DODATAK A

Upotreba dokumentacionih komentara u Javi

Kao što smo spomenuli u prvom delu ove knjige, Java podržava tri vrste komentara. Prve dve vrste označavaju se sa // i /* */. Treća vrsta se zove *dokumentacioni komentar* (engl. *documentation comment*). On počinje sekvencom /**, a završava se sa */. Dokumentacioni komentari omogućavaju da informacije o programu ugradite u sâm program. Pomoćni program **javadoc** (sastavni deo J2SE 5) koristi se za izvlačenje informacija i njihovo smeštanje u HTML datoteku. Dokumentacioni komentari olakšavaju dokumentovanje programa. Najverovatnije ste već videli dokumentaciju generisanu pomoću programa **javadoc**, budući da je firma Sun koristila taj program da napiše dokumentaciju za razvojno okruženje i osnovne klase Jave.

OZNAKE PROGRAMA JAVADOC

Pomoćni program **javadoc** prepoznaje sledeće oznake:

Oznaka	Značenje
@author	Određuje ime autora klase.
{@code}	Prikazuje podatke kakvi jesu, fontom kojim je pisan kôd, bez obrade HTML stilova. (Dodato u J2SE 5.)
@deprecated	Određuje zastarelost klase ili člana.
{@docRoot}	Određuje putanju do korenskog direktorijuma tekuće dokumentacije.
@exception	Određuje poruku o izuzetku koju vraća metoda.
{@inheritDoc}	Nasleđuje određeni komentar od prve prethodne natklase.
{@link}	Ubacuje unutrašnju vezu s drugim odeljkom.
{@linkplain}	Ubacuje unutrašnju vezu s drugim odeljkom, ali se veza prikazuje fontom običnog teksta.
{@literal}	Prikazuje podatke kakvi jesu, fontom kojim je pisan kôd, bez obrade HTML stilova. (Dodato u J2SE 5.)
@param	Opisuje parametar metode.
@return	Opisuje vrednost koju metoda vraća kao rezultat.
@see	Određuje vezu sa drugim odeljkom.
@serial	Opisuje podrazumevano serijalizovano polje.
@serialData	Opisuje podatke koje su ispisale metode **writeObject()** i **writeExternal()**.

Oznaka	Značenje
@serialField	Opisuje komponentu **ObjectStreamField**.
@since	Naznaka verzije u koju je uneta određena izmena.
@throws	Isto kao i **@exception**.
{@value}	Prikazuje vrednost određene konstante, koja mora biti statično (**static**) polje.
@version	Određuje verziju klase.

Oznake za dokumentaciju koje počinju znakom @ nazivaju se *samostalne* (engl. *stand-alone*) i moraju stajati u zasebnom redu. Oznake koje počinju vitičastom zagradom, kao {**@code**}, nazivaju se *unutrašnje* (engl. *in-line*); one se mogu stavljati i unutar većih opisa. Za dokumentacione komentare možete koristiti i druge standardne HTML oznake. Međutim, neke oznake, kao što su zaglavlja, ne bi trebalo koristiti pošto remete izgled HTML datoteke koju pravi program **javadoc**.

Dokumentacione komentare možete koristiti za opisivanje klasa, interfejsa, polja, konstruktora i metoda. U svakom slučaju, dokumentacioni komentar mora da bude neposredno ispred stavke na koju se odnosi. Kada opisujete promenljivu, možete da koristite dokumentacione oznake **@see**, **@since**, **@serial**, **@serialField**, {**@value**} i **@deprecated**. Za klase možete da koristite **@see**, **@author**, **@since**, **@deprecated**, **@param** i **@version**. Metode možete opisivati sa **@see**, **@return**, **@param**, **@since**, **@deprecated**, **@throws**, **@serialData**, {**@inheritDoc**} i **@exception**. Oznake {**@link**}, {**@docRoot**}, {**@code**}, {**@literal**} i {**@linkplain**} možete koristiti bilo gde. Sada ćemo objasniti svaku od ovih oznaka.

@author

Oznaka **@author** pruža informacije o autoru klase. Sintaksa izgleda ovako:

@author *opis*

Ovde je *opis* najčešće ime osobe koja je napisala klasu. Oznaka **@author** može da se koristi samo u opisivanju klasa. Ako želite da se polje **@author** pojavi u HTML dokumentaciji, moraćete da zadate opciju **-author** prilikom izvršavanja programa **javadoc**.

{@code}

Oznaka {**@code**} omogućava ugrađivanje teksta, recimo odlomka koda, u komentar. Taj tekst se potom prikazuje takav kakav jeste, fontom kojim je kôd pisan, bez ikakve dalje obrade, kao što bi se obrađivao HTML. Sintaksa izgleda ovako:

{@code *odlomakKoda*}

@deprecated

Oznaka **@deprecated** označava da su klasa ili član zastareli. Preporučuje se da stavite i oznaku **@see** ili {**@link**} da biste programere obavestili o raspoloživim alternativama. Sintaksa izgleda ovako:

@deprecated *opis*

Ovde je opis poruka koja opisuje zastarelost. Oznaku **@deprecated** možete da koristite za promenljive, metode i klase.

{@docRoot}

Oznaka {**@docRoot**} određuje putanju do korenskog direktorijuma tekuće dokumentacije.

@exception

Oznaka **@exception** opisuje izuzetak generisan u metodi. Sintaksa izgleda ovako:

@exception *ime-izuzetka objašnjenje*

Ovde *ime-izuzetka* predstavlja potpuno kvalifikovano ime izuzetka, a *objašnjenje* tekst koji objašnjava kako nastaje izuzetak. Oznaka **@exception** može da se koristi samo u opisivanju metoda.

{@inheritDoc}

Ova oznaka nasleđuje komentar od neposredno prethodne natklase.

{@link}

Oznaka {**@link**} obezbeđuje vezu prema dodatnim informacijama u samom redu programskog koda. Sintaksa izgleda ovako:

{@link *paket.klasa#član tekst*}

Ovde *paket.klasa#član* predstavlja ime klase ili metode za koju se dodaje veza, a tekst je tekst koji se prikazuje.

{@linkplain}

Umeće unutrašnju vezu ka drugoj temi. Veza se prikazuje fontom običnog teksta. U ostalom se ova oznaka ponaša kao oznaka {**@link**}.

{@literal}

Oznaka {**@literal**} ugrađuje tekst u komentar. Taj tekst se potom prikazuje takav kakav jeste, fontom kojim je kôd pisan, bez ikakve dalje obrade, kao što bi se obrađivao HTML. Sintaksa izgleda ovako:

{@litaral *opis*}

Opis označava ugrađeni tekst.

@param

Oznaka **@param** opisuje parametar metode ili naziv određenog parametra tipa generičke klase. Sintaksa izgleda ovako:

@param *ime-parametra objašnjenje*

Ovde *ime-parametra* predstavlja ime parametra metode. Značenje parametra opisuje *objašnjenje*. Oznaku **@param** možete koristiti samo za opisivanje metoda, konstruktora i generičkih klasa.

@return

Oznaka **@return** opisuje povratnu vrednost metode. Sintaksa izgleda ovako:

@return *objašnjenje*

Ovde *objašnjenje* opisuje vrstu i značenje vrednosti koju metoda vraća kao rezultat. Oznaka **@return** može da se koristi samo za opisivanje metoda.

@see

Oznaka **@see** obezbeđuje referencu do dodatnih podataka. Najčešće korišćeni oblici su:

@see *adresa*
@see *paket.klasa#član tekst*

U prvom obliku, *adresa* predstavlja vezu do apsolutne ili relativne URL adrese. U drugom obliku, *paket.klasa#član* određuje ime stavke, a *tekst* je tekst koji se prikazuje za tu stavku. Parametar *tekst* nije obavezan, a ako se ne koristi, prikazuje se stavka zadata s *paket.klasa#član*. Ime člana takođe nije obavezno. Prema tome, pored reference na određenu metodu ili polje možete zadati referencu na paket, klasu ili interfejs. Ime može biti potpuno kvalifikovano ili delimično kvalifikovano. Međutim, tačka koja se nalazi ispred imena člana (ako ono postoji) mora da se zameni znakom #.

@serial

Oznaka **@serial** definiše komentar za podrazumevano serijalizovano polje. Sintaksa izgleda ovako:

@serial *opis*

Ovde je *opis* komentar tog polja.

@serialData

Oznaka **@serialData** dokumentuje podatke koje su ispisale metode **writeObject**() i **writeExternal**(). Sintaksa izgleda ovako:

@serialData *opis*

Ovde je *opis* komentar tog podatka.

@serialField

Za klasu koja realizuje interfejs **Serializable**, oznaka **@serialField** obezbeđuje komentar komponente **ObjectStreamField**. Sintaksa izgleda ovako:

@serialField *ime vrsta opis*

Ovde je *ime* ime polja, *vrsta* njegova vrsta i *opis* je komentar tog polja.

@since

Oznaka **@since** opisuje da su klasa ili član uvedeni u određenoj verziji. Sintaksa izgleda ovako:

@since *verzija*

Ovde je *verzija* tekst koji određuje verziju u kojoj se osobina pojavila. Oznaka **@since** može da se koristi za opisivanje promenljivih, metoda i klasa.

@throws

Oznaka **@throws** ima isto značenje kao i oznaka **@exception**.

{@value}

Ova oznaka ima dva oblika. Prvi oblik prikazuje vrednost konstante kojoj prethodi, a koja mora biti statično polje. Taj oblik izgleda ovako:

{@value}

Drugi oblik prikazuje vrednost određenog statičnog polja i ima ovaj oblik:

{@value *paket.klasa#član*}

Ovde *paket.klasa#član* označava naziv statičnog polja.

@version

Oznaka **@version** određuje verziju klase. Sintaksa izgleda ovako:

@version *info*

Ovde je *info* tekst koji sadrži podatak o verziji, na primer, 2.2. Oznaka **@version** može da se koristi samo za opisivanje klasa. Ako želite da se polje **@version** pojavi u HTML dokumentaciji, moraćete da zadate opciju **-version** kada izvršavate program **javadoc**.

Opšti oblik dokumentacionog komentara

Posle /** kao početka, prvi red ili redovi postaju glavni opisi vaše klase, promenljive ili metode. Nakon toga, možete da dodate jednu ili više oznaka @. Svaka od njih mora da se nalazi na početku novog reda, tj. iza jedne ili više zvezdica na početku reda. Sve oznake iste vrste treba grupisati. Na primer, ako imate tri oznake **@see**, stavite ih jednu iza druge. Unutrašnje oznake (one koje počinju vitičastom zagradom) možete stavljati bez ikakvog tekstualnog opisa.

Ovo je primer dokumentacionog komentara klase:

```
/**
 * Ova klasa crta trakasti dijagram.
 * @author Herbert Schildt
 * @version 3.2
*/
```

Kako izgleda rezultat programa javadoc

Program **javadoc** uzima izvornu datoteku Java programa i pravi nekoliko HTML datoteka u kojima se nalazi programska dokumentacija. Informacije o svakoj klasi nalaziće se u zasebnoj HTML datoteci. Program **javadoc** takođe pravi indeks i hijerarhijsko stablo. Mogu se generisati i druge HTML datoteke.

PRIMER ZA DOKUMENTACIONE KOMENTARE

Sledi primer programa koji koristi dokumentacione komentare. Videćete da svaki komentar neposredno prethodi stavci koju opisuje. Kada je obradi program **javadoc**, dokumentacija klase **Kvadrat** biće u datoteci **Kvadrat.html**.

```
import java.io.*;

/**
 * Ova klasa pokazuje dokumentacione komentare.
 * @author Herbert Schildt
 * @version 1.2
*/
public class Kvadrat {
  /**
   * Ova metoda vraća kvadrat broja.
   * Ovaj opis ima više redova. Opis može da
   * ima proizvoljno mnogo redova.
   * @param broj Broj koji se diže na kvadrat.
   * @return kvadrat broja.
  */
  public double kvadriraj(double broj) {
    return broj * broj;
  }

  /**
   * Metoda traži da korisnik unese broj.
   * @return Uneti broj u pokretnom zarezu dvostruke tačnosti.
   * @exception IOException Ulazna greška.
   * @see IOException
  */
  public double unosBroja() throws IOException {
    // pravimo BufferReader pomoću System.in
    InputStreamReader isr = new InputStreamReader(System.in);
    BufferReader inData = new BufferReader(isr);
    String str;

    str = inData.readLine();
    return (new Double(str)).doubleValue();
  }
  /**
```

```
   * Ova metoda prikazuje funkciju kvadriraj().
   * @param args Ne koristi se.
   * @exception IOException Ulazna greška.
   * @see IOException
  */

  public static void main(String args[ ])
    throws IOException
  {
    Kvadrat ob = new Kvadrat();
    double vrednost;

    System.out.println("Unesite vrednost: ");
    vrednost = ob.unosBroja();
    vrednost = ob.kvadriraj(vrednost);

    System.out.println("Kvadrat broja je: " + vrednost);
  }
}
```