coupling); observable-k ez daki observer-aren mota.
* Entzuleei jakinarazpenak bidali
* Hedagarria