Osnove GIS-a

- **GIS (geoinformacioni sistemi)** - softverski sistemi sa mogućnošću unosa, skladištenja, manipulacije, analize, generisanja i prikaza geoprostornih podataka
- GIS posjeduje prostornu komponentu - predstavlja kartu sa bazom podataka iza nje
- Modeli podataka u GIS-u:
  - **Rasterski**
  - **Vektorski** - raspolažu dodatnim podacima
- Vrste GIS okruženja:
  - **Desktop GIS**
    - Desktop softverska rješenja (*QGIS*, *ArcGIS*...)
    - Primarna svrha je vizualizacija i analiza geoprostornih podataka
  - **Web GIS**
    - Baziran na troslojnoj arhitekturi: *klijentski sloj* (*frontend*), *serverski sloj* (*backend*), *sloj podataka*
    - Primarna svrha je vizualizacija i diseminacija geoprostornih podataka (*GeoSrbija*...)

Razvoj web aplikacija

- Iz dva dijela:
  - **Frontend** - *HTML*, *CSS*, *JavaScript*
  - **Backend** - *Java*, *.Net*, *Python*, *SQL*,...

Frontend razvoj

- Bavi se onim dijelom sajta ili *web* aplikacije koji se prikazuje u *web browser*-u
- Frontend razvoj odnosi se na razvoj *web* aplikacija kojima se značajan dio funkcionalnosti zasniva upravo na frond endu. To zahtjela temeljno poznavanje *JavaScript* programskog jezika
- Geo-*web* aplikacije su uglavnom fokusirane na implementaciju potrebnih funkcionalnosti i manje su orjentisane ka *web* dizajnu

HTML (Hyper Text Markup Language)

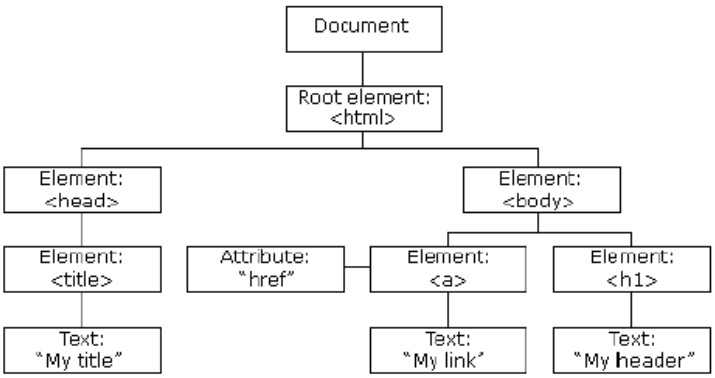
- Standardan jezik za označavanje koji služi za kreiranje *web* stranica
- Opisuje strukturu stranice pomoću takozvanih "oznaka" (*tagova*)

- *HTML* elementi su gradivni blokovi *HTML* stranice i predstavljeni su *tagovima*
- *HTML tagovi* obilježavaju dijelove sadržaja kao što su "naslov", "paragraf", "tabela", "link", "slika"...
- *Web browser* ne prikazuje *HTML* oznake, već ih koristi za prikazivanje sadržaja stranice
- Primjer *HTML* koda:

```
<html>
        <head>
                <title> Naslov stranice </title>
        <\head>

        <body>
                <h1> Naslov veličine 1 </h1>
                <p> Paragraf </p>
        </body>
</html>
```

- ***HTML DOM*** je objektni model za *HTML*. On definiše:
  - *HTML* elemente kao objekte
  - Atribute za sve *HTML* elemente
  - Metode za sve *HTML* elemente
  - Događaje za sve *HTML* elemente
- Služi kao API (programski interfejs) za *JavaScript*, tako da *JavaScript* može:
  - Dodati, izmijeniti ili ukloniti *HTML* elemente
  - Dodati, izmijeniti ili ukloniti *HTML* atribute
  - Dodati, izmijeniti ili ukloniti *CSS* stilove
  - Reagovati na *HTML* događaje
  - Dodati, izmijeniti ili ukloniti *HTML* događaje
- Primjer *DOM* stabla:



- *HTML* elementima se, upotrebom *JavaScripta*, pristupa pomoću *DOM* stabla

CSS (Cascading Style Sheets)

- Jezik koji opisuje stil *HTML* dokumenta - kako bi *HTML* elementi trebalo da budu prikazani
- Primjer *CSS* koda:

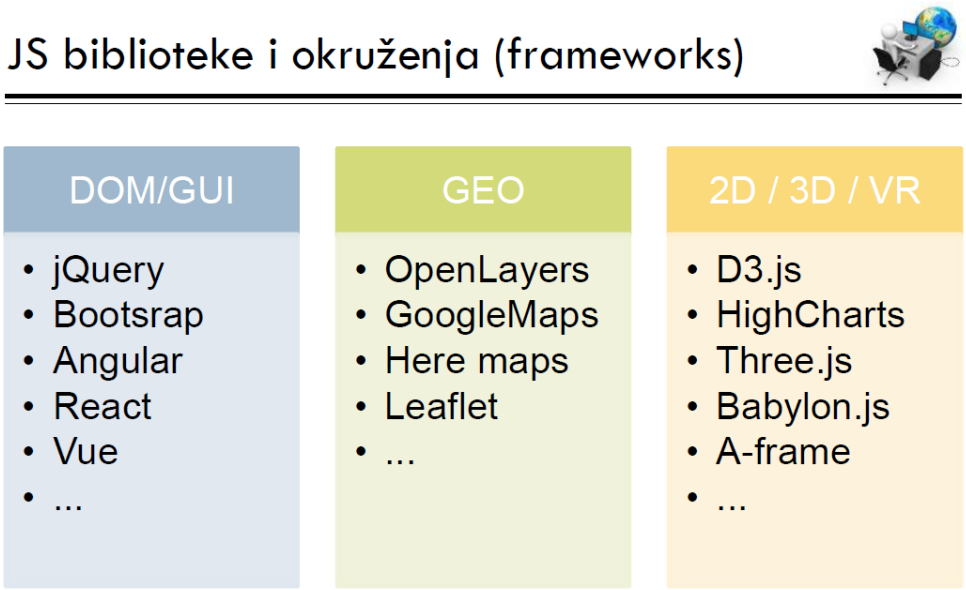
```
<style>
body {
        backgroud-color: lightblue;
}

h1 {
        color: while;
        text-align: center;
}

p {
        font-family: verdana;
        font-size: 40px;
}
</style>
```

JavaScript

- Zovu ga još i *vanila JavaScript* (običan, osnovni)
- Skriptni programski jezik koji se prvenstveno koristi za definisanje funkcionalnosti *web* stranica na klijentskoj strani
- Dinamičan, slabo tipiziran jezik sa skromnom podrškom za objektno orjentisano programiranje



- *JS AJAX* omogućava :
  - Asinhronu komunikaciju sa *web* serverom - asinhrono ažuriranje *web* stranice razmjenom podataka sa *web* serverom
  - Čitanje podataka sa *web* servera
  - Ažuriranje *web* stranice bez ponovnog učitavanja stranice
  - Slanje podataka na *web* server - u pozadini
- Koristi kombinaciju *XML HTTP Request* objekta koji je ugrađen u *browser*, kao i *JavaScript* i *HTML DOM* (za prikaz ili upotrebu podataka)
- Radi tako što, kada se desi događaj (npr. kliknuto dugme), *JS* kreira *XML HTTP Request* objekat koji šalje zahtjev *web* serveru. Nakon obrade zahtjela, server šalje odgovor *web* stranici koji je pročitan od strane *JS*-a, te se izvrši odgovarajuća akcija
- Razmjena podataka ide u *JSON* formatu - tekst napisan u *JavaScript* objektnoj notaciji
- Primjer *JSON* teksta:

```
'{"name": "Petar", "age": 30, "city": "Novi Sad"}'
```

- Bilo koji *JavaScript* objekat može se konvertovati u *JSON* i kao takav poslati serveru, kao i obratno.

GeoJSON

- *GeoJSON* je format za kodiranje različitih geoporostornih struktura podataka
  - Podržava tipove geometrije kao što su:
    - *Point* i *MultiPoint*
    - *LineString* i *MultiLineString*
    - *Polygon* i *MultiPolygon*
  - Geometrijski objekti sa dodatnim svojstvima su *Feature* objekti. Setovi funkcija su sadržani u objektima *FeatureCollection*
- Primjer *GeoJSON*-a:

```
{
        "type": "Feature",
        "geometry": {
                "type": "Point",
                "coordinates": [125.6, 10.1]
        },
        "properties": {
                "name": "Dinagat Islands"
        }
}
```

jQuery

- *jQuery* je *JavaScript* biblioteka za pristupanje *HTML DOM* stablu (elementima *HTML* stranice)
- Značajno pojednostavljuje *JavaScript* programiranje
- Primjer *JavaScript* koda sa *jQuery*:

```
$(document).ready(function(){
        $("p").click(function(){
                $(this).hide();
        });
});
```

Bootstrap

- Najpopularniji *HTML*, *CSS* i *JavaScript* framework za razvoj odzivnih (*responsive*) *web* sajtova - sajtova koji se prilagođavaju da izgledaju dobro na svim uređajima. *Bootstrap* koristi *grid system* kako bi postigao datu prilagođenosti
- Primarno za mobilne uređaje
- Uključuje obrasce za tipografiju, forme, dugmad, tabele, navigaciju, modalne forme, pokretne slike...

Node.js

- Višeplatformsko *JavaScript* okruženje za izvršavanje *JavaScript* koda na serverskoj strani.
- Omogućava da se sadržaj dinamičkih *web* stranica generiše na serveru prije nego što se pošalje do *web browser*-a korisnika.

Angular

- Popularan *framework* za brz razvoj modernih mobilnih i desktop *web* aplikacija
- Za razvoj su potrebni *Node.js*, kao i *npm package manager*
- *Angular CLI* se koristi za kreiranje projekata, generisanje aplikacija i koda, testiranje i postavljanje aplikacije
- *Angular, Angular CLI i Angluar* aplikacije direktno zavise od funkcionalnosti koje pružaju biblioteke dostupne kao *npm* paketi
- Dijeli aplikaciju na komponente od kojih je početna *app.component*
- *Angular Material* je biblioteka koja sadrži *UI* komponente

OpenLayers

- *JavaScript* biblioteka koja služi za postavljanje dinamičkih mapa na *web* stranice
- Može da prikaže markere, rasterske i vektorske podatke iz bilo kog izvora
- Napravljen je da unaprijedi diseminaciju geoprostornih informacija svih vrsta
- Omogućava prikazivanje rastera iz različitih izvora, kao i prikaz i modifikaciju vektorskih geopodataka

Google Maps

- *JavaScript API* koji omogućava korisnicima prilagođavanje mapa i dodavanje svog sadržaja i slika za prikaz na *web* stranicama i mobilnim uređajima
- Sadrži četiri osnovna tipa mape:
  - Putevi
  - Satelitske mape
  - Hibridne mape
  - Terenske mape
- Korisnik može modifikovati date mape upotrebom slojeva i stilova, kontrola i događaja
- Zahtjeva *API* ključ - nije potpuno besplatna
- Koristi podatke sa *Google* servisa

Leaflet

- *JavaScript* biblioteka za interaktivne mape namijenjene mobilnim uređajima
- Osmišljen sa osnovnim ciljem da postigne jednostavnost, performanse i upotrebljivost
- Može biti proširen brojnim *pluginima*

Backend razvoj

- *Backend* development tiče se poslovne logike koja se izvršava na serveru i direktno komunicira sa bazom podataka dajući u isto vrijeme klijentskoj strani interfejs za pristup tim podacima
- Obuhvata širok tehnološki spektar, od skripte koja pokreće kontakt formu na sajtu do kompleksnih softverskih sistema:
  - Programski jezici: *Java*, *.Net*, *Python*,...
  - Razvojna okruženja: *Spring*,...
  - Baze podataka - *SQL*,...
  - *JPA* - objektno relaciono mapiranje
  - ...

Spring

- Okruženje koje pruža sveobuhvatan programski i konfiguracioni model za moderne *Java* bazirane *enterprise* aplikacije - na bilo kojoj platformi za razvoj i instalaciju
- Ključni element *Spring*-a je infrastrukturna podrška na aplikativnom nivou - *Spring* na taj način omogućava razvojnom timu da se fokusira isključivo na poslovnu logiku, bez nepotrebnih veza na određenu platformu i mrežno okruženje
- *Spring Boot* implementira *Spring* okruženje tako da maksimalno olakša razvoj uz minimum konfigurisanja

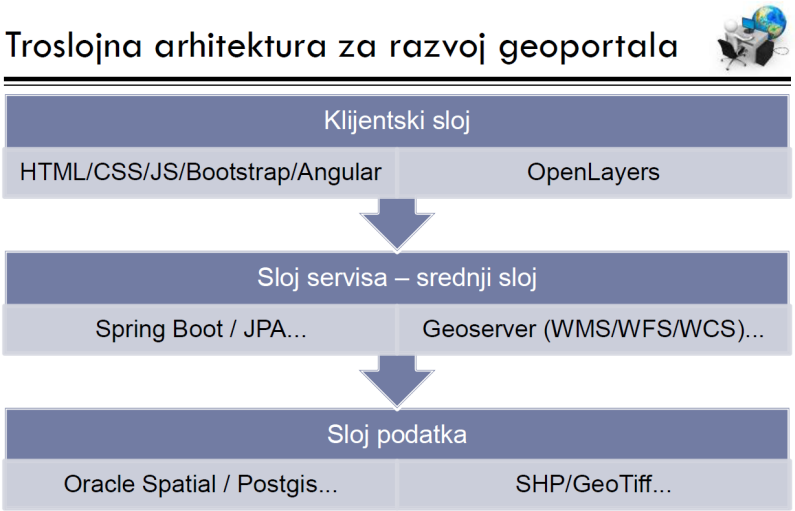
SQL - Structured Query Language

- Standardan jezik za skladištenje, čuvanje i preuzimanje podataka iz baze podataka
- Namijenjen za upravljanje podacima u relacionim sistemima za upravljanje bazama podataka. Obuhvata unos podataka, upite, ažuriranje i brisanje, kreiranje i mijenjanje šeme, kao i podatke za kontrolu pristupa
- Relacioni upitni jezik. Uniforman - podaci i rezultati operacija se prikazuju u vidu tabele

JPA - Java Persistance API

- *API* za objektno-relaciono mapiranje
- *Java Persistance API* je *Java API* (interfejs za programiranje aplikacija) koji opisuje upravljanje relacionim podacima u objektno-orjentisanim aplikacijama napisanim u *Java* programskom jeziku
- Podržava osnove *CRUD* (*create/read/update/delete*) operacije
- Izbjegnuto pisanje *SQL*-a (ubrzan razvoj)

Troslojna arhitektura



Uvod u GIS

- **Geografski informacioni sistemi** - specijalizovani sistemi koji prate ne samo pojave, aktivnosti i događaje već i gdje se takve pojave, aktivnosti i događaji nalaze.
- Na taj način, geografska lokacija ili ono gdje se nešto nalazi postaje bitan atribut aktivnosti, rukovođenja, strategija, planova i odluka
- Izvori ulaznih podataka za *GIS* su *globalni sistemi pozicioniranja* (*GPS*) i *remote sensing*
  - *Globalni sistem pozicioniranja* (*GPS*) je sistem satelita koji mogu obezbjediti preciznu lokaciju na Zemljinoj površini
  - *Remote sensing* je korišćenje satelita ili aviona za prikupljanje informacija o površini Zemlje
- Definisanje geografskih informacionih sistema:
  - Zajednička osnova između obrade informacija i podataka iz mnogih oblasti koje koriste tehnike prostorne analize (*Tomlison, 1972.*)
  - Moćan skup alata za prikupljanje, čuvanje, preuzimanje, transformaciju i prikaz prostornih podataka iz stvarnog svijeta (*Burroughs, 1986.*)
  - Kompjuterizovani sistem upravljanja bazom podataka za čuvanje, pronalaženje, analiziranje i prikazivanje prostornih podataka (*NCGIA, 1987.*)
  - Sistem podrške odlučivanju koji uključuje integraciju prostorno referenciranih podataka u rješavanju problema životne sredine (*Coven, 1988.*)
  - Sistem integrisanih računarskih alata za procesiranje (snimanje, skladištenje, pronalaženje, analiziranje, prikazivanje) podataka koristeći lokaciju na površini Zemlje u cilju upravljanja operacijama, donošenja odluka i razvoja nauke
  - Virtuelna reprezentacija infrastrukture stvarnog svijeta koja može da se pretražuje u cilju sprovođenja neophodnih operacija, analiza, donošenja odluka i formulisanje polisa
- U *GIS*-u, karte su povezane sa tabelama alfanumeričkih podataka. To znači da, kombinacijom različitih mapa, zapravo kombinujemo podatke da bi se dobile željene informacije
- *GIS* omogućava postavljanje pitanja sa mape ili iz baze podataka, postavljanje pitanja u odnosu na geografsku lokaciju, organizaciju velikog broja tipova prostornih informacija i sl.
- Primjene *GIS*-a:
  - 80% aktivnosti samouprave se bazira na prostornim podacima
  - Značajan dio državne uprave ima geografsku komponentu
  - Poslovne firme koriste *GIS* za veoma širok spektar primjena (precizna poljoprivreda, istraživanje prirodnih resursa, urbanizam, građevinarstvo...)

    - Naučna istraživanja (geologija, botanika, sociologija, epidemiologija...)
- *GIS* aplikacije:
    - Omogućavaju automatizaciju aktivnosti koje uključuju geografske podatke:
        - Proizvodnja mapa
        - Računanje površine, dužine ruta...
        - Logistika: planiranje rute, upravljanje saobraćajem...
    - Vezivanjem podataka za mape, dopušta se komunikacija složenih prostornih obrazaca
    - Daju odgovor na prostorne upite
    - Izvode složeno prostorno modelovanje (upravljanje resursima, planiranje katastrofa...)

Komponente geografskih informacionih sistema

- Sadrži četiri važne komponente:
    - Kompjuterski hardver
    - Niz aplikacionih softverskih modula
    - Odgovarajući organizacioni sadržaj koji obuhvata obučene ljude
    - Podaci
- Četiri interaktivne komponente:
    - Podsistem za unos:
        - Vrši konverziju karata i drugih prostornih podataka u digitalni oblik - *data input*
        - Korisnički upiti - *query input*
    - Podsistem za skladištenje i pozivanje podataka - *geographic database*
    - Podsistem za analizu i transformaciju
    - Izlazni podsistem za izradu karata, tabela i pružanje odgovara na postavljene upite
- Podaci su organizovani u slojevima, pri čemu svaki sloj predstavlja zajedničku osobinu. Slojevi su integrisani korišćenjem eksplicitne lokacije na Zemljinoj površini
    - Primjer: Tri sloja - putevi, hidrologija i topografija.

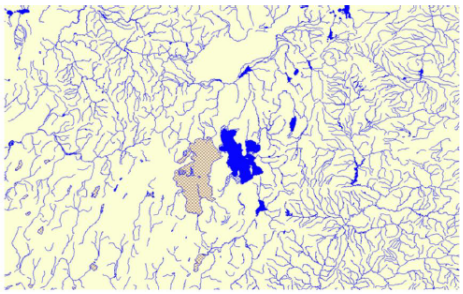
Tipovi prostornih fenomena

- Potreba za smještanjem fenomena realnog svijeta u model geoprostornih podataka iziskuje poznavanje strukture tih fenomena. Vrši se podjela na dvije vrste:

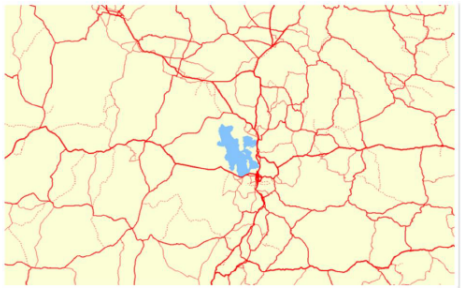
    - **Diskretni fenomeni**
        - Entitet koji postoji ali se individualno izdvaja od drugih fenomena

- Jasno definisane granice i lako je vidjeti gdje počinje, a gdje završava
- Primjer: rijeke, jezera, putevi (jasne granice početka i završetka),...

Rečni tokovi i jezera

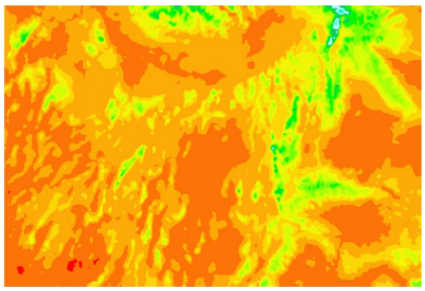


Putevi

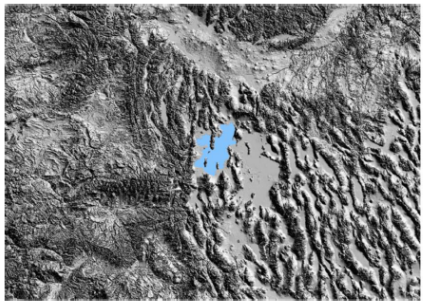


- **Kontinualni fenomeni**
  - Kontinualni podaci koji se ne mogu izdvojiti kao individualni
  - Primjer: temperatura, elevacija...

Temperatura



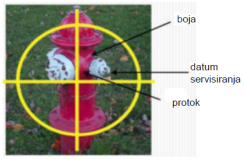
Elevacija



Atributi

- Neprostorne karakteristike koje opisuju prostorne entitete
- Organizuju se u tabele u kojima se red odnosi na jedan entitet, a kolona na jedan atribut koji opisuje entitet
- Svaki red se odnosi na jedan objekat u prostornom modelu. Objekat, po običaju, posjeduje više atributa koji se ispisuju u tabeli
- Smješteni u bazama podataka ili kao tekstualne datoteke

- Primer: vatrogasni hidrant
- Da bi smo smestili podatke o vatrogasnom hidrantu u prostorni model, neophodno je sačuvati njohovu geografsku lokaciju, ali i dodatne atribute koji opisuju hidrant.
- Atributi koji opisuju hidrant mogu biti boja, datum servisiranja i protok. Pozicija, boja, datum servisiranja i protok će biti smešteni kao jedan red u tabeli atributa koja sadrži 4 kolone zato što postoje 4 atributa koja opisuju vatrogasni hidrant.



- Četiri osnovna tipa podataka atributa:
  - *Integer*
  - *Float*/*Real*
  - *Text*/*String*
  - *Date*

Koncepti vektora i rastera

- **Vektorski model podataka**:
  - Definiše diskretne objekte - putevi, jezera, katastarske parcele, zgrade...
  - Tri tipa vektorskih modela podataka (sva tri tipa se sastoje od dodatnih podataka):
    - *Tačke*
      - Koriste jedan par koordinata da opišu svoju poziciju
      - Nemaju dimenziju iako je objekti reprezentovani njima u realnom svijetu imaju
      - Primjer: ulične svjetiljke, šahtovi,...
    - *Linije*
      - Određene setom koordinata
      - Svaka linija i kriva određene mnoštvom linijskih segmenata
      - Opisuje se čvorovima i verteksima (tačkama između čvorova). Čvorovi su tačke gdje linija počinje i završava se, dok su verteksi tačke gdje linija mijenja smjer
      - Atributi mogu biti vezani za cijelu liniju, ali i za pojedinačnu tačku linije ili linijski segment, što znači da linija može imati više redova u tabeli atributa
      - Primjer: putevi, cijevi, elektroenergetski kablovi,...
    - *Poligoni*
      - Formira se kao set linijskih segmenata u kojem je početna tačka ista kao i krajnja. Poligoni su tako zatvorene forme koje imaju unutrašnje regione
      - Atributi su vezani za centar poligona bez obzira koliko on složen bio
      - Primjer: jezera, gradovi, šume,...
  - Podaci su, dakle, struktuirani na sledeći način:
    - Prostorni objekat (*Feature* - opština)
    - Neprostorni atributi (tablearni podaci - naziv, površina, tip plana,...)
    - Prostorni atribut (geometrija - poligon)
  - Vektorski podaci se smještaju u *shapefile*-ove.
    -

- **Rasterski model podataka**:
  - Prostorni objekat opisan skupom *pixela* (fotografija).
  - Najbolje reprezentuje kontinualne objekte (temperatura, elevacija,...)
  - Može imati obllik seta ćelija kao što su *pixeli* na fotografiji ili da ćelije budu organizovane u *grid pattern* (matricu)
  - Svaka ćelija sadrži vrijednost, a koordinata ćelije ukazuje na njen centar
    - Svaka ćelija opisana je dimenzijom (njenom visinom i širinom)
  - Rasteri se smještaju u različite formate datoteka - *JPEG*, *TIFF*
    - *TIFF*
      - Jedan od najpopularnijih rasterskih formata
      - *Lossless* vid kompresije sa čuvanjem detalja
      - *GeoTIFF* - verzija *TIFF* formata sa ugrađenim geografskim podacima u okviru *tagova TIFF* fajla
      - Za smještanje i prenos digitalnih satelitskih snimaka, modela terena, aero snimaka, skeniranih mapa i sl.
    - *JPEG*
      - *Lossy* tehnika kompresije
      - Redukuje veličinu fajla na otprilike 5% od originalne verzije, ali se prilikom kompresije gube detalji
      - *JGW - JPEG World File* - koristi se da georeferencira *JPEG* fajl. Sadrži informacije o koordinatama i omogućuje da *JPEG* fajl bude ispravno pozicioniran na mapi
    - *World file*
      - Tekstualni fajl kojeg koriste geoinformacioni sistemi za georeferenciranje rastera.
      - Opisuje lokaciju, razmjeru i rotaciju mape

GIS softver

- **Desktop GIS**
  - Koristi se za kreiranje, uređivanje, upravljanje, analizu i prikaz geografskih podataka.
  - Klasifikuje se u tri kategorije funkcionalnosti:
    - *GIS Viewer*
    - *GIS Editor*
    - *GIS Analyst*
  - *QGIS*, *GRASS GIS*, *MapWindow GIS*, *Autodesk*, *ERDAS*, *ESRI*,...
  -

- **Sistemi za upravljanje prostornim bazama podataka**
  - Koriste se za čuvanje podataka, ali pružaju i funkcionalnost analize i manipulacije podacima
  - *PostGIS*, *MySQL Spatial*,...
- **WebMap serveri**
  - Za distribuciju mapa preko interneta (*Open Geospatial Consortium*)
    - *Web Map Service* - skup specifikacija interfejsa koji daju uniforman pristup od strane web klijenata mapama renderovanim na map serveru na internetu
    - *Web Feature Service* - web servis koji dopušta korisniku da objavi geoprostorne objekte na internetu zajedno sa definicijom njihove strukture
    - *Web Coverage Service* - podržava elektronsku razmjenu geoprostornih podataka u formi *coverage*-a, koji je definisan kao digitalna geoprostorna informacija koja predstavlja fenomen koji varira u prostoru
    - *GeoServer*, *MapServer*,...
- **Serverski GIS**
  - Serverski orjentisan *GIS* koji u osnovi pruža istu funkcionalnost kao i Desktop *GIS*, ali omogućava pristup ovoj funkcionalnosti putem mreža
- **WebGIS Clients**
  - Prestavljaju klijentske aplikacije koje se koriste za prikaz podataka i pristupaju funkcijama analize i upita na osnovu serverskog *GIS*-a preko interneta ili intraneta
  - Dvije vrste klijenata
    - *Thin* klijenti
      - Primjer: *web browser* koji se koristi za prikazivanje *Google* mapa
      - Pružaju samo funkciju prikaza i upita
    - *Thick* klijenti
      - Primjer: *Google Earth*, *Desktop GIS*
      - Obezbjeđuju dodatne alate za uređivanje podataka, analizu i prikazivanje
- **Biblioteke i ekstenzije**
  - Pružaju dodatne funkcionalnosti koje nisu dio osnovnog *GIS* softvera
  - Mogu pokriti alate za analizu terena, čitanje određenih formata podataka ili alate za kartografski prikaz geografskih podataka
  - *OpenLayers* (*open-source AJAX* biblioteka za pristup slojevima prostornih podataka), *GeoDjango*, *MapFish*, *OpenMap*,...
- **Mobile GIS**
  - Aplikacije za mobilne uređaje koje se koriste za prikupljanje podataka na terenu
  - Integrisano hardversko/softversko okruženje za pristup geoprostornim podacima i servisima preko mobilnih uređaja kao što su mobilni telefoni, tableti,...
  - Omogućava ažuriranje podataka u realnom vremenu

Klasifikacija podataka

- Kategorizuje objekte u različite klase na osnovu skupa definisanih uslova
- Klasifikacija može da dodaje ili modifikuje atribute podataka za svaki geoprostorni objekat
- Metode klasifikacije:
  - **Binarna klasifikacija**
    - Dijeli geo-objekte u dvije klase (0 ili 1, *true* ili *false* i sl.)
    - Najčešće se koriste za smještanje rezultata kada operacije vraćaju odgovor *true* ili *false*
    - Primjer: klasifikacija država na sjeverne i južne

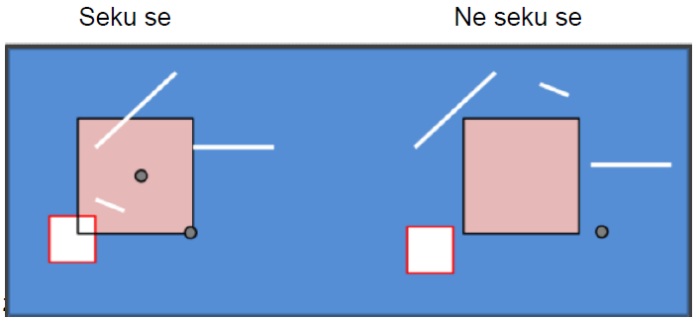
  - **Automatska klasifikacija**
    - Računar izvršava set korisnički definisanih pravila i daje rezultat
    - Primjer: klasifikacija jednakih frekvencija, standardna devijacija,...

Upiti nad atributima

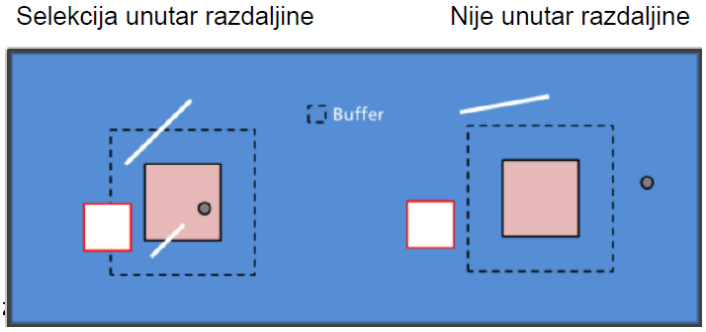
- Veoma čest oblik *GIS* prostornih operacija
- Selektuju podskup podataka na osnovu određenih vrijednosti atributa
- Svaki upit mora specifirati tri stvari:
  - Naziv atributa za pretragu
  - Naziv operatora poređenja
  - Vrijednost atributa
- Najčešće korišćeni operatori su *I*, *ILI* i *NE*
- Primjer: "površina veća od $60m^2$"
-

Prostorni upiti

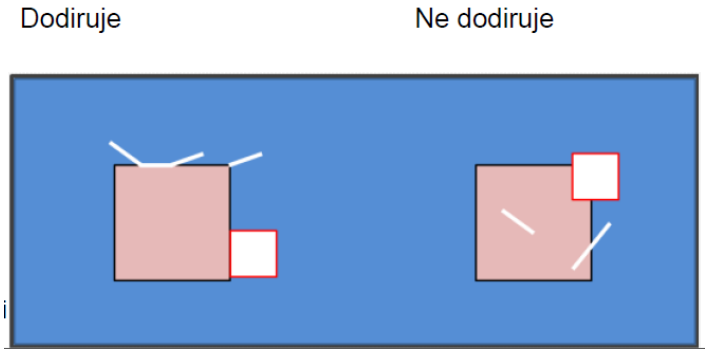
- Prostorni objekti se selektuju u zavisnosti od njihove lokacije u odnosu na druge prostore
- Primjer: Desio se nestanak struje u nekom dijelu grada. Formiranjem poligona koji prekriva područje bez struje mogu se izdvojiti postrojenja unutar poligona na kojima je mogao da se desi prekid
- **Operacija presjeka (intersection)**
  - Selektuje prostorne objekte koji imaju zajednički dio površine sa polaznim prostornim objektom
  - Primjer:



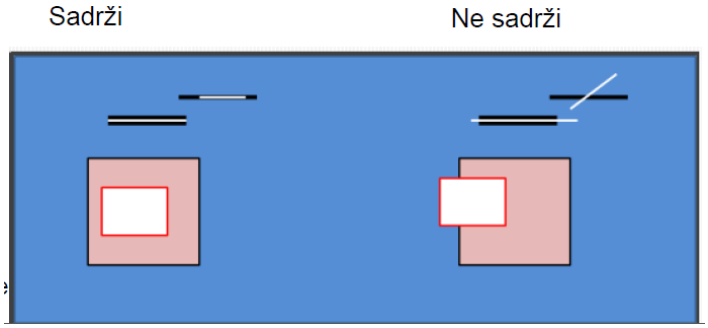
- **Provjera pripadnosti unutar razdaljine (are within a distance)**
  - Neophodno je odrediti rastojanje od polaznog objekta u cilju kreiranja *buffera*. Upit tada selektuje sve prostorne objekte koje se sijeku sa *bufferom*
    - *Buffer*
      - Region oko prostornog objekta koji je manji ili jednak zadatoj udaljenosti od prostornog objekta
      - Može biti kreiran oko tačaka, linija, poligona ili rasterskih skupova podataka
      - Koristi se za identifikaciju prostornih objekata koji na neki način interaguju sa *bufferom* određenog prostornog objekta
  - Primjer:



- **Provjera da li dodiruje graničnu liniju (touches the boundary of)**
  - Provjerava da li ciljni prostorni objekti dodiruju graničnu liniju polaznog objekta tako da se ne sijeku sa njegovom unutrašnjom površinom
  - Primjer:

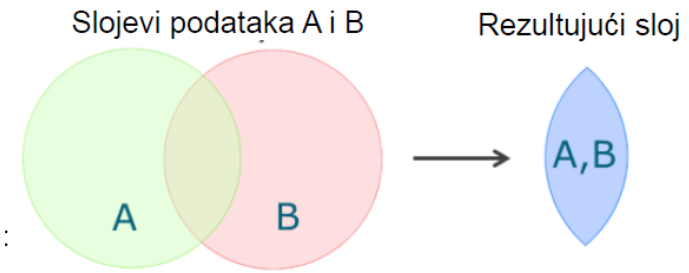


- **Sadrži (contains)**
  - Provjerava da li polazni objekat sadrži ciljne prostorne objekte u cijelosti zajedno sa graničnim linijama, selektujući ih

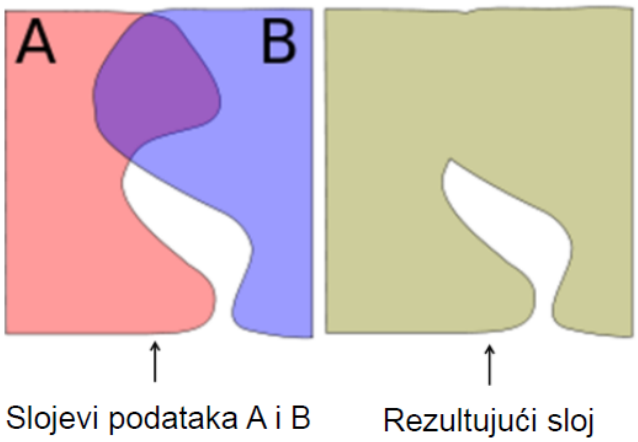


- **Overlay operacije**
  - Podrazumijevanju kombinovanje prostornih i atributivnih podataka iz dva ili više slojeva prostornih podataka
  - Podaci se učitavaju jedni preko drugih i gleda se gdje se preklapaju
  - Svi slojevi moraju biti u istom koordinatnom sistemu
  - Kreiraju novu geometriju i novi skup geoprostornih podataka
  - Kombinuju atribute slojeva vektorskih podataka koji su učestvovali u operaciji
  - Neke od operacija:
    - **Clip**
      - Siječe skup podataka na osnovu polaznog poligona i kreira novi sloj podataka sa rezultujućim podacima
      - U rezultujući sloj se prenosi samo geometrija i atributi podataka sloja koji se siječe

    - **Intersection**
      - Kombinuje atribute i geometrije svih slojeva koji učestvuju u presjeku ali samo u onim regionima gdje svi slojevi imaju podatke



- **Union**
  - U rezultujući sloj se prenose svi podaci i atributi iz polaznih slojeva
  - Kombinuje ulazne slojeve u jedan



GIS Analize

Geoprostorne analize

- Obuhvataju prikupljanje, prikazivanje i manipulaciju slikama, podacima sa *GPS*-a, satelitskim, avio, *UAV* fotografijama i istorijskim podacima, opisani eksplicitno geografskim koordinatama ili implicitno geografskim identifikatorima kao što je adresa

Vektorske analize

- **Analiza udaljenosti** (*Proximity analysis*)
  - Skup alata koji se koriste za analiziranje relacija između izabranih geoprostornih objekata i njihovih susjeda - *buffer* i *multiple rings buffer*
  - ***Workflow* i *GeoWorkflow***
    - Poslovni proces (*workflow*) je model koji opisuje ponovljivu sekvencu operacija i u računarstvu se koristi da predstavi interakciju korisnika i računara
    - Koraci za obradu se sastoje od:
      - ulaza

        - transformacionih pravila
        - izlaza
    - *GeoWorkflow* predstavlja proširenje pojma poslovnog procesa
        - Geoprostorni model - skup geoprostornih procesa sa kontrolnom strukturom baziranom na ponašanju ulaza i izlaza koji predstavljaju znanje eksperta iz domena
        - Svi procesi organizovani sekvencijalno ili konkurentno, gdje za svaki par susjednih procesa važi da je izvršenje prvog neophodno za izvršenje drugog
- **Izbor lokacije** (*Site selection*)
    - Postupak kojim se mjeri potreba projekta u odnosu na veći broj ponuđenih lokacija
    - Primjer: pronalaženje idealne lokacije za pravljenje novog postrojenja, pronalaženje idealne lokacije za smještanje aerodroma,...
- **Mrežna analiza** (*Network analysis*)
    - Sistem povezanih tačaka i linija koje predstavljaju moguće putanje od jedne lokacije ka drugoj
    - Odgovara na pitanja tipa:
        - "Koji je najbrži put do...?"
        - "Koja patrola se nalazi najbliže...?"
        - "Koje je najbolje mjesto za otvaranje nove poslovnice?"
    - Pruža informacije pomoću kojih se može napraviti logistička i strateška analiza
    - Isključivo u vektorskom formatu, dok su veze predstavljene linijama
    - Sve mreže uključuju *impedansu* - prepreke koje usporavaju vrijeme putovanja (distanca, trošak za gorivo, plata vozača,...)
    - Alati mrežne analize:
        - *Najbliži objekat*
            - Mjeri trošak putovanja između objekata ili određuje koji je objekat najbliži biranom mjestu
            - Moguće je specificirati *impedansu* duž puta
            - Primjer: Uputiti vatrogasno vozilo ka lokaciji požara iz najbliže vatrogasne stranice
        - *Izbor putanje*
            - Omogućava izbor najbolje putanje bazirane na zadatim kriterijumima
            - Obično se koristi da nađe putanju uz najmanje troškove koja posjećuje više lokacija
            - Primjer: Pronaći put koji će potrošiti najmanje vremena i goriva

    - *Servisne mreže*
        - Identifikuje pristupačne ulice unutar zadate impedanse
        - Primjer: Sva domaćinstva koja su unutar $10km$ od škole, uzimajući u obzir troškove puta, počevši od polazišta
    - *Alokacija lokacija*
        - Upotrebom odgovarajućih ulaznih parametara, ovaj alat locira najbolje lokacije za nove objekte uzimanjem u obzir zahtjevane tačke
        - Primjer: Gdje treba locirati restoran brze hrane da bi se minimizovalo vrijeme puta mušterija do restorana
- Topologija
    - Studija o geometrijskim svojstvima koja se ne mijenjaju kada objekat prolazi kroz transformacije ili modifikacije
    - Reprezentuje i nameće geometrijske relacije između objekata
    - Dvije vrste:
        - **Planarna**
            - Smatra se da geometrijske reprezentacije postoje na jednoj dvodimenzionalnoj površini
            - Nisu dozvoljena preklapanja poligona, a da ne prave novi poligon
            - Ako se linije sijeku, mora da postoji presjek na svakom prelasku linije preko linije
            - Ako se jedna topologija pomjeri, tada se čvor koji predstavlja presjek takođe pomjera i na taj način se zadržava svojsvo presjeka između dvije linije. Važi i za poligone
        - **Neplanarna**
            - Objekti postoje u više ravni sa neznatnim preklapanjem kod ivica
            - Nije neophodno da postoji presjek gdje linija prelazi preko linije, ako su one u različitim ravnima. Važi i za poligone
    - Prednosti topologija:
        - Osiguravaju kvalitet podataka
        - Osiguravaju povezanost linija
        - Obezbjeđuju integritet podataka kroz nametanje logičkih pravila
    - Mane topologija:
        - Osiguravanje topologije je vremenski intenzivan proces
        -

Rasterske analize

- **Triangulacija**
  - ***TIN*** - ***Triangular Irregular Network***
    - Mreža nepravilnih trouglova - vektorski bazirana struktura podataka napravljena na bazi triangulacije između tačaka
    - Mreža međusobno povezanih trouglova koja obično predstavlja vrijednosti visina za topografske podatke
    - Prednost mreže nepravilnih trouglova naspram rastera je da može upravljati neuniformnim rastojanjima između ulaznih tačaka, kreirajući time neuniformne veličine trouglova
    - *Delaunay* triangulacija
      - Računska geometrija za triangulaciju tačaka (kreiranje mreže nepravilnih trouglova)
    - Upotreba: Mapiranje fizičke površine Zemlje, mapiranje dna mora, analiza nagiba terena,...

- **Tehnike analize terena**
  - **Analiza nagiba**
    - Kreira površinu koja prikazuje vrijednost nagiba terena. Takva površina prikazuje gdje je teren ravan, umjereno ili jako strm
    - Ovi alati zahtjevaju da ulazi budu rasteri, a izlaz je također raster koji definiše jačinu nagiba u svakom pikselu
    - Nagib se iskazuje u procentima ili se može klasifikovati da bi se pojednostavio prikaz (ako je nagib veći od 15%, smatrati ga prestrmim)
  - **Analiza pravca nagiba**
    - Površinski aspekt prikazuje orijentaciju terena u svakoj tački
    - Predstavlja usmjerenje nagiba koje pokazuje u kom pravcu se nagib prostire
  - ***Hillshades***
    - *Hillshades* modeli bacaju sjenku na terena za zadatu lokaciju sunca (reljef sa sijenkom)
  - ***Viewshed***
    - Prikazuje područja koja su vidljiva iz određene izvorne tačke
    - Koristi vrijednost visina da odredi šta je vidljivo, a šta nije
    - Određuje koji objekti smetaju pogledu i koristi ih kod određivanja pravca pogleda, da odredi šta je vidljivo sa pozicije posmatrača

- **Mapiranje gustine**

    - Omogućava korisniku da vidi gdje se nalazi veliki broj observacija ili viših vrijednosti
    - Gustine prikazuju prostorne relacije među različitim lokacijama podataka
    - Omogućava da se kreiraju predikcije iz podataka gdje će se druge lokacije ili vrijednosti dogoditi
    - Primjer: Mapiranje lokacija sa najviše ili najmanje bolnica,...
        - *Gustina tačaka*
            - Izračunava gustinu tačaka na određenoj oblasti tako što broj tačaka dijeli površinom zadate oblasti
        - *Gustina linija*
            - Određuje oblast sa većom gustinom linija naspram oblasti sa manjom
    - *Kernel* gustina
        - Računa gustinu tačaka ili linija
        - Postavlja veću težinu nekim objektima bazirano na specifičnim vrijednostima
        - Primjer: prilikom pretrage vatrogasnih hidranata, neki hidranti neki imaju veći pritisak i samim time pokrivaju veću površinu - imaće veću prednost od hidranta sa manjim pritiskom
        - Generiše glatkije formacije gustine
    - *QGIS plugin Heatmap* omogućava generisanje površine gustine tačaka (bazirane na nekom atributu)
- **Reklasifikacija rastera**
    - Proces da se promjene vrijednosti svake ćelije rastera u neku drugu vrijednost
    - Primjer: pojavila se nova informacija i treba reklasifikovati raster,...
    - *Map algebra* - kombinacija rasterskih *layera* u ćeliji kroz proces ćelije (sabiranje, oduzimanje i sl. na nivou svake ćelije)
    - *Ponderisanje*
        - Pri analizi podataka, neki podaci mogu biti važniji od drugih
        - Kako bi se reflektovala važnost nekih vrijednosti od drugih, oni se mogu različito ponderisati u zavisnosti od nivoa važnosti
        - Primjer: nagib je važniji od ostalih parametara kada se posmatra oticanje vode

Konverzije između vektorskih i rasterskih podataka

- **Konverzija vektora u raster**
    - Konverzija tačaka, linija ili poligona u raster
    - Omogućava korisnicima da vide kontinualni pogled na podatke
    - Može dovesti do degradacije tačnosti podataka

- **Konverzija rastera u vektor**
  - Konverzija rastera u tačku, liniju ili poligon
  - Pri konverziji, ako ne postoji vrijednost date ćelije, tu se neće kreirati tačka, linija ili poligon

Geostatistika

- Vrsta statistike koja se koristi da analizira i predvidi vrijednost pridružene prostornim ili prostorno-vremenskim fenomenima
- Prvobitno bili razvijeni kao praktična sredstva da opišu prostorne obrasce i da interpoliraju vrijednosti za lokacije na kojima uzorci nisu uzeti
- Danas mogu dati mjeru neizvjesnosti za svaku lokaciju - ključno za pouzdano donošenje odluka
- Primjena:
  - Rudarstvo - kvantifikacija mineralnih resursa i procjena ekonomske izvodljivosti projekta
  - Ekološke nauke - procjena nivoa zagađujućih materija
  - Nauke o zemljištu - mapiranje nivoa hranljivih materija u tlu
  - Meteorološka primjena - predviđanje temperature, padavina,...
- Geostatička analiza
  - Koristi uzorke na različitim lokacijama nekog područja i stvara (interpolira) kontinualnu površinu
  - Izvodi povšinu koristeći vrijednosti sa izmjerenih lokacija za predviđanje vrijednosti za svaku drugu lokaciju datog područja
  - Geostatički tok rada:
    - Mapiranje i proučavanje podataka
    - Izgradnja geostatičkog modela
      - *Pre-processing* podataka
      - Modelovanje prostorne strukture
      - Definisanje strategije pretrage
    - Predviđanje vrijednosti na lokacijama koje nisu ispitane
    - Kvantifikovanje neodredjenosti predikcije
    - Provjera modela
    - Upotreba informacija u analizama rizika i donošenju odluka
- Metode:
  - Oslanjaju se na sličnosti bliskih tačaka
  - Dvije grupe interpolacionih tehnika:
    - **Determinističke**

    - Koriste matematičke funkcije za interpolaciju koje određuju glatkoću rezultujuće površine
    - Direktno bazirani na izmjerenim vrijednostima okruženja
    - Primjer: Najčešće korišćeni tip površine je digitalni elevacioni model terena
    - Glavni izazov sa kojim se suočava većina *GIS* modela je stvaranje najtačnije moguće površine iz postojećih podataka uzoraka, kao i određivanje greške i varijabilnosti predviđene površine
    - **IDW - Inverse Distance Weighting**
        - Da bi se uzela u obzir udaljenost dvije tačke, vrijednosti bližih tačaka su ponderisane više nego kod onih koje su udaljene - osnova za *IDW* interpolacionu tehniku - težina vrijednosti opada sa porastom razdaljine od lokacije za predikciju
        - *Vizuelizacija globalne polinomijalne interpolacije*:
            - Prvog reda:
                - Kontinualni nagib
                - Ne uzima u obzir lokalne varijacije
            - Drugog reda
                - Za modelovanje doline
                - Dozvoljeno jedno savijanje ravni
        - *Vizuelizacija lokalne polinomijalne interpolacije*:
            - Uzima u obzir lokalne varijacije
            - Uklapa mnogo manjih preklapajućih ravni, a zatim koristi centar svake ravni kao predviđanje za svaku lokaciju
            - Rezultujuća površina fleksibilnija i preciznija
        - *Vizualizacija funkcije radijalne osnove*
            - *Radial basis function*
            - Omogućava praćenje globalnog trenda uz uzimanje u obzir lokalne varijacije
- **Geostatističke**
    - Bazirane na statističkim modelima koji uključuju *autokorelaciju* (statističke relacije između izmjerenih tačaka)
    - Osim predviđanja površine, mogu dati i određenu mjeru neizvjesnosti ili tačnosti predviđanja
    - **Kriging**
        - Ponderiše okružujuće izmjerene vrijednosti kako bi izveo predviđanje za svaku lokaciju (sličan *IDW*-u)

        - Ponderi nisu samo bazirani na udaljenosti između izmjerenih tačaka i lokacija za predviđanje, već i na cjelokupnom prostornom rasporedu između izmjerenih tačaka
        - Prostorna autokorelacija mora biti kvantifikovana
            - Prirodni fenomeni često predstavljaju prostornu autokorelaciju - da su vrijednosti uzoraka koje se uzimaju jedni pored drugih sličnije od uzoraka uzetih daleko jedan od drugog
            - Kada postoji, tradicionalne statističke metode ne mogu se pouzdano koristiti
    - *ESRI ArcGIS Geostatical Analyst* je alat koji pruža mogućnost modelovanja površine upotrebom deterministčkih i geostatističkih metoda u *ESRI ArcGIS* okruženju
        - Omogućava generisanje interpolacionih modela i procjenu njihovog kvaliteta

JavaScript

- Obezbjeđuje interaktivnost na *web* stranicama
- Izvršava (interpretira) se u *web browseru*
- Ugrađuje se u *HTML* stranice
    - *Tag* `<script>` specificira kod koji se pokreće direktno u *browser*-u
    - Može se javiti bilo gdje u *HTML* dokumentu
        - Kad je definisan u `<body>` sekciji, izvršava se prilikom crtanja stranice
        - Kad je definisan u `<head>` sekciji, ne izvršava se automatski, već se poziva iz skripta u `<body>` sekciji

```
<html>
        <head>
                <script type="text/javascript">
                ...
                </script>
        </head>

        <body>
                <script type="text/javascript">
                ...
                </script>
        </body>
</html>
```

- Nema tipove podataka

- Nema kreiranja novih klasa (ugrađene funkcije i objekti)
- Sistem događaja

Promjenljive u JavaScript-u

- Deklaracija ključnom rječju *var*

```
var a;
var b = 5;
var c = "Pera";
```

- Jezik definiše 5 primitivnih tipova podataka:
  - *Number*
  - *String*
  - *Boolean*
  - *Undefined*
  - *Null*
- Nakon deklaracije, varijabla se može:
  - Inicijalizovati
  - Promijeniti tip

```
var x = 5;
x = 5.3;
x = "Mika";
```

Operatori u JavaScript-u

- Aritmetički operatori
  - Operatori sabiranja, oduzimanja, množenja, dijeljenja, inkrementacije,...
  - Operator sabiranja ima posebno značenje kada su operandi strigovi
    - Kada sabiramo *stringove* i brojeve, rezultat je ***string***
- Operatori dodjele
  - Operator jednakosti, "+=", "-=",...
- Relacioni operatori
  - Operatori poređenja (<, >, <=, >=,...)
- Logički operatori
  - Logičko *I*, *ILI*, *NE*
- Uslovni operator
  - *promjenljiva* = *uslov* ? *vrijednost1* : *vrijednost2*

Kontrola toka programa

- ***if else***

```
if(uslov_1)
        tijelo_1;
else if(uslov_2)
        tijelo_2;
else
        tijelo_3;

a = 5
if(a == 5) {
        document.write("Broj je pet");
}
else if(a > 5) {
        document.write("Broj je veci od pet");
}
else {
        document.write("Broj je manji od pet");
}
```

- ***switch***
  - Izraz u *switch()* mora proizvesti cjelobrojnu vrijednost - u suprotnom se mora koristiti *if*
  - Kod *default case*-a nije potreban *break*

```
switch(a) {
        case 1:
                tijelo_1;
                break;
        case 2:
                tijelo_2;
                break;
        default:
                tijelo_3;
}

a = 5;
switch(a) {
        case 5:
                document.write("Broj je pet");
        case 4:
                document.write("Broj je cetiri");
        default:
```

```
                document.write("Broj je nepoznat");
}
```

- ***while*/*do while***
  - Za cikličnu strukturu kod koje se samo zna uslov za prekid
  - *while*
    - Ne mora nijednom da se izvrši
  - *do while*
    - Bar jednom će se izvršiti

```
while(uslov) {
        tijelo_while;
}

do {
        tijelo_do_while;
}while(uslov);

a = 5;
while(a != 10) {
        a++;
}

a = 5;
do {
        document.write(a);
}while(a != 5);
```

- ***for***
  - Za cikličnu strukturu za koju je unaprijed poznat broj koraka

```
for(inicijalizacija; uslov; korak) {
        tijelo_for;
}

for(var i = 0; i < 10; i++) {
        document.write(i);
}
```

- ***break*/*continue***
  - *break* prekida tijelo tekuće ciklične strukture (ili *case* dijela) i izlazi iz nje
  - *continue* prekida tijelo tekuće ciklične strukture i otpočinje iteraciju sledeće petlje

Funkcije u JavaScript-u

- Definicija funkcije unutar `<head>` *tag*-a

```
<html>
        <head>
                <title> JavaScript <title>
                <script type="text/javascript">
                        function ispis() {
                                document.write("Drugi pasus, ali iz funkcije");
                        }
                </script>
        </head>

        <body>
                <p> Prvi pasus </p>
                <script language="JavaScript">
                        ispis();
                </script>
        </body>
</html>
```

Događaji u JavaScript-u

- Registruju i obrađuju se odgovarajućim *event hanlder*-ima
- Primjer:

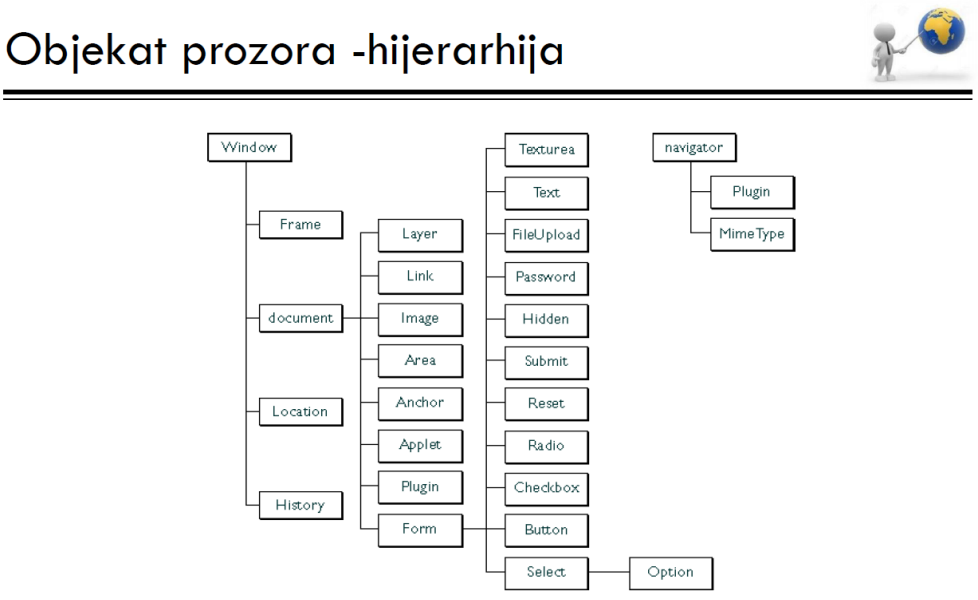
```
<body onload=funkcija()>
```

Događaji

Atribut	Događa se kada ...
onabort	se prekine učitavanje slike
onblur	element izgubi fokus
onchange	korisnik promeni sadržaj polja
onclick	se klikne mišem na objekat
ondblclick	se dva puta klikne po objektu
onerror	se dogodi greška prilikom učitavanja dokumenta ili slike
onfocus	element dobije fokus
onkeydown	se pritisne taster
onkeypress	se pritisne, pa otpusti taster, ili se drži pritisnut
onkeyup	se otpusti taster
onload	se stranica ili slika učita
onmousedown	se pritisne dugme miša
onmousemove	se miš pomera
onmouseout	miš izađe izvan zone elementa
onmouseover	miš pređe preko elementa
onmouseup	se otpusti dugme miša
onreset	se klikne na reset dugme
onresize	se prozoru ili frejmu promeni veličina
onselect	je tekst selektovan
onsubmit	se klikne na dugme subit u formi
onunload	korisnik napusti stranicu

- Pozivanje *JavaScript* funkcije kao reakciju na neki događaj - potrebno definisati tu funkciju u `<head>` *tag*-a
- Neke od ugrađenih funkcija:
  - *isNaN()*
    - Vraća *true* ako prosleđeni *string* nije broj
  - *eval()*
    - Interpretira prosleđeni *string* kao *JavaScript* kod
  - *parseInt(string)*
    - Parsira *string* u *integer*
    - Bitno kod formi jer vrijednosti koje se unose u polja se tretiraju kao *stringovi* - ovu funkciju koristimo ako nam treba broj koji ćemo dalje da koristimo u aritmetičkim funkcijama
  - *parseFloat(string)*
    - Parsira *string* u *float* promjenljivu
  - *alert()*
    - Ispis poruke u *MessageBox*-u
  - *escape()*, *unsescape()*
    - Kodira / dekodira *URL*-ove

Objekat prozora - hijerarhija



- ***Window* objekat**
  - Omogućuje manipulaciju prozorima
  - Sadrži informacije o tekućem prozoru
  - Metode:

        - *alert()*, *confirm()*, *prompt()* - poruka u prozoru (*MessageBox*)
        - *back()*, *forward()* - povratak na prethodnu stranicu/odlazak na sledeću - istorija odlazaka sa i dolazaka na stranicu
        - *moveBy()*, *moveTo()* - pomjera prozor
        - *open()* - otvara novi prozor
        - *setTimeout()*, *clearTimeout()* - podešava/isključuje kod koji će se izvršavati kad istekne *timeout*
        - *setInterval()*, *clearInterval()* - zadaje funkciju koja će se periodično izvršavati
    - Atributi:
        - *history* - istorija odlazaka sa i dolazaka na stranicu
        - *document* - tekući *HTML* dokument
        - *frames* - niz svih *frame*-ova u prozoru
        - *location* - kompletan *URL* tekuće stranice
        - *statusbar* - statusna linija na dnu ekrana
- ***Confirm dialog***
    - Dijalog za potvrdu - zavisno od klika na koje dugme, dobijaju se različiti *alert*-ovi
- ***Location* objekat**
    - Reprezentuje *URL* stranicu koja je učitana u navigator
    - Primjer: `location = "https://www.google.com"`
    - Sadrži informacije o tekućem dokumentu
    - Metode:
        - *reload()* - ponovno učitavanje tekućeg prozora
        - *replace()* - učitava novi *URL*
    - Atributi:
        - *href* - pun *URL* do stranice
        - *protocol* - protokol iz *URL*-a
        - *host* - adresa servera iz *URL*-a
        - *port* - port iz *URL*-a
        - *pathname* - putanja do resursa
        - *search* - parametri forme
- ***History* objekat**
    - Omogućuje kontrolu pristupa već posjećenim stranicama
    - Sadrži listu adresa već posjećenih stranica
    - Metode:
        - *back()* - učitava prethodnu stranicu iz liste
        - *forward()* - učitava sledeću stranicu iz liste
        - *go()* - učitava zadatu adresu iz liste

    - Atributi:
        - *current* - trenutno učitana adresa
        - *lenght* - broj stavki u *history* listi
        - *next* - zadavanje sledećeg elementa
        - *previous* - zadavanje prethodnog elementa
- ***Document*** **objekat**
    - Omogućuje ispis teksta na ekran
    - Sadrži informaciju o tekućem dokumentu
    - Metode:
        - *write()* - ispisuje tekst na ekran
    - Atributi
        - *forms* - niz svih formi u dokumentu
        - *links* - niz svih linkova u dokumentu
        - *applets* - niz svih apleta u dokumentu
        - *title* - sadržaj *title taga*
- ***String*** **objekti**
    - Reprezentuje *string*
    - Metode:
        - *substring()* - vraća dio *stringa*
        - *split(':')* - vraća niz stringova kao rezultat "razbijanja" *stringa*
        - *indexOf()*, *lastIndexOf()* - vraća poziciju nekog *podstringa*
    - Atributi:
        - *lenght* - dužina stringa
- **Forme**
    - Reprezentovane *form* objektom
    - Metode:
        - *submit()* - šalje podatke iz forme na odredište definisano *action* atributom *form taga*
        - *reset()* - simulira pritisak na *reset* dugme forme
    - Atributi:
        - *elements* - niz elemenata forme
            - Svaki element ima *value* atribut za pristup sadržaju
            - *lenght* - broj elemenata na formi
            - *action* - sadržaj *action* atributa
    - Primjeri:
        - Preuzimanje vrijednosti polja u *JavaScript*-u
        - Provjera da li je selektovan *checkbox* ili *radiobutton*

```
var vrijednost = document.forms['forma'].polje1.value;

var remember = document.forms['forma'].polje2;
if(remember.checked) {...}
else {...}
```

UML - modelovanje podataka

Modelovanje podataka

- Model sistema je prikaz bitnih funkcija nekog stvarnog (ili zamišljenog) sistema koji predstavlja obilježja sistema u upotrebljivom obliku
- Modeli se izrađuju u cilju boljeg razumijevanja sistema jer pomažu u vizualizaciji stvarnog ili zamišljenog sistema
  - Opisuju strukturu takvog sistema
  - Predstavljaju šablon po kom se sistem može implementirati
  - Dokumentuju sve korake izgradnje sistema
- Izbor vrste modela utiče na pristup problemu i na taj način oblikovanja rješenja
  - Nijedan model sam po sebi nije dovoljan - potrebno modelovati iz više raznih gledišta
- Za potrebe modelovanja sistema razvijen je jezik za modelovanje sistema ***UML*** (*Unified Modeling Language*)
  - Alat za vizualizaciju, opis, izgradnju i dokumentovanje softverske podrške kod analize i izrade prvenstveno softverskog rješenja
  - Rječnik i pravila za izražavanje znanja o modelovanom sistemu
  - Formalan jezik namijenjen za formalno specificiranje, modelovanje, analizu, vizualizaciju, dokumentovanje i razvoj (softverskih) sistema
  - Olakšava razmjenu informacija među učesnicima projekta budući da je usmjeren na višestruke komunikacije
  - Nudi skup dobro određenih grafičkih prikaza i dijagrama, razumljivih i projektantima i korisnicima sistema
  - Tehnički opis precizan, nedvosmislen i potpun
  - Zasnovan na principima objektne orijentacije
  - Polazni koncepti:
    - *Klasa*
      - Skup objekata koji imaju iste osobine (atribute) i funkcionalnosti (operacije), istu semantiku i zajedničke veze sa drugim objektima
      - Model skupa entiteta realnog sistema koji imaju zajedničku osobinu
    - *Objekat*

        - Pojava posmatrane klase koja ima određenu ulogu
        - Opisuje pojedinačni entitet, bio stvaran ili apstraktan, sa dobro definisanom ulogom u modelu problema
        - Model konkretnog entiteta realnog sistema
    - *Veza*
        - Model mogućih odnosa između klasa objekata i/ili samih objekata.
        - Osnovne vrste veze:
            - *Asocijacija* (relacija udruživanja)
            - *Agregacija* i *kompozicija* (relacije sastavljanja)
            - *Zavisnost*
            - *Generalizacija* (relacija uopštavanja)
    - *Poruka*
        - Oblik moguće interakcije između objekata
        - Poziv operacije klase nad jednim objektom klase
- Dijagrami za modelovanje se razvijaju alatima za vizuelno modelovanje kao što su *Enterprise Architect*, *Power Designer*, *IBM Rational Rose*,...

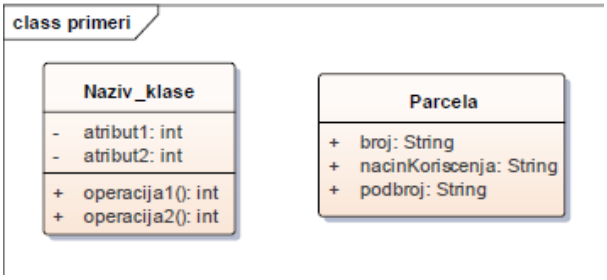
Tipovi dijagrama

- Podjela se vrši na osnovu toga da li opisuju statičku strukturu sistema ili ponašanje sistema
- ***Dijagrami strukture***
    - *Dijagram klasa*
        - Opisuje statički pogled na model kroz klase i veze između njih
    - *Dijagram objekata*
        - Specijalni slučaj dijagrama klasa i naglašava vezu između instanci klasa u nekom trenutku vremena
    - *Dijagram komponenti*
        - Pokazuje komponente sistema, osnovne gradivne elemente sistema
    - *Dijagram paketa*
        - Koristi se za podjelu modela u logičke cjeline i prikaz veza između tih cjelina
    - *Dijagram rasporeda*
        - Modeluje fizički sistem sistemskog mapiranja softverskih elemenata na hardver u kojem se oni izvršavaju i njihovu komunikaciju
    - *Dijagram kompozitne strukture*
        - Pokazuje unutrašnju strukturu klase i saradnje koje je omogućavaju i opisuju
- ***Dijagrami ponašanja***
    - *Dijagram slučajeva korišćenja*

        - Služi za specificiranje poslovne funkcije sistema
        - Opis niza akcija koje sistem izvršava prilikom realizacije poslovne funkcije, a koja se pokreće na zahtjev korisnika
    - *Dijagram aktivnosti*
        - Služi da opiše redosljed izvršavanja aktivnosti u okviru jednog slučaja korišćenja
    - *Dijagram stanja*
        - Pokazuje ponašanje sistema kao odgovor na neki ulaz
        - Prikazuje stanja sistema i način kako sistem prelazi iz jednog u drugo stanje
    - *Dijagram poruka*
        - Modeluje interakcije između objekata ili dijelova u smislu sekvenciranih poruka
    - *Dijagram sekvenci*
        - Pruža grafički prikaz interakcija objekata tokom vremena
        - Jedan dijagram sekvence obično predstavlja jedan scenario slučaja korišćenja ili tok dijagrama
    - *Dijagram vremena*
        - Koristi se za prikaz promjene stanja ili vrijednosti jednog ili više elemanata modela tokom vremena
    - *Dijagram interakcije*
        - Koristi kontrole iz dijagrama aktivnosti kako bi se kontrolna logika postavila oko dijagrama nižih nivoa

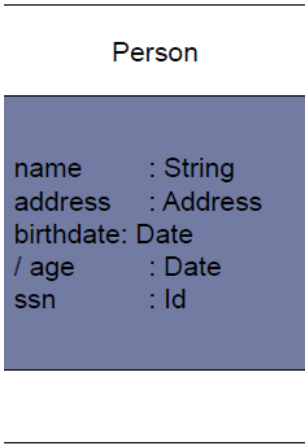
Dijagram klasa

- Opisuje tipove objekata nekog sistema i različite vrste statičkih odnosa koji postoje između tih objekata
- Grafička reprezentacija statičkog pogleda statičkih elemanata
- Najvažniji elementi *UML* dijagrama klasa su:
    - *Klase*
        - Opisi skupa objekata koji imaju slične atribute, operacije,...
        - Definiše svoje ponašanje korišćenjem operacija

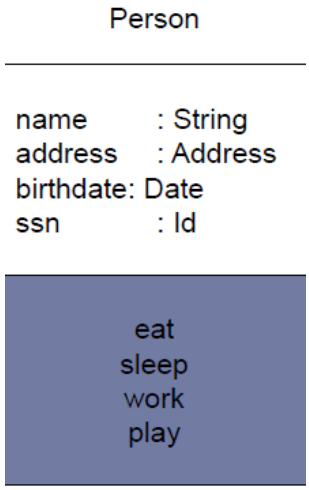


    - *Atributi*

- Osobine objekta (instance) klase
- Vidljivost atributa predstavlja pravo pristupa i odnosi se na:
  - *Javni*
    - Pristup nije ograničen, "+"
  - *Privatni*
    - Pristup ograničen na članove iste klase i podklase, "'-"
  - *Zaštićeni*
    - Privatno pravo pristupa prošireno na klase izvedene iz date klase, "#"
  - *Implementacioni*
    - Jedinstvena identifikacija klase realizovana označavanjem atributa za ključ uvođenjem znaka "**"
- Obično navedeni u formi *attributeName : Type*
  - Izvedeni atributi su oni koji se mogu izračunati iz drugih, ali zapravo ne postoje (označeni sa "/")

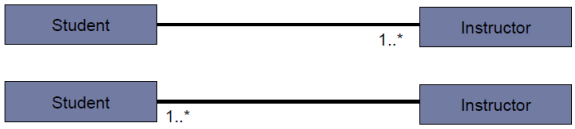


- *Operacije*
  - Opisuju ponašanje klase i pojavljuju se u trećem dijelu klase
  - Nazivaju se i metode

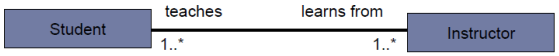


- *Odnosi*

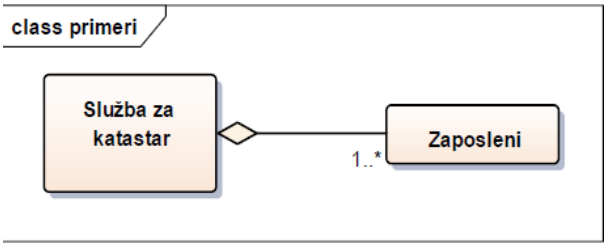
- Razlikuju se relacije:
  - *Asocijacije*
    - Specificiraju da je objekat neke klase u vezi sa objektom druge (moguće i iste) klase
    - Može imati svoje ime
    - Na oba kraja definiše se višestrukost ili multiplikativnost
      - Označava broj instanci klase i ukazuje na to da li je asocijacija obavezna ili ne



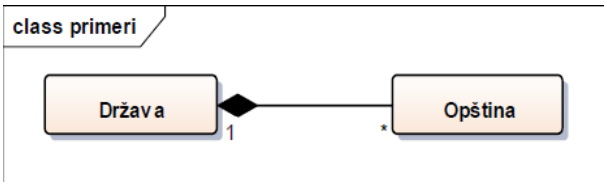
      - Sadrže neku ulogu (rolu) u relaciji između klasa



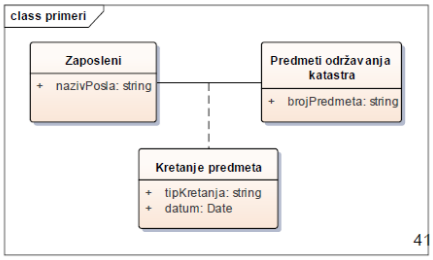
      - Postoje i usmjerene relacije koje označavaju da se komunikacija između objekata odvija samo u jednom pravcu
      - ***Ukoliko su dva objekta nezavisna, onda je relacija asocijacija***
  - *Agregacija*
    - Specijalan oblik asocijacija koji modeluje odnos "*cjelina - dio*"
    - Kod agregacije, dijelovi postoje nezavisno od cjeline



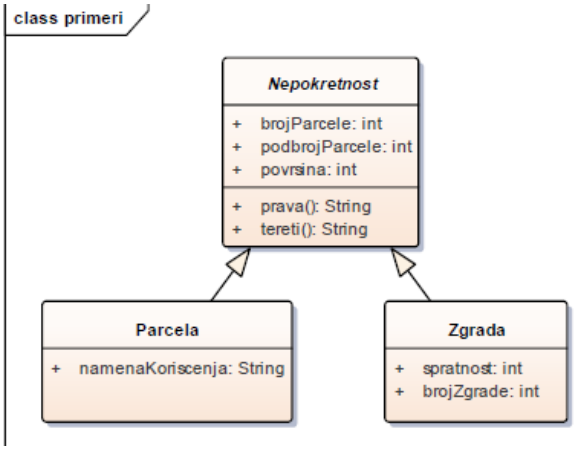
    - ***Ukoliko su dva objekta usko povezana relacijom cjelina - dio, onda je relacija agregacija***
  - *Kompozicija*
    - Jači oblik agregacije
    - Takođe modeluje odnos "*cjelina - dio*" s tim da je cjelina isključivi i jedini vlasnih njenih dijelova
    - Životni vijek dijela isključivo zavisi od cjeline



- *Asocijacije*
  - Klasa koja dozvoljava da asocijacija između dvije klase ima operacije i atribute
    - primer dodeljivanja predmeta za provođenje promena u katastru nepokretnosti zaposlenom: jednostavna asocijacija između dve klase nije dovoljna jer zaposleni vrši konkretan posao na predmetu koji je definisan tipom kretanja i datumom. Kako je ovo složen entitet jer sadrži detalje koji ne pripadaju ni klasi *Zaposleni* ni klasi *Predmeti održavanja katastra,* definše se nova klasa *Kretanje predmeta* koja ima ulogu asocijacije

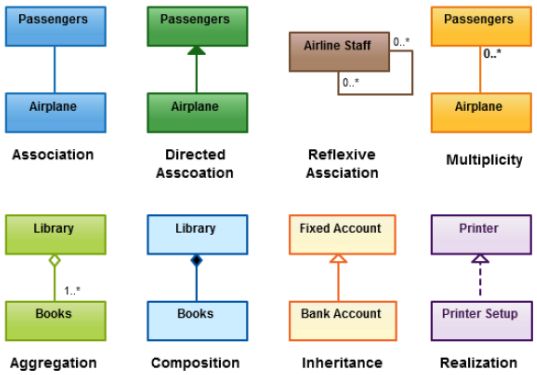


- *Generalizacije*
  - Odnosi se na nasljeđivanje i koristi se za prikazivanje odnosa "*roditelj - dijete*"
  - Podklase (*djeca*) nasljeđuju sve atribute, operacije i asocijacije od *superklase* (*roditelja*)
  - Podklasa može dodavati atribute i operacije, relacije, *override*-uje nasljeđene operacije



- *Zavisnosti*
- *Realizacija*
- *Pravila ograničenja i zapisi*

Moguće relacije na dijagramu klasa



TypeScript

- Nadskup *JavaScript*-a
- Izgrađen na *JavaScript*-u
- Kompajlira se u običan *JavaScript* kod upotrebom *TypeScript* kompajlera
- Koristi ekstenziju "*.ts*" umjesto "*.js*" ekstenzije
- Osnovni ciljevi *TypeScript*-a su:
  - Uvođenje opcionih tipova u *JavaScript*
    - Poboljšava produktivnost i istovremeno pomaže u izbjegavanju grešaka
  - Implementiranje planiranih svojstava budućeg *JavaScript*-a, zvanog *ECMAScript Next* u trenutni *JavaScript*
    - Mogućnost korišćenja novih *JavaScript* funkcionalnosti prije nego što ih *web browser*-i podrže

Tipovi u TypeScript-u

- Za razliku od dinamičko tipiziranog *JavaScript*-a, *TypeScript* uvodi opcioni sistem tipova da bi riješio problem *bug*-ova
- Tip je labela koja opisuje atribute i metode koje data vrijednost ima
- Kategorizacije tipova u *TypeScript*-u:
  - *Osnovni tipovi*
    - *String*, *Number*, *Boolean*, *Null*, *Undefined*,...
  - *Objektni tipovi*
    - *Nizovi*
      - metode *lenght()*, *forEach()*, *map()*, *reduce()*, *filter()*, *push()*

        - *Tuple*
            - Slični nizovima, samo što je element u *tuple*-u fiksiran
            - Tipovi elemenata su poznati i ne moraju biti isti
        - *Enum*
            - Grupa konstantnih vrijednosti koja ima ime
        - *Any*
            - Omogućava da se smjesti vrijednost bilo kog tipa
            - Daje instrukciju kompajleru da preskoči provjeru tipa
        - *Never*
            - Ne sadrži vrijednost
            - Predstavlja povratni tip funkcije koja uvijek baca grešku ili sadrži beskonačnu petlju
        - *Union*
            - Omogućava da se smjesti vrijednost jednog ili više tipova u promjenljivu
        - *object* i *Object*
            - Tip *object* predstavlja sve neprimitivne vrijednosti
            - Tip *Object* opisuje funkcionalnost svih objekata
                - Posjeduje metode *toString()* i *valueOf()*
    - *Napredni tipovi*
        - *Intersection types*
            - Tip presjeka koji kombinuje dva ili više tipova da bi napravio novi tip koji ima svojstva postojećih tipova
        - *Type Guards*
            - Sužavaju tip promjenljive unutar uslovnog bloka
            - Koriste se operatori *retypeof* i *instanceof* da bi se implementirali "čuvari" tipova uslovnom bloku
        - *Type Casting*
            - Konverzija promjenljive iz jednog tipa u drugi
            - Koristi se ključna riječ *as* ili operator <>
- Svrhe tipova su:
    - Kompajler ih koristi da analizira kod na greške
    - Omogućavaju da se razumije kakve vrijednosti su pridružene promjenljivim
- Anotacija tipova:

```
// Primitivni tipovi
let variable_name: type = value;
const variable_name: type = value;
```

```
let name: string = 'John';
let age: number = 25;
let active: boolean = true;
###################################
// object tipovi
let person: {
        name: string;
        age: number;
}

person = {
        name: "John";
        age: 25;
}
###################################
// Funkcije
let greeting : (name: string) => string;

greeting = function(name: string) {
        return 'Hi ${name}'
}

function add(a: number, b: number): number {
        return a + b;
}
###################################
// Nizovi
let names: string[] = ['John', 'Jane', 'Mary']
let series = [1, 2, 3]
let doubleIt = series.map(e => e * 2)
doubleIt.push(8)
###################################
// Tuple-ovi
let skill: [string, number];
skill = ['Programming', 5]
###################################
// Enum-i
enum months {'January', 'February', 'March'}
###################################
// Any tip
let result: any;
result = 10.123;
###################################
//Union tip
let result: number | string;
result = 10;
```

```
result = 'Hi';
###################################
// Intersection tip
type type_AB = type_A & type_B;
###################################
// Type Guards
if(typeof a === 'string' && typeof b == 'string') {
        return a.concat(b);
}
###################################
let a: type_A;
let b = a as type_B;
let c = <type_C> a;
```

- Ako se tip eksplicitno ne anotira, *TypeScript* kompajler može zaključiti tip iz dodjeljene vrijednosti prilikom inicijalizacije
- *TypeScript* koristi *The best common type algorithm* za odabir kandidata tipova koji su kompatabilni sa varijablom
- Dobra praksa je koristiti eksplicitnu anotaciju

Kontrola toka programa

- Isti mehanizmi kao i u *JavaScript*-u (pogledati)

Funkcije

- *Funkcije*
  - Pređeno u **Tipovima**
- *Function* tipovi
  - Pređeno u **Tipovima**
- *Opcioni parametri*
  - Sintaksa `parameter?:type` omogućava da parametar bude opcion
  - Izrazom `typeof(parameter) !== 'undefined'` provjerava se da li je parametar bio inicijalizovan
- *Default parametri*
  - Sintaksa `parameter:=default_value` se koristi da se inicijalizuje *default* vrijednost za parametar
  - Suštinski podvrsta opcionih parametara
- *Rest parametri*
  - Omogućavaju da se beskonačan broj argumenata predstavi kao niz
  - *Rest* parametar se pojavljuje poslednji u listi parametara

    - Tip *rest* parametra je niz
- *Function overloading*
    - Omogućava da se uspostavi veza između tipova parametara i rezultujućeg tipa funkcije
    - Primjer: ako su parametri *number*, funkcija treba da vrati *number* ; ako su parametri *string*, funkcija treba da vrati *string*

Klase

- *JavaScript* ne posjeduje koncept klase
- *TypeScript* omogućava kreiranje klasa
- *Modifikatori pristupa*
    - *private*, *protected*, *public*
- *Getteri/setteri*
    - Koriste se da kontrolišu pristup atributima klase kroz *get* i *set* metode koje čitaju i mijenjaju vrijednost atributa, respektivno.
    - Poznati i pod imenom *accesors/mutators*
- *Nasljeđivanje*
    - Kroz ključnu riječ *extends*
    - Metoda *super()* koristi se u konstruktoru klase nasljednika da pozove konstruktor roditeljske klase
- *Statički atributi*
    - Atributi ili metode dijeljeni između instanci svih klasa
- *Apstraktne klase*
    - Ne mogu se instancirati
- *Interfejsi*
    - Definišu ugovore u kodu i pružaju eksplicitne nazive za provjeru tipova
    - Mogu biti korišćeni kao tipovi funkcije
    - Proširuju (*extends*) klase
        - Interfejs nasljeđuje atribute i metode klase
    - Mogu biti generički
    - *Interfejsi se implementiraju, ne proširuju!*
- *Generičke klase*
    - Imaju listu generičkih tipova parametara uglastim zagradama <> koji slijede iza naziva klase

```
// Klase i nasljeđivanje
class Person {
        private ssn: string;
```

```
        private first_name: string;
        private last_name: string;

        protected nickname: string;

        constructor(ssn:string, first_name:string, last_name:string,
nickname:string){
                this.ssn = ssn;
                this.first_name = first_name;
                this.last_name = last_name;
                this.nickname = nickname;
        }

        public get_full_name(): string {
                return '${this.first_name} ${this.second_name}';
        }
}

class Employee extends Person {
        private static headcount: number = 0;

        constructor(first_name:string, last_name:string){
                super(first_name, last_name)
        }
}
###################################
// Apstraktne klase i interfejsi
abstract class Shape {
        get Area(): number {}
}

interface Json {
        toJSON():string
}
###################################
// Generičke klase
class Stack<T> {
        private elements: T[] = [];

        constructor(private size: number) {}

        is_empty(): boolean {
                return this.elements.lenght === 0;
        }

        is_full(): boolean {
```

```
                return this.elements.lenght === this.size;
        }

        push(element: T): void {
                if(this.elements.lenght === this.size) {
                        throw new Error('Stack overflow!');
                }
                this.elements.push(element);
        }
}
```

TypeScript moduli

- Modul se izvršava unutar sopstvenog opsega
- Deklaracija promjenljivih, funkcija, klasa i sl. van modula izvodimo ključnom rječju *export*
- Ako želimo pristupiti promjenljivim, funkcijama, klasama i sl. nekog drugog modula, koristimo ključnu riječ *import*

```
export interface Validator {
        isValid(s: string): boolean;
}
###################################
import { Validator } from './Validator';

class EmailValidator implements Validator {...}
```