UML dijagrami

Šta je UML?

Do sada smo uspešno naučili osnove OOP-a i možemo da projektujemo probleme iz realnog sveta. Međutim, šta ukoliko nije jasno definisan problem koji treba da rešimo? Koje su to stvari koje su nam neophodne kako bismo uspešno preveli neki realan sistem u kod?

Kako bismo rešili iznad navedene probleme, dogovrićemo se da koristimo jedinstven način postavljanja problema, razumljiv programerima i ljudima koji nemaju kontakta sa programiranjem.

Jedan od ovakvih načina jeste **jezik za ujedinjeno modeliranje** (**UML** - **Unified Modeling Language**).

Važno je napomenuti da UML nije programski jezik već vizuelni jezik. Za UML takođe možemo reći da je model po kome se opisuju konkretne ili konceptualne stvari.

Jedinstvenom skicom možemo precizno naglasiti zahteve, te pri interpretaciji takve skice neće biti problema.

Sam UML konkretno služi kako bi se prikazala ponašanja i strukture nekog sistema.

Komponente UML modela

Kako bismo detaljno mogli da razumemo dijagrame pisane u UML-u, moramo da utvrdimo iz kojih komponenata(gradivnih blokova) se UML model sastoji.

UML model u osnovi ima 3 gradivne komponente:

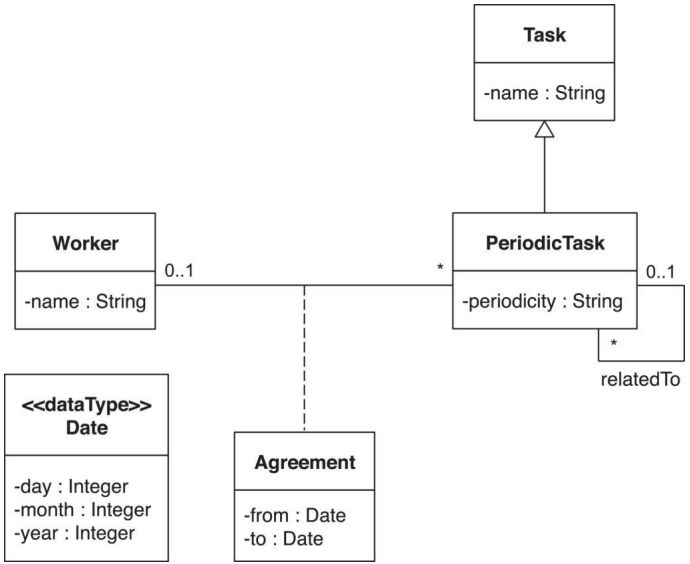
- Stvari(things)
- Relacije(relationships)
- Dijagrami(diagrams)

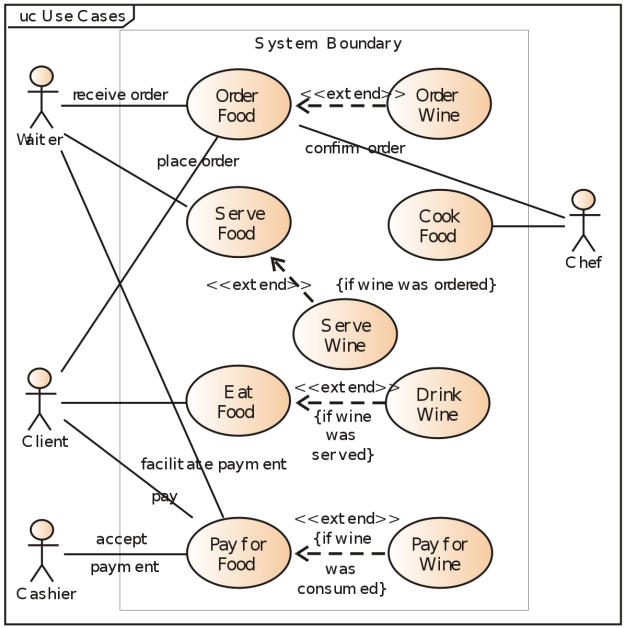
Kada govorimo o stvarima, reč je o svim statičkim, dinamičkim, organizacionim i objašnjavajućim komponentama modela. Svaki sistem mora da sadrži ove komponente jer su oni osnova sistema.

Relacije su svakakvi odnosi koji mogu da se uspostave među različitim stvarima.

Dijagrami su grafičke reprezentacije skupa povezanih elemenata, gde su najčešće elementi koji se povezuju stvari, a grane kojim se vezuju relacije.

Primeri UML dijagrama





Tipovi UML-a i strukturni UML dijagrami

Važno je napomenuti da UML nije skup pravila za prikaz jedinstvenog dijagrama, već skup više skupova pravila za prikaze različitih dijagrama.

Takav skup je jasno podeljen u dve celine:

1. Strukturni dijagrami (Structural Diagrams)
2. Dijagrami ponašanja(Behavior Diagrams)

Strukturni dijagrami govore o tome kakav je sistem postavljen u osnovi(koji entiteti postoje i kakve su njihove osobine) - statički aspekat.

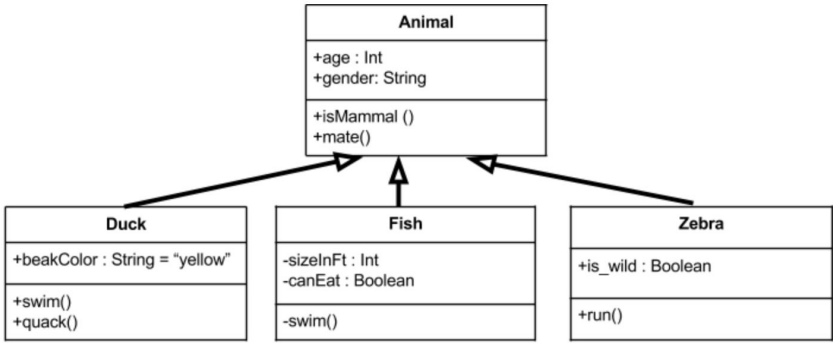
Dijagrami ponašanja govore o tome kako sistem treba da se ponaša u odgovarajućim situacijama(načini na koji entiteti interaguju u okviru sistema) - dinamički aspekat.

Za potrebe ovog kursa, razmatraćemo isključivo strukturne UML dijagrame. Konkretno, biće reči o klasnim dijagramima koji se svrstavaju u strukturne dijagrame.

UML klasni dijagrami

Kao što smo napomenuli, akcenat će biti na UML klasnom dijagramu (Class Diagram). Ova vrsta dijagrama se najčešće koristi u praksi i predstavlja osnovu svih objektno-orijentisanih softverskih sistema.

Klasni dijagrami služe za opisivanje statičkih (nepromenljivih) komponenti nekog sistema. Neki primeri takvih komponenti su klase po kojima se prave objekti, kao i definicije njihovih stanja i ponašanja.



Komponente klasnog dijagrama

Svaki klasni dijagram sastoji se iz 4 komponente:

- samih definicija klasa
- atributa
- operacija ili metoda
- odnosa među objektima(klasama)

U zaglavlju(najčešće boldirano) navodi se naziv klase.

U narednom nivou nalazi se skup atributa. Svaki atribut nalazi se u novom redu i obavezno je karakterisan nazivom i oznakom vidljivosti.

- (+) označava javne atribute
- (-) označava privatne atribute

U poslednjem nivou nalazi se skup operacija odosno metoda. Pravila vidljivosti i ovde važe.

Definicija svojstava

Za definisanje svojstava u okviru UML klasnog dijagrama moguća su dva pristupa.

Na prethodnom slajdu pojavljuje se klasa bez potpisa. Kod nje može biti uočeno da nisu navedeni tipovi svojstava.

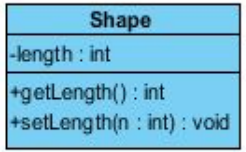
Međutim, nekada nam je lakše, drugim prilikama strogo neophodno da naglasimo kojih tipova određena svojstva moraju biti. Tada koristimo klase sa potpisima.

Ispod je dat primer iste klase kao malopre, samo što su dati i tipovi atributa, kao i povratne vrednosti i tipovi argumenata funkcija.

Primer:

+ naziv_promenljive : tip_promenljive

+ naziv_metode : (tip_arg) : tip_povratne_vrednosti



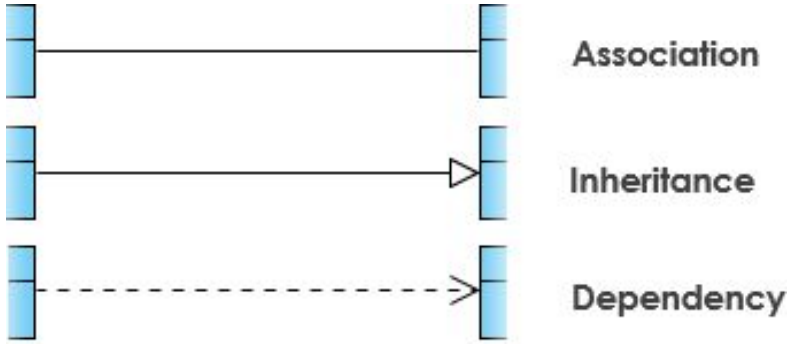
Odnosi komponenata u klasnom dijagramu

U okviru klasnog dijagrama, komponente se povezuju uspostavljanjem odnosa između klasa.

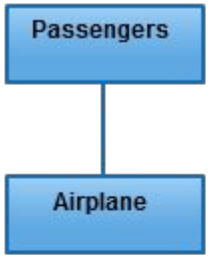
Postoje više tipova odnosa:

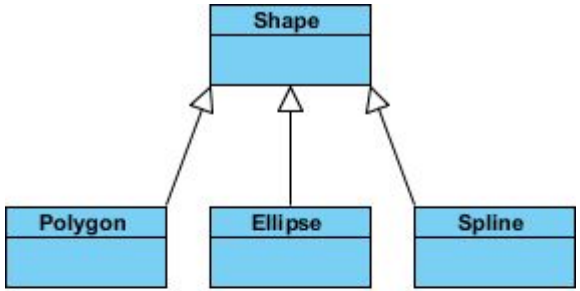
- asocijacija
- nasleđivanje
- zavisnost
- realizacija
- agregacija
- kompozicija

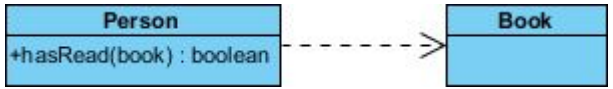
Za izradu naših dijagrama, glavni akcenat biće na ispod datim odnosima.



Primeri asocijacije, nasleđivanja i zavisnosti

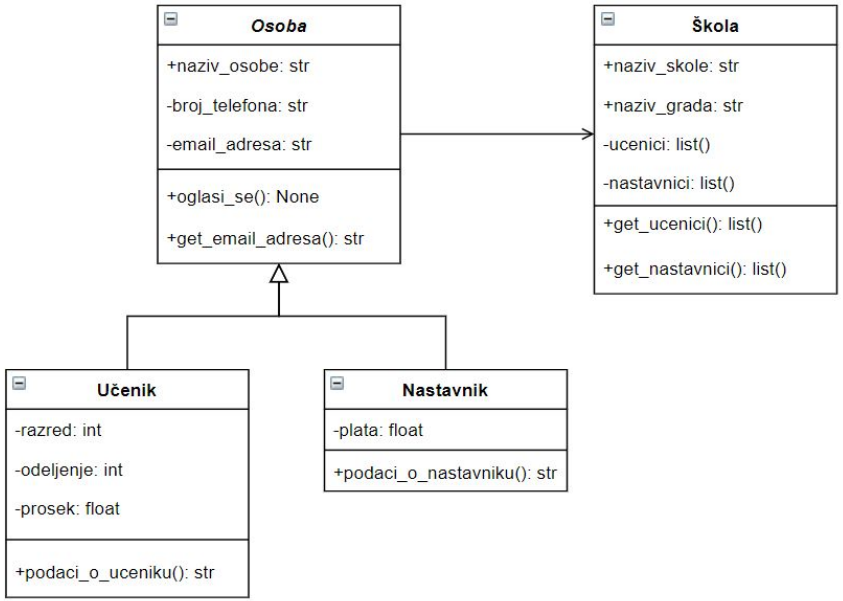






Primer X.

Napisati skelet za Python kod po ugledu na dati UML klasni dijagram.



```
from abc import ABCMeta, abstractmethod

class Osoba(metaclass=ABCMeta):
      @abstractmethod
      def __init__(self, naziv_osobe, broj_telefona, email_adresa):
            Pass
      @abstractmethod
      def oglasi_se(self):
            Pass
      @abstractmethod
      def get_email_adresa(self):
            pass
```

```
class Ucenik(Osoba):
      def __init__(self):
            self.naziv_osobe = "Petar"
            self.__broj_telefona = "+381644092717"
            self.__email_adresa = "peraperic.15377@gmail.com"
            self.__razred = 6
            self.__odeljenje = 2
            self.__prosek = 4.80

      def podaci_o_uceniku(self):
            return "Petar 6/2 (4.80)"
      def oglasi_se():
            print("Zdravo!")
      def get_email_adresa(self):
            return self.__email_adresa
```

```
class Nastavnik(Osoba):
      def __init__(self):
            self.naziv_osobe = "Milorad"
            self.__broj_telefona = "+381625492717"
            self.__email_adresa = "milo_milo154.17@gmail.com"
            self.__plata = 99.999
      def oglasi_se(self):
            print("Postovanje!")
      def get_email_adresa(self):
            return self.__email_adresa
      def podaci_o_nastavniku(self):
            print("Milorad - [+381625492717] -  (99.999 RSD)")
```

```
class Skola:
	def __init__(self):
		self.naziv_skole = "Prva kragujevacka gimnazija"
		self.naziv_grada = "Kragujevac"
		self.__ucenici = [Ucenik()]
		self.__nastavnici = [Nastavnik()]
	def get_ucenici(self):
		return self.__ucenici
	def get_nastavnici(self):
		return self.__nastavnici
```