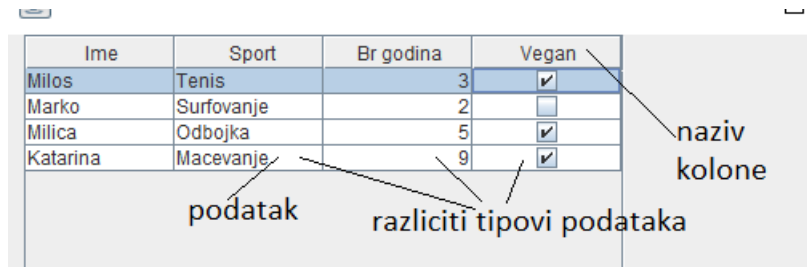


**JTABLE**

## ČEMU SLUŽI JTABLE?

Klasa JTable se koristi za prikaz tabele podataka, sa mogućnošću izmene podataka. Jtable ne sadrži i ne čuva podatke, već služi samo za prikaz.

Tipičan izgled tabele:



Ime	Sport	Br godina	Vegan
Milos	Tenis	3	<input checked="" type="checkbox"/>
Marko	Surfovanje	2	<input type="checkbox"/>
Milica	Odbojka	5	<input checked="" type="checkbox"/>
Katarina	Macevanje	9	<input checked="" type="checkbox"/>

Kreiranje tabele

Da bi se kreirala tabela korišćenjem JTable-a potrebno je najpre definisati niz tipa String koji će odgovarati nazivima kolona buduće tabele.

```
String[] naziviKolona={"Ime","Sport","Br godina","Vegan"};
```

A zatim I definisati dvodimenzionalni niz(Matricu) podataka:

```
new Object [][] {  
    {"Milos", "Tenis", new Integer(3), new Boolean(true)},  
    {"Marko", "Surfovanje", new Integer(2), null},  
    {"Milica", "Odbojka", new Integer(5), new Boolean(true)},  
    {"Katarina", "Macevanje", new Integer(9), new Boolean(true)}  
},
```

JTable sadrži dva različita konstruktora koji mogu odmah da prikažu podatke:

```
JTable(Object[][] podaci, Object[] naziviKolona);
```

```
JTable(Vector podaci, Vector naziviKolona);
```

Korišćenje ovih konstruktora je intuitivno jasno, ali njihovim korišćenjem, sva polja postaju Editable(promenljiva) i sva polja bez eksplicitnog navodjenja postaju tipa Object.

## Dodavanje tabele u Kontejner

Najčešće korišćen način za dodavanje tabele u gui je dodavanje u ScrollPane. To se izvršava sledećim naredbama:

**JScrollPane scrollPane = new JScrollPane(table);**//odgovarajući JScrollPane konstruktor prihvata tabelu kao objekat i odmah je postavlja u ScrollPane.

**table.setFillViewportHeight(true);**// postavljanje tabele u ceo scrollPane, celokupnu njegovu visinu

## Širine kolona

Što se vizuelnog dela tiče, veoma je bitno da svaka kolona ima svoju odgovarajuću širinu. Širina se uspostavlja pozivanjem ColumnModela odgovarajuće tabele(npr.

**table.getColumnModel().getColumn(0).setPreferredWidth(100)**-podešava prvu kolonu na širinu 100).

**Postoje tri operacije** : setPreferredWidth(podrazumevana širina),setMinWidth(minimalna širina),setMaxWidth(maksimalna širina).

Ukoliko se ne navede opcija setAutoResizeMode, širina kolona se neće menjati prilikom promene veličine prozora, ukoliko je prethodno eksplicitno definisana prethodno definisanim operacijama.

## Komande u Tabeli

Pri korišćenju tabele, za označavanje željenih polja, koristi se miš i tastatura. Tabela korišćenja prikazana je na slici:

Operacija	Miš	Tastatura
Izaberi red	Click.	Up Arrow or Down Arrow.
Proširi izbor	Shift-Click or Drag over rows.	Shift-Up Arrow or Shift-Down Arrow.
Predji na sledeci red	Click	Enter
Dodaj izabrani red	Control-Click	Move lead selection with Control-Up Arrow or Control-Down Arrow, then use Space Bar to add to selection or Control-Space Bar to toggle row selection.

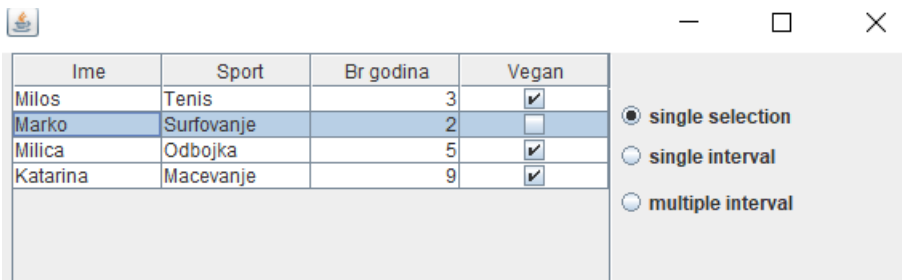
Takodje, može se i onemogućiti selektovanje više redova istovremeno. To se radi korišćenjem funkcije `setSelectionMode` na objekat tipa `JTable`.

```
JTable.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
```

Postoje tri različite opcije podešavanja:

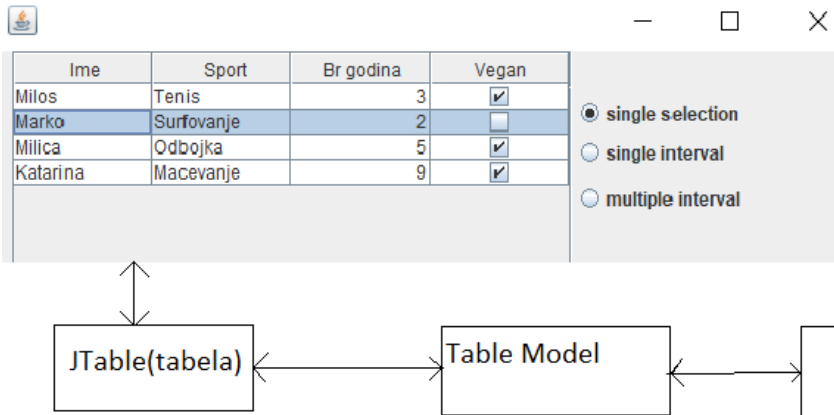
**SINGLE\_SELECTION, MULTIPLE\_INTERVAL\_SELECTION, SINGLE\_INTERVAL\_SELECTION.**

Funkcije poput **`setRowSelectionAllowed(Boolean)`** i **`setColumnSelectionAllowed(Boolean)`** omogućuju selekciju u pojedinačnim dimenzijama tabele. `CellSelectionAllowed` podrazumeva obe od prethodno navedenih funkcija u jednoj.



## Model Tabele

Kao što znamo, tabela ne čuva podatke niti ih ima, već ih samo prikazuje. Model Tabele povezuje podatke i tabelu u celinu. Ukoliko nije drugačije naveden, podrazumevani model tabele je `DefaultTableModel`.



Kreiranje novog **Modela Tabele** podrazumeva proširenje abstraktne klase **AbstractTableModel** i omogućuje navodjenje različitih svojstava tabele. Na slici je prikazano overrideovanje metoda klase **AbstractTableModel** koje nisu implementirane.

```
public class TableModel extends AbstractTableModel{
    String[] naziviKolona = {"Ime", "Sport", "Vegan", "Broj godina"};
    Object[][] podaci= new Object [][] {
        {"Milos", "Tenis", new Integer(3), new Boolean(true)},
        {"Marko", "Surfovanje", new Integer(2), null},
        {"Milica", "Odbojka", new Integer(5), new Boolean(true)},
        {"Katarina", "Macevanje", new Integer(9), new Boolean(true)}
    };

    @Override
    public int getRowCount() {
        return podaci.length;
    }

    @Override
    public int getColumnCount() {
        return 4; }

    @Override
    public Object getValueAt(int rowIndex, int columnIndex) {
        return podaci[rowIndex][columnIndex];
    }
}
```

Podešavanje table modela na tabelu se izvršava na sledeći način:

```
package jtable;

/**
 *
 * @author Milos <mm20160088@student.fon.bg.ac.rs>
 */
public class TableModeled extends javax.swing.JFrame {

    /**
     * Creates new form TableModeled
     */
    public TableModeled() {
        table.setModel(new TableModel());
        initComponents();
    }

    /**
     * This method is called from within the constructor to initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is always
     * regenerated by the Form Editor.
     */
}
```

Takođe, možemo, radi dodatnih zahteva, podesiti i druge proprietije.

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

@Override
public String getColumnName(int column) {
    return naziviKolona[column]; //To change body of generated methods, choose Tools | Templates.
}

public void setValueAt(Object value, int row, int col) {
    podaci[row][col] = value;
    fireTableCellUpdated(row, col);
}
```

## Promena podataka

Na prethodnoj slici, jasno se vidi da je, kod metode `setValueAt`, posle postavljanja vrednosti u matrici podataka, pozvana metoda `fireTableCellUpdated(row,col)`. To je jedna od metoda koja se koristi da bi promene u podacima (npr matrici iz modela podataka), bile prikazane u GULu tj u Tabeli (objektu `JTable`).

Metoda	Promena
<code>fireTableCellUpdated</code>	Promena određene ćelije
<code>fireTableRowsUpdated</code>	Promena reda
<code>fireTableDataChanged</code>	Osvezava podatke iz cele tabele.
<code>fireTableRowsInserted</code>	Unet novi red,
<code>fireTableRowsDeleted</code>	Obrisan postojeći red.
<code>fireTableStructureChanged</code>	Promena vrednosti i structure podataka.

## Iscrtavanje tabele

Iscrtavanje ćelija, tabela, izvršava se pomoću takozvanih cell renderera. Ukoliko su sve ćelije tipa `object`, možemo zamisliti cell renderer kao jedan isti kalup koji se otiskuje u prostoru određeni broj puta. Npr. ako su sve ćelije tipa `Integer`, renderer će iscrtavati veliki `JLabel` sa desnim poravnanjem. Za različite tipove podataka, renderer koristi različite kalupe za iscrtavanje.

**Boolean**- check box

**Number**- desno poravnan `JLabel`

**Double, Float**- isto kao **Number**, ali `object-to-text` prevođenje se vrši preko `NumberFormat` instance

**Date**- preko `JLabel`a, ali prevođenje se vrši preko `DateFormat` instance

**ImageIcon, Icon**-label

**Object**-label koji prikazuje `Object.toString()` vrednost

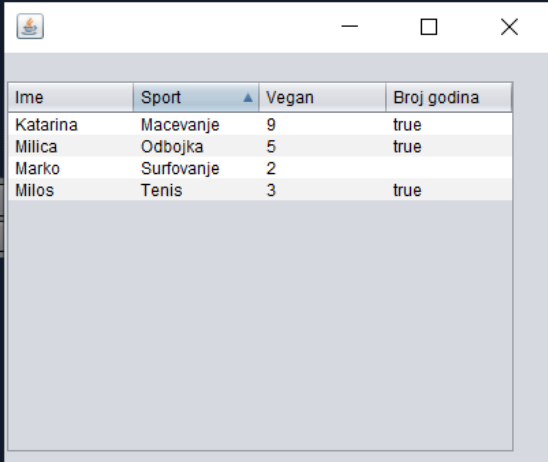
Promena defaultnog iscrtavanja vrši se metodom `setCellRenderer`.

Projektant može, za svoje potrebe da kreira klasu koja proširuje `DefaultTableCellRenderer` i u njoj kreira sopstveni način renderovanja ćelija u tabeli.

## Sortiranje tabele

Sortiranje i Filtriranje vrši se putem sorter objekta. Najjednostavniji način za sortiranje je da se na objekat tabele postavi `setAutoCreateRowSorter`. Ali, da bi mogli da koristimo sortiranje po svakoj koloni, koristimo instancu objekta **TableRowSorter** na sledeci način.

```
public class TableModeled extends javax.swing.JFrame {  
  
    /**  
     * Creates new form TableModeled  
     */  
    public TableModeled() {  
        initComponents();  
        table.setModel(new TableModel());  
        table.setAutoCreateRowSorter(true);  
        TableRowSorter<TableModel> sorter = new TableRowSorter<TableModel>((TableModel) table.getModel());  
        table.setRowSorter(sorter);  
    }  
}
```



Ime	Sport	Vegan	Broj godina
Katarina	Macevanje	9	true
Milica	Odbojka	5	true
Marko	Surfovanje	2	true
Milos	Tenis	3	true

Dodatno, **TableRowSorter** koristi **java.util.Comparator** da bi poredio tj sortirao redove. Korišćenjem Comparatora jasno možemo da definišemo način na koji će se podaci u tabeli sortirati.

## Akcije

Dodavanjem specijalizovanog `ActionListener`a na tabelu, možemo definisati događaje koji se aktiviraju određenim akcijama od strane korisnika. Najčešće korišćeni događaji nad tabelom su klik mišem na tabelu, davanje fokusa tabeli, klik dugmeta tastature kao i promene vrednosti tabele.

**Note:** Kodovi su dodati na [github repozitorijum](#).