Kreiranje modela tabele u Javi – JTable

Poetni problem

Ukoliko bismo na GUI formu jednostavno postavili komponentu JTable, pokrenuli program, a zatim pokušali da popunimo postavljenu tabelu, vrednosti bi prividno bile unete, ali samo dok ne napustimo eliju u kojoj se trenutno nalazimo. Nakon toga, sadržaj elije bi se izbrisao.
Kako rešiti ovaj problem i kako formatirati tabelu (recimo, napraviti neko zaglavlje)? Odgovor leži u definisanju modela tabele, tanije u klasi koja e opisivati model tabele. Ukoliko pogledamo "properties" za našu tabelu, videemo da se jedno od navedenih svojstava ove komponente zove "model". Trenutno se u ovom polju nalazi default vrednost koja generiše izgled tabele koji se dobija inicijelnim postavljanjem komponente JTable na formu. Nakon što napravimo naš model tabele, ubaciemo ga upravo u ovaj property.
Svi kodovi u ovom tekstu e se odnositi na primer programa za kladionicu, tanije, formatiranje i popunjavanje tiketa. Ovaj program neemo u potpunosti napisati, ve emo se pozabaviti samo onim delovima koji su obuhvaeni temom ovog teksta.

Naš model – nasleivanje i apstraktne metode

Kao što smo rekli, naš model e zapravo biti klasa. Dakle, prvo treba da napravimo klasu ModelTabeleTiket koja e opisivati model naše tabele. Predpostaviemo da treba da napravimo tabelu koja e predstavljati tiket za kladionicu sa kolonama, recimo Šifra utakmice i Tip. Za naziv vrste postaviemo Utakmica + redni broj vrste: Utakmica 1, Utakmica 2 itd. Pošto ne želimo da korisnik menja naziv zaglavlja (prva vrsta i kolona), omoguiemo mu da upisuje vrednosti samo u elije tabele koje se ne nalaze u zaglavlju.
Prvo što treba da uradimo kako bismo mogli da predstavimo našu klasu kao model tabele je da postavimo da naša klasa ModelTabeleTiket nasleuje (extends) klasu AbstractTableModel ili DefaultTableModel. Nakon ovog koraka, dobiemo obaveštenje da moramo da implementiramo tri apstraktne metode iz nadklase, i to: getRowCount(), getColumnCount() i getValueAt(int rowIndex, int columnIndex).
Metoda getColumnCount(), kao što joj ime kaže, treba da vraa broj kolona tabele. U našem sluaju kolone su unapred poznate, a to su: kolona zaglavlja, kolona Šifra utakmice i kolona Tip. Naša tabela e uvek imati tano 3 kolone. Zato u telu ove metode pišemo samo jednu liniju koda:
return 3;
Sledea metoda koju je potrebno implementirati je metoda getRowCount(). Ovde se pria samo prividno komplikuje: neko igra singl tiket (jedna utakmica), neko se kladi na 5 utakmica, neko na 11 itd. Drugim reima, broj redova je promenljiv od sluaja do sluaja. Kao što smo rekli, komplikacija je samo prividna, ovaj problem se jako lako može rešiti na više naina.
Recimo, možemo da postavimo na poetnu formu komponentu JTextField koja e predstavljati broj utakmica na koje neko želi da se kladi i da se ta vrednost uva u recimo: public static int brojUtakmica. U ovom sluaju, u telu metode getRowCount() upisujemo nešto kao:

return FmPocetna.brojUtakmica;

Ono što je bitno naglasiti je da ovde nismo dodali još jedan red za red zaglavlja, jer komponenta JTable po default-u izdvaja jedan red koji služi kao zaglavlje.

Pre nego što ponemo sa implementacijom metode getValueAt, treba da definišemo neku strukturu u kojoj emo uvati vrednosti unete u tabelu. Obzirom da radimo sa tabelom, najpogodnije je da koristimo matricu (dvodimenzionalni niz) tipa String, jer unete vrednosti u koloni Tip mogu biti 1, 2, ali i X, 3+, 1-1, X-2 itd. Dakle, definišemo globalnu promenljivu String[ ][ ] data. Zatim, pravimo konstruktor klase ModelTabeleTiket u kojem emo iskoristiti dve metode koje smo upravo implementirali:

```
public ModelTabeleTiket () {
  data = new String[getRowCount()][getColumnCount()];
}
```

Sada možemo da implementiramo metodu getValueAt(int rowIndex, int columnIndex). Ono što možemo da naslutimo iz potpisa metode je da ona služi da iz elije tabele koja se nalazi u preseku reda sa rednim brojem rowIndex i kolone sa rednim brojem columnIndex proita unetu vrednost, a zatim je i vrati. Inae, indeksi redova i kolona kreu od nule. Ovu metodu emo implementirati na sledei nain:

```
public Object getValueAt(int rowIndex, int columnIndex) {
  if(columnIndex == 0)
    return "Utakmica " + (rowIndex + 1);
  return data[rowIndex][columnIndex];
}
```

Ovom metodom smo ujedno i imenovali prvu kolonu tabele, tanije uneli smo vrednosti Utakmica 1, Utakmica 2 itd u elije prve kolone. Ali, sada se javlja novi problem – korisnik može da pristupa vrednostima ovih elija.

Naš model: override-ovanje metoda

Da bi smo rešili ovaj problem, potrebno je da izmenimo neka default ponašanja tabele, tanije da definišemo kojim elijama želimo da korisnik ima pristup, a kojim ne. Mi želimo da onemoguimo korisniku pristup vrednostima prve kolone, pa treba da override-ujemo metodu isCellEditable(int rowIndex, int columnIndex):

```
@Override
public boolean isCellEditable(int rowIndex, int columnIndex) {
  if(columnIndex > 0)
    return true;
  return false;
}
```

Kao što smo rekli, indeksi redova i kolona kreu od nule, pa “nulta kolona” predstavlja zapravo prvu kolonu naše tabele.

Kada smo poeli, rekli smo da želimo da postavimo imena kolona. To postižemo override-ovanjem metode getColumnName(int column):

```
@Override
public String getColumnName(int column) {
  if(column == 0) return ""; // prva kolona je deo zaglavlja
  if(column == 1) return "Šifra utakmice";
  if(column == 2) return "Tip";
  return "nikada"; // ova mogunost se nee ostvariti nikada, pa
} // je potpuno nebitno šta se vraa u tom sluaju
```

And the last, but not the least… metoda koja uva unete vrednosti i upisuje ih u matricu data. I za ovu metodu možemo da override-ujemo jednu ve postojeu metodu i to: setValueAt(Object aValue, int rowIndex, int columnIndex):

```
@Override
public void setValueAt(Object aValue, int rowIndex, int columnIndex) {
  String value;
  String a = "";
  if(aValue instanceof String)
    a = (String)aValue;
  if(a.length()== 0)
   value = "";
  else
   value = a;
  data[rowIndex][columnIndex] = value;
}
```

Ova metoda uzima vrednost unetu u eliju tabele i unosi je u matricu data. Na ovaj nain se uneta vrednost uva dok program radi, jer metoda getValueAt pristupa odgovarajuem polju u matrici i ispisuje tu vrednost na ekranu u odgovarajuoj eliji tabele. Ukoliko želimo da trajno uvamo vrednosti unete u tabelu, to možemo rešiti, na primer, uvoenjem opcije “Save” u naš program, ijom aktivacijom bismo vrednosti iz matrice data upisali u neku bazu podataka, XML fajl ili obian .txt (ili neki drugi) fajl.

Primena našeg modela

Pre nego što proglasimo posao završenim, treba da odemo u klasu u kojoj se nalazi komponenta JTable i u kodu pronaemo konstruktor ove klase. Ukoliko nam se instanca klase JTable koja predstavlja tiket zove svojim default imenom – jTable1, u konstruktor upisujemo:

```
jTable1.setModel(new ModelTabeleTiket());
```

Na ovaj nain smo naš model tabele proglasili zvaninim modelom tabele jTable1. Ukoliko i posle ovoga tabela u GUI builder delu izgleda isto, bez panike, promena na GUI-u e biti vidljiva tek kada se pokrene program.

P.S. U prilogu se nalaze zipovan source code i .jar fajl aplikacije koja je napravljena kao primer za sve opisano u ovom tekstu. Ova aplikacija kao takva ne služi niemu drugom sem kao ilustracija za kodove opisane u tekstu. Kao što je reeno, potrebno ju je doraditi, ubaciti makar opciju “Save” i dodati još neke funkcionalnosti, ali to se ostavlja na volju svakom itaocu ovog teksta. Kod je pisan

u Javi SE 1.7. Za pokretanje aplikacije, dovoljno je instalirati JRE za ovu verziju Jave, dok je za rad na kodu potreban JDK (ukljuen i JRE).