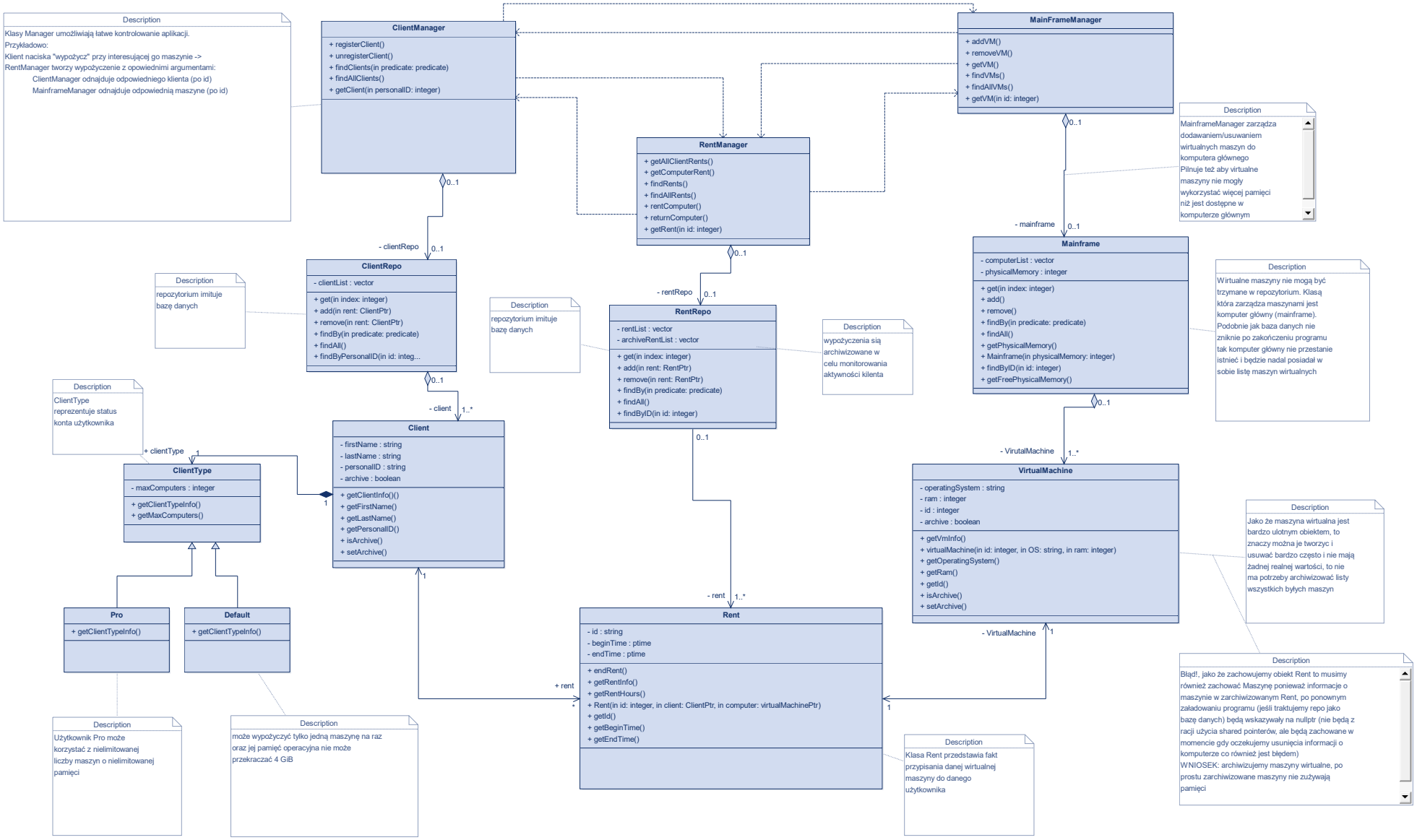
Description
Klasy Manager umożliwiają łatwe kontrolowanie aplikacji.
Przykładowo:
Klient naciska "wypożycz" przy interesującej go maszynie ->
RentManager tworzy wypożyczenie z opowiednimi argumentami:
ClientManager odnajduje odpowiedniego klienta (po id)
MainframeManager odnajduje odpowiednią maszyne (po id)
ClientManager
+ registerClient()
+ unregisterClient()
+ findClients(in predicate: predicate)
+ findAllClients()
+ getClient(in personalID: integer)
MainFrameManager
+ addVM()
+ removeVM()
+ getVM()
+ findVMs()
+ findAllVMs()
+ getVM(in id: integer)
RentManager
+ getAllClientRents()
+ getComputerRent()
+ findRents()
+ findAllRents()
+ rentComputer()
+ returnComputer()
+ getRent(in id: integer)
Description
MainframeManager zarządza dodawaniem/usuwaniem wirtualnych maszyn do komputera głównego
Pilnuje też aby virtualne maszyny nie mogły wykorzystać więcej pamięci niż jest dostępne w komputerze głównym
0..1
- clientRepo
0..1
- mainframe
0..1
ClientRepo
- clientList : vector
+ get(in index: integer)
+ add(in rent: ClientPtr)
+ remove(in rent: ClientPtr)
+ findBy(in predicate: predicate)
+ findAll()
+ findByPersonalID(in id: integ...
Description
repozytorium imituje bazę danych
Mainframe
- computerList : vector
- physicalMemory : integer
+ get(in index: integer)
+ add()
+ remove()
+ findBy(in predicate: predicate)
+ findAll()
+ getPhysicalMemory()
+ Mainframe(in physicalMemory: integer)
+ findByID(in id: integer)
+ getFreePhysicalMemory()
Description
Wirtualne maszyny nie mogą być trzymane w repozytorium. Klasą która zarządza maszynami jest komputer główny (mainframe). Podobnie jak baza danych nie zniknie po zakończeniu programu tak komputer główny nie przestanie istnieć i będzie nadal posiadał w sobie listę maszyn wirtualnych
- rentRepo
0..1
RentRepo
- rentList : vector
- archiveRentList : vector
+ get(in index: integer)
+ add(in rent: RentPtr)
+ remove(in rent: RentPtr)
+ findBy(in predicate: predicate)
+ findAll()
+ findByID(in id: integer)
Description
repozytorium imituje bazę danych
Description
wypożyczenia są archiwizowane w celu monitorowania aktywności kilenta
Description
ClientType reprezentuje status konta użytkownika
- client
1..*
+ clientType
1
Client
- firstName : string
- lastName : string
- personalID : string
- archive : boolean
+ getClientInfo()()
+ getFirstName()
+ getLastName()
+ getPersonalID()
+ isArchive()
+ setArchive()
ClientType
- maxComputers : integer
+ getClientTypeInfo()
+ getMaxComputers()
- VirutalMachine
1..*
VirtualMachine
- operatingSystem : string
- ram : integer
- id : integer
- archive : boolean
+ getVmInfo()
+ virtualMachine(in id: integer, in OS: string, in ram: integer)
+ getOperatingSystem()
+ getRam()
+ getId()
+ isArchive()
+ setArchive()
Description
Jako że maszyna wirtualna jest bardzo ulotnym obiektem, to znaczy można je tworzyc i usuwać bardzo często i nie mają żadnej realnej wartości, to nie ma potrzeby archiwizować listy wszystkich byłych maszyn
Pro
+ getClientTypeInfo()
Default
+ getClientTypeInfo()
- rent
1..*
Rent
- id : string
- beginTime : ptime
- endTime : ptime
+ endRent()
+ getRentInfo()
+ getRentHours()
+ Rent(in id: integer, in client: ClientPtr, in computer: virtualMachinePtr)
+ getId()
+ getBeginTime()
+ getEndTime()
+ rent
*
- VirtualMachine
1
Description
Błąd!, jako że zachowujemy obiekt Rent to musimy również zachować Maszynę ponieważ informacje o maszynie w zarchiwizowanym Rent, po ponownym załadowaniu programu (jeśli traktujemy repo jako bazę danych) będą wskazywały na nullptr (nie będą z racji użycia shared pointerów, ale będą zachowane w momencie gdy oczekujemy usunięcia informacji o komputerze co również jest błędem)
WNIOSEK: archiwizujemy maszyny wirtualne, po prostu zarchiwizowane maszyny nie zużywają pamięci
Description
Użytkownik Pro może korzystać z nielimitowanej liczby maszyn o nielimitowanej pamięci
Description
może wypożyczyć tylko jedną maszynę na raz oraz jej pamięć operacyjna nie może przekraczać 4 GiB
Description
Klasa Rent przedstawia fakt przypisania danej wirtualnej maszyny do danego użytkownika