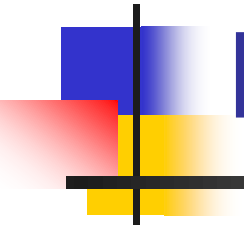
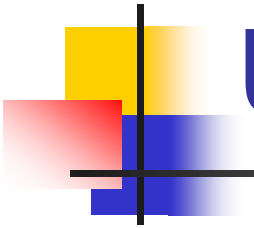


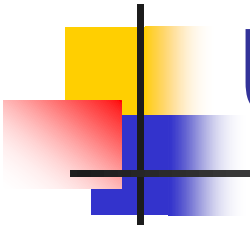
Baze podataka

Rad sa relacionim bazama podataka u programskom jeziku Java



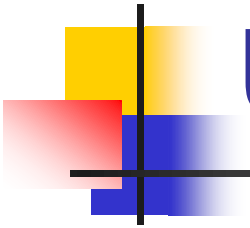
Uvod

- JDBC (Java Database Connectivity) je standardni Javin interfejs za komunikaciju sa relacionim bazama podataka
- Definiše skup klasa i interfejsa koji se koriste za pristup bazama podataka putem jezika SQL



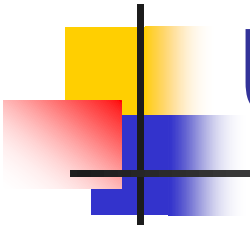
Uvod

- Za komunikaciju sa serverima za baze podataka koristi se TCP/IP familija protokola
- Za pristup konkretnoj bazi podataka putem JDBC-a potrebna je odgovarajuća biblioteka (obično zapakovana u JAR fajl) – to je JDBC drajver



Uvod

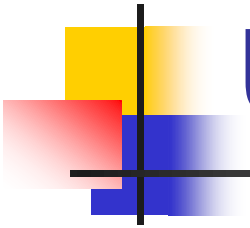
- Za Oracle 8i bazu podataka postoji Oracle-ov JDBC drajver – classes12.jar
- On predstavlja implementaciju JDBC 2.0 interfejsa za pristup bazi pomoću Java SDK-a verzije 1.2 ili kasnije
- Prednost ovakvog pristupa – za pristup drugoj bazi podataka pomoću njenog JDBC drajvera kod se ne bi mijenjao, osim u slučaju upotrebe nestandardnih SQL naredbi



Uvod

- Sve JDBC klase i interfejsi su definisani u standardnom paketu *java.sql*
- Za upotrebu JDBC drajvera (ili više njih) je dovoljno da on (oni) budu uključeni u CLASSPATH
- Uobičajeno je da proizvođač RDBMS-a nudi JDBC drajver za svoj sistem
- Tri bitne verzije JDBC interfejsa: 1.22, 2.0, 3.0

Uspostavljanje veze sa bazom podataka



Učitavanje Oracle drajvera

- Da bi se uspostavila veza sa nekom bazom podataka putem JDBC-a, neophodno je:
  - učitavanje drajvera
  - otvaranje konekcije sa bazom podataka

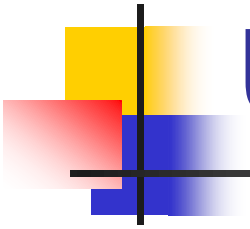
Učitavanje Oracle drajvera

- Učitavanje Oracle-ovog drajvera:

```
Class.forName("oracle.jdbc.driver.OracleDriver");
```

- **`forName`** je poziv statičke metode klase **`Class`**
- Instance klase **`Class`** predstavljaju klase i interfejse pokrenute Java aplikacije
- Statička metoda **`forName`** vraća inicijalizovan objekat klase **`Class`** koji odgovara klasi čiji naziv je dat parametrom metode

Učitavanje Oracle drajvera



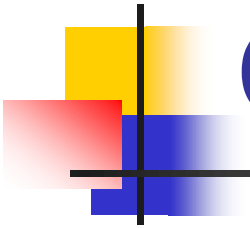
- Klasa **`OracleDriver`** je osnovna klasa Oracle-ovog drajvera
- Njeno učitavanje pozivom metode **`forName`** poziva metodu **`registerDriver`** klase **`DriverManager`** čime se konkretan drajver registruje na jedinstvenom mjestu

Otvaranje konekcije

- Otvaranje konekcije nakon što je drajver učitan

```
Connection conn =
  DriverManager.getConnection(
  "jdbc:oracle:thin:@adresa:1521:SID",
  "username", "password");
```

- Prvi parametar metode **`getConnection`** je string koji sadrži podatke potrebne drajveru za povezivanje sa SUBP

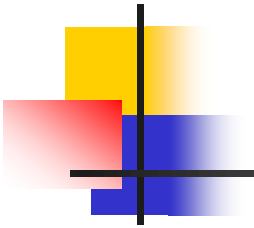


Otvaranje konekcije

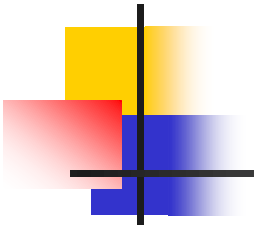
- Taj string obično sadrži:
  - adresu računara na kom se nalazi SUBP (localhost, ako je SUBP na lokalnom računaru; etf.ac.me)
  - TCP port na kom SUBP očekuje klijente (1521)
  - naziv baze kojoj se pristupa (baza)

Otvaranje konekcije

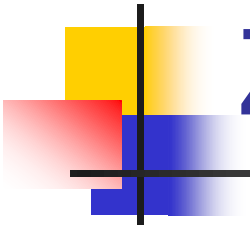
- Druga dva parametra metode **`getConnection`** su korisničko ime (baze2) i lozinka (kljuc) kojima se vrši prijavljivanje na SUBP



- Rezultat uspješnog uspostavljanja veze sa SUBP je inicijalizovani **`Connection`** objekat
- **`Connection`** je interfejs i ne može imati svoje instance, ali ovdje se radi o instanci klase koja implementira **`Connection`** interfejs
- U slučaju Oracle drajvera, ta klasa je **`OracleConnection`**, koja sadrži kod koji je specifičan za komunikaciju sa Oracle serverima

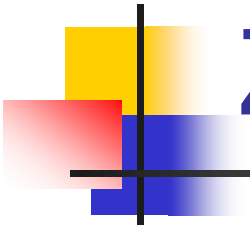


- Metoda **`forName`** može da izazove izuzetak **`ClassNotFoundException`**
- metoda **`getConnection`**, kao i sve ostale JDBC metode, može da izazove izuzetak **`SQLException`**
- Pozivi metoda moraju biti smješteni u **`try/catch`** blok
- Prilikom završetka komunikacije sa bazom, potrebno je zatvoriti otvorenu konekciju pozivom metode **`close()`** klase **`Connection`**



Zaključak

- Konačan rezultat ovakvog koncepta je program koji operiše nad objektima kojima pristupa preko njihovih standardnih interfejsa
- Konkretne klase koje implementiraju te interfejse ne spominju se nigdje u programu



Zaključak

- Jedino mjesto gde se specificira koja biblioteka klasa će biti upotrijebljena je učitavanje drajvera i otvaranje konekcije sa SUBP
- Isti JDBC programski kod se može koristiti sa različitim SUBP zamjenom drajvera

Microsoft Access baza

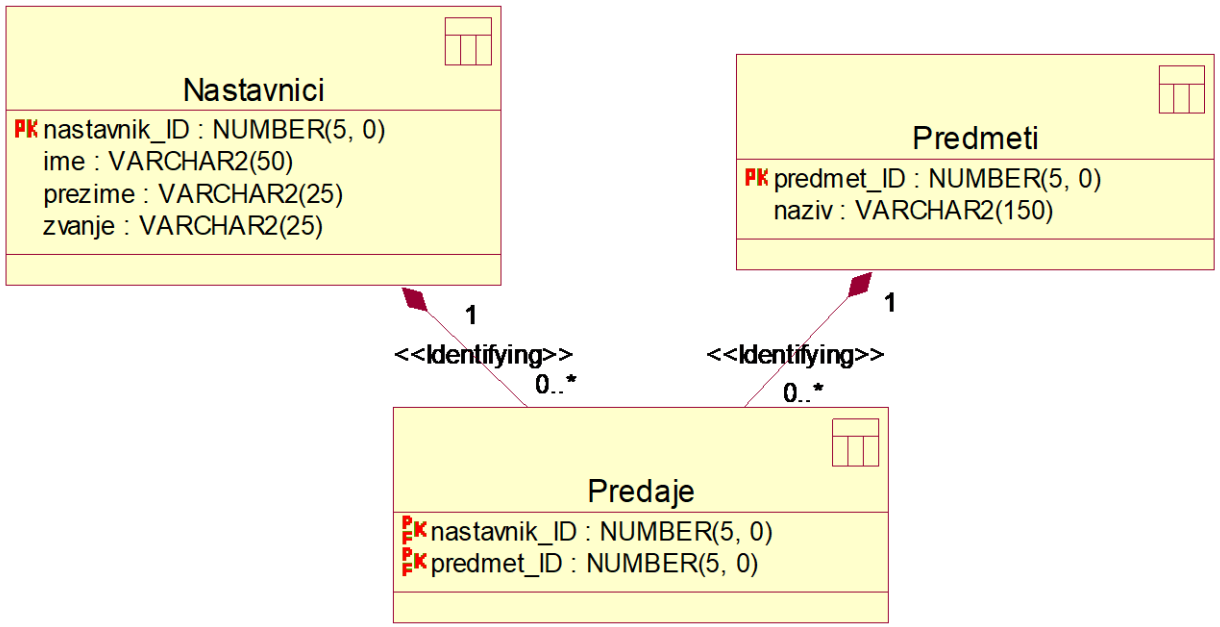
- Primer povezivanja na Microsoft Access bazu:
  - učitavanje drajvera

```
Class.forName("sun.jdbc.odbc.
  JdbcOdbcDriver");
```

  - otvaranje konekcije

```
Connection conn =
  DriverManager.getConnection(
  "jdbc:odbc:baza", "", "");
```

Postavljanje upita


Nastavnici
PK nastavnik_ID : NUMBER(5, 0)
ime : VARCHAR2(50)
prezime : VARCHAR2(25)
zvanje : VARCHAR2(25)
Predmeti
PK predmet_ID : NUMBER(5, 0)
naziv : VARCHAR2(150)
1
<<Identifying>>
0..*
1
<<Identifying>>
0..*
Predaje
PK FK nastavnik_ID : NUMBER(5, 0)
PK FK predmet_ID : NUMBER(5, 0)

Postavljanje upita

- SQL naredba je, u okviru JDBC drajvera. definisana Statement objektom
- On se kreira nakon uspostavljene veze iskazom:

```
Statement stmt =
  conn.createStatement();
```

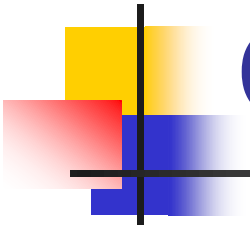
- **`conn`** je inicijalizovani **`Connection`** objekat

Slanje upita serveru

- Za slanje upita serveru koristi se metoda **`executeQuery`** čiji parametar je string koji sadrži tekst SQL upita

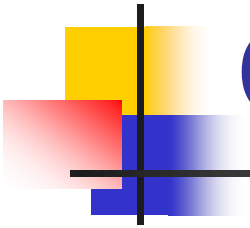
```
ResultSet rset =
   stmt.executeQuery("SELECT ime,
   prezime from Nastavnici");
```

- Rezultat ove metode je inicijalizovani objekat klase ResultSet, koji je namijenjen za skladištenje rezultata upita u vidu pomoćne tabele



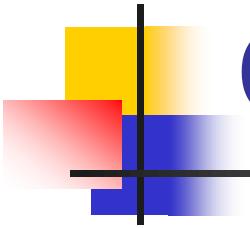
Čitanje rezultata

- Čitanje rezultata upita se odvija red-po-red, a za kretanje kroz zamišljenu tabelu se koristi koncept tekućeg reda
- Verzija 1.22 *JDBC* interfejsa dozvoljava pomijeranje tekućeg reda od početka, ka kraju tabele, bez preskakanja i bez više prolaza



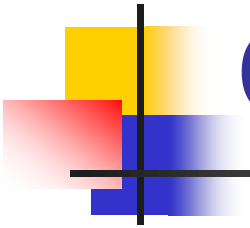
Čitanje rezultata

- Verzija 2.0 *JDBC* interfejsa omogućava prolazak kroz tabelu u oba smjera – veća fleksibilnost
- Po otvaranju upita nijedan red se ne proglašava tekućim do poziva metode **`next()`** klase **`ResultSet`**
- Ukoliko tabela nije prazna, prvi red postaje tekući



Čitanje rezultata

- **`next()`** vraća *boolean* vrijednost koja označava da li novi tekući red postoji ili ne
- Za tekući red rezultata, pojedine vrijednosti polja se čitaju pomoću metoda **`getString()`, `getInt()`, `getDate()`,** itd. (za svaki tip podatak u bazi postoji odgovarajuća metoda)
- Parametar **`getXXX()`** metoda je redni broj kolone koja se čita
- Redni brojevi počinju od 1!



Čitanje rezultata

- Za konverziju između tipova podataka baze i jezika *Java* je zadužen *JDBC* drajver
- Nakon prestanka korišćenja objekta **`ResultSet`**, potrebno je pozvati metodu **`close()`** radi oslobađanja resursa koje je objekat zauzimao
- Ukoliko se prethodna dva pravila ne poštuju može doći do grešaka koje se teško otkrivaju