Biblioteke klasa

Jedan od karakterističnih aspekata razvoja softvera na objektno orijentisani način je mogućnost upotrebe biblioteka. Objektno orijentisano okruženje treba da obezbedi dobre biblioteke i mehanizme za pravljenje novih.

Osnovne biblioteke

Osnovne strukture podataka iz računarstva, kao što su skupovi, liste, stabla, stekovi i odgovarajući algoritmi, za sortiranje, pretraživanje i sl., se u razvoju softvera stalno koriste. Ako se koriste konvencionalni pristupi ove strukture i algoritmi se uvek iznova implementiraju. Ovo nije samo nepotrebno trošenje energije, nego i šteti kvalitetu softvera. Malo je verovatno da će pojedinac dostići optimalno rešenje u pogledu pouzdanosti i efikasnosti, ako neku strukturu pdoataka implementira ne zbog nje same , već kao deo šireg rešenja.

Okruženje za objektno orijentisani razvoj softvera treba da obezbedi gotove klase, koje su posvećene ovim opštim zadacima razvoja softvera. Za strukture i algoritme koji se najviše koriste treba da na raspolaganju postoje gotove klase.

Grafika i korisnički interfejsi

Mnogi moderni softverski sistemi su interaktivni i sa korisnicima komuniciraju preko grafičkih interfejsa. Ovo je jedno od područja u kome se objektno orijentisani softver najbolje dokazao. Oni koji razvijaju softver treba da pri ruci imaju grafičke biblioteke kako bi mogli da interaktivne aplikacije prave brzo i efikasno.

Za razvoj aplikacija koje svojim korisnicima pružaju grafički korisnčki interfejs treba da na raspolaganju budu gotove klase.

Mehanizmi za razvoj i indeksiranje biblioteka

Razvoj biblioteka je dugačak i težak posao. Nije moguće garantovati da će dizajn biblioteke prvi put biti perfektan. Zbog toga se javlja važan problem. Kako onima koji prave biblioteke omogućiti ažuriranje i modifikaciju, bez bojazni da će to izazvati haos u postojećim sistemima, koji zavise od tih biblioteka. Ovo je pitanje razvoja biblioteke, ali i pitanje jezika koji se koristi za pisanje tih biblioteka.

Još jedan problem koji se javlja kod biblioteka je potreba za mehanizmom kojim se identifikuju klase koje rešavaju određene zadatke. Ovaj problem zadire u tri različite oblasti. Sa jedne strane to je pitanje kojim se bavi jezik (pošto on mora da obezbedi način da se indeksne informacije navedu unutar klase), samu biblioteku i alate (koji moraju da postoje da bi se postavio upit koji pronalazi klasu koja zadovoljava neke uslove).

To znači da klase iz biblioteka moraju da sadrže indeksne informacije po kojima se mogu pronaći.

Biblioteke klasa u Javi

U Javi postoji standardna biblioteka klasa koja se može koristiti kada je potrebno.

Biblioteka klsa je skup klasa koje podržavaju rzvoj programa. Kompajler ili razvojno okruženje često dolaze sa pripremljenim bibliotekama klasa. Biblioteke klasa se mogu dobiti i od nezavisnih proizvođača. Klase koje se nalaze u ovim bibliotekama su za programere bitne, potšo sadrže specijalne funkcije. Programeri često postaju zavisni od ovih biblioteka i počinju da o njima razmišljaju kao delu jezika. Tehnički govoreći, međutim, ove biblioteke nisu deo jezika.

Klasa String, na primer, koju smo do sada više puta koristili nije sastavni deo jezika Java. Ona je deo standardne biblioteke klasa u Javi. Klase koje se nalaze u ovoj biblioteci klasa su napravljene u firmi Sun Microsystems, koja je i kreirala jezik Java.

Biblioteka klasa se sastoji iz više delova. Svaki deo čine uzajamno povezane klase. Ovi delovi se ponekad nazivaju Java API (application programming interface). Na primer, postoji Java API za pristup bazama podataka. Tu su klase koje se koriste za rad sa bazama. Postoji i Java Swing API, gde su klase koje sadrže specijalne grafičke komponente, koje se koriste kod kreiranja korisničkog interfejsa. Ponekad se cela standardna biblioteka klasa naziva Java API-jem.

Klasa u okviru standardne biblioteke klasa su grupisane u pakete (package). Svaka klasa je deo nekog paketa. Klasa String, je na primer, deo paketa java.lang. Klasa System je deo istog paketa.

Paketi

Sve klase u Javi su organizovane u pakete. Paketi u Javi su slični direktorijumima (fasciklama) koji se koriste za organizaciju podataka na disku. I klase koje smo do sada koristili se nalaze u paketima, ali do sada na to nismo obraćali pažnju.

Pakovanje klase se izvodi tako što se u datoteku u kojoj se nalazi klasa na početku doda iskaz za pakovanje. Ovaj iskaz mora biti prvi u toj datoteci.

```
package Proba;
public class Sfera …
```

Prethodni iskaz kaže da je klasa Sfera pripadnik paketa Proba. U jednom paketu može biti više klasa. Ključna reč public u definiciji klase ukazuje na to da se klasi Sfera može pristupiti i iz drugih paketa i klasa u programu. Ako se prilikom definicije klase ne navede ključna reč public, onda se toj klasi može pristupati samo iz metoda klasa koje se nalaze u istom paketu.

Paketi su povezani sa strukturom direktorijuma na kome se nalaze datoteke sa Java klasama. Već znamo da se definicija klase nalazi u datoteci čije je ime isto kao ime klase. Sve datoteke u paketu ImePaketa moraju se nalaziti u direktorijumu ImePaketa.

Paketi mogu da imaju strukturu. Na primer, ako želite da napravite jedan vrhovni paket za rad sa geomtrijom i nazovete ga Geometrija, a da u njemu razdvojite klase koje se koriste za 2D i 3D geometriju, možete da napravite još dva paketa po imenu 2dOblici i 3dOblici, koji se nalaze u paketu Geometrija. Ti paketi se definišu na sledeći način:

```
package Geometrija.2dOblici;
public class Linija …
```

Klasa Linija iz ovog primera mora da se nalazi u direktorijumu 2dOblici, koji se nalazi u direktorijumu Geometrija. Ova struktura može imati dubinu koja odgovara Vašim potrebama.

Prilikom kompajliranja datoteka sa izvornim kodom, nastaju datoteke sa bajt kodom, čija je ekstenzija .class. Struktura direktorijuma sa ovim datotekama, takođe mora da poštuje pakete koji se primenjuju. O ovome brine kompajler, ali i vi to morate znati, jer se prilikom prenošenja tih klasa ta struktura mora da ispoštuje.

Dodavanje klasa iz nekog drugog paketa u program

Ako su klase u drugom paketu definisane sa ključnom reči public, onda u novi program te klase možete dodati preko iskaza import. Ovaj iskaz treba da se nalazi na početku datoteke sa definicijom klase, ali iza iskaza za definisanje paketa. Na primer, ako u datoteci želite da učinite dostupnim klase iz paketa Geometrija.3dOblici, možete da napišete:

```
import Geometrija.3dOblici.*;
```

Zvezdica na kraju prethodnog iskaza znače da se uključuju sve klase iz paketa. Ako se napiše konkretna klasa, pripadnik tog paketa, onda se može pristupati samo toj klasi.

```
import Geometrija.3dOblici.Sfera; // uključuje klasu Sfera
```

Ime klase, pored imena same klase, sadrži i ime paketa u kome se ta klasa nalazi. To znači da u programu možete da imate dve klase sa istim osnovnim imenom, ako se one nalaze u različitim paketima. Ovo je isto kao sa datotekama na disku, koje mogu da imaju isto ime, ako se nalaze u različitim direktorijumima.

Standardne klase i paketi

Klase koje se nalaze u paketu java.lang se automatski mogu koristiti kod pisanja programa. Ove klase su toliko osnovne da se mogu smatrati proširenjem jezika. Svaka klasa iz paketa java.lang, kao što su klase String ili System, se može koristiti u programu bez ikakvih dodatnih radnji.

Pored paketa java.lang u standardnoj javinoj biblioteci klasa postoje i drugi paketi. Ako želite da koristite neki od ovih paketa, morate ih u program uključiti preko deklaracije import.

Neki od najvažnijih paketa u standardnoj biblioteci klasa su:

Paket	Opis
java.applet	Kreiraju se programi (apleti) koji se lako prebacuju preko Weba
java.awt	Crta se grafika i kreira grafički korisnički interfejs (AWT je skraćenica za Abstract Windowing toolkit)
java.beans	Definišu se softverske komponente koje se lako ubacuju u aplikacije.
java.io	Klase koje obavljaju različite funkcije vezane za ulaz i izlaz
java.lang	Generalna podrška. Ovaj paket je automatski uvezen u sve programe.
java.math	Klase za matematička izračunavanja
java.net	Klase za komunikaciju preko mreže
java.rmi	Klase za pisanje programa koji se mogu distribuisati na više računara (RMI je skraćenica za Remote Method Invocation).
java.security	Klase vezane za sigurnost
java.sql	Klase za rad sa bazama podataka. SQL je skraćenica za Structure Query Language)
java.text	Klase za formatiranje teksta
java.util	Generalne pomoćne klase
javax.swing	Klase za kreiranje grafičkog korisničkog interfejsa. Proširenje klasa iz paketa AWT

Pretraživači klasa i generisanje informacija potrebnih za pretraživanje

Već smo pomenuli da je kod kreiranja biblioteka klasa vrlo bitno da postoje mehanizmi za pretraživanje tih klasa, kao i mehanizmi za postavljanje informacija u klase, tako da se one kasnije mogu pretraživati.

U Javi postoji i jedno i drugo. Prilikom pisanja izvornog koda za klasu, moguće je u tu klasu postaviti informacije, koje će kasnije primenom posebnog alata biti izdvojene i prikazane kao dokumentacija klase.

Javadoc je poseban program koji se koristi za kreiranje dokumentacije u HTML formatu, na osnovu izvornog Java koda. Ovaj program ispituje izvorni kod i vadi specijalno označene informacije, a onda pravi web strane koje te informacije prikazuju.

Javadoc čita specijalne komentare i oznake koje su ubačene u izvorni kod. On dolazi kao deo Java SDK-a, koji preuzimate sa Sunovog sajta. Izvršni program javadoc.exe se nalazi u direktorijumu bin, instalacije jave. Program se poziva slično kao što se poziva Java kompajler.

javadoc mojaKlasa.java

Poziva se obavezno sa izvornom datotekom kao argumentom. Ona mora imati ekstenziju .java.

Komentari za dokumentaciju

Komentari koji se koriste za dokumentaciju se dele na opis i oznake. Opis treba da da pregled funkcije koda. Oznake se koriste za definisanje funkcija kao što su verzija koda, ili povratni tipovi iz metoda.

Javadoc obrađuje komentare koji su postavljeni između otvorene oznake /** i zatvorene oznake */. Komentari se ogu širiti u više linija, ali svaka linija mora počinjati karakterom *. Javadoc će ove karaktere kasnije odbaciti. U komentarima se mogu nalaziti i HTML oznake. Evo kako izgleda jedan javadoc komentar:

```
/**
* Ovo je javadoc komentar.
*/
```

Komentare treba postavljati pažljivo. Alat će automatski kopirati prvu rečenicu svakog ovakvog komentara na vrh HTML dokumenta. Opis koji sledi treba da bude sažet i kompletan. Program prepoznaje samo komentare koji su postavljeni neposredno ispred klase, konstruktora, metoda, interfejsa ili deklaracije polja.

Oznake

Oznake se pišu u okviru komentara za dokumentaciju. Svaka oznaka mora da se nalazi u posebnom redu i mora joj prethoditi karakter *. Kod upotrebe oznake mora se voditi računa o malim i velikim slovima. Oznake uvek počinju simbolom @.

Različite oznake se koriste u različitim situacijama. Na primer, oznaka @param mora da se da za svaki argument i koristi se za opis argumenata metoda. Oznaka @return se mora dati za svaki metod i opisuje povratnu vrednost iz metoda.

Neke od oznaka su:

- @author - ubacuje stavku Author sa definisanim tekstom.
- @deprecated - Ubacuje reč Deprecated (zastarelo) ispisanu masnim slovima, a iza nje zadati tekst.
- @param - Ubacuje deo Params sa listom argumenata metoda i njihovim opisom.
- @return - Ubacuje deo Returns sa listom povratnih vrednosti i njihovim opisom.
- @version - Dodaje zaglavlje Version, ako se koristi opcija programa sa komandne linije - version.

……..

Datoteke koje nastaju

HTML datoteke koje nastaju posle upotrebe programa javadoc imaju jasno definisanu strukturu.

U prvom delu dokumenta se nalazi generalni opis klase. Tu je ime klase, iza kojeg sledi grafički prikazano stablo nasleđivanja. Zatim je prikazan tekst sa opisom klase.

Iza generalnog opisa sledi lista konstruktora i metoda. Prikazana je signatura svih konstruktora i metoda, sa po jednom rečenicom opisa. Ime metoda ili konstrktora je hiperlink na detaljniji opis koji se nalazi u trećem delu dokumenta.

U trećem delu se nalazi kompletan opis svih metoda. Prvo se prikazuje signatura, a zatim sledi opis tog metoda, koji je dat u komentaru. Tu su prikazane i liste argumenata i povratna vrednost.

Ako se program javadoc primeni na paket, pored informacija o svakoj pojedinačnoj klasi će se prikazati i stablo sa svim klasama koje učestvuju u tom paketu.

Na sledećoj slici je prikazana HTML datoteka nastala upotrebom programa javadoc. Obratite pažnju na listu svih klasa i paketa sa leve strane, kao i opis klase u glavnom prozoru.

Neke korisne klase iz standardne Javine biblioteke

Klasa Random

Prilikom pisanja programa se često javlja potreba za slučajnim brojevima. Često se slučajni brojevi koriste kod igara, na primer kod prikaza sledeće karte. Simulator leta može da koristi slučajne brojeve da bi odredio koliko često treba da simulira kvar motora. Program koji pravi testove može slučajne brojeve da upotrebi za određivanje sledećeg pitanja.

Klasa Random, koja se nalazi u paketu java.util, predstavlja generator pseudo slučajnih brojeva. Generator slučajnih brojeva bira jedan slučajan broj iz zadatog opsega. Koristi se termin pseudo slučajan broj, pošto ovo nije zaista slučajan broj. Generator pseudo slučajnih brojeva obavlja niz složenih izračunavanja, koja se zasnivaju na nekoj inicijalnoj vrednosti i tako daje traženi broj. Iako ovi brojevi nisu zaista slučajni (pošto se izračunavaju) vrednosti su ipak dovoljno slučajne za većinu slučajeva.

Neki od metoda klase Random su:

float nextFloat()	Vraća slučajan broj između 0.0 (uključujući i taj broj) i 1.0 (bez tog broja)
int nextInt()	Vraća slučajan broj koji može biti bilo koji ceo broj (pozitivan ili negativan)
int nextInt(int num)	Vraća slučajan broj koji se nalazi u intervalu od 0 do num-1

U narednom primeru je pokazano kako se koristi ova klasa za dobijanje slučajnih brojeva iz raznih opsega.

```
import java.util.Random;
public class SlucajniBrojevi{
   public static void main (String[] args) {
      Random generator = new Random();
      int num1;
      float num2;

      num1 = generator.nextInt();
      System.out.println ("Slučajan ceo broj: " + num1);

      num1 = generator.nextInt(10);
      System.out.println ("Od 0 do 9: " + num1);

      num1 = generator.nextInt(10) + 1;
      System.out.println ("Od 1 do 10: " + num1);

      num1 = generator.nextInt(15) + 20;
      System.out.println ("Od 20 do 34: " + num1);

      num1 = generator.nextInt(20) - 10;
      System.out.println ("Od -10 do 9: " + num1);

      num2 = generator.nextFloat();
      System.out.println ("Slučajan float (izmedju 0-1): " + num2);

      num2 = generator.nextFloat() * 6;  // 0.0 to 5.999999
      num1 = (int)num2 + 1;
      System.out.println ("Od 1 do 6: " + num1);
   }
}
```

Izlaz iz ovog programa može biti:

```
Slucajan ceo broj: 435003399
Od 0 do 9: 1
Od 1 do 10: 8
Od 20 do 34: 22
```

```
Od -10 do 9: -7
Slucajan float (izmedju 0-1): 0.69418126
Od 1 do 6: 6
```

U programu smo za dobijanje vrednosti u nekom intervalu koristili dodavanje i oduzimanje određenog broja. Na primer, da bismo dobili slučajan broj u opsegu od 1 do 6, možemo pozvati nextInt(6), koji vraća vrednost od 0 do 5, pa onda rezultatu dodati 1.

Klasa Math

Klasa Math se koristi za matematičke operacije koje su potrebne kod izračunavanja. Ova klasa se nalazi u paketu java.lang. Neki od korisnih metoda ove klase su dati u sledećoj tabeli:

static int abs(int num)	Vraća apsolutnu vrednost od num
static double acos(double num)	Vraća arkus kosinus od num
static double asin(double num)	Vraća arkus sinus od num
static double atan(double num)	Vraća arkus tangens od num
static double cos(double ugao)	Vraća kosinus ugla
static double sin(double ugao)	Vraća sinus ugla
static double tan(double ugao)	Vraća tangens ugla
static double ceil(double num)	Vraća najmanji ceo broj veći ili jednak sa num
static double exp(double stepen)	Vraća vrednost e podignutu na zadati stepen
static double floor(double num)	Vraća se najveći ceo broj manji ili jednak sa num
static double pow(double num, double stepen)	Vraća se num podignut na stepen
static double sqrt(double num)	Vraća se kvadratni koren od num. Num mora biti pozitivna vrednost.

Svi metodi ove klase su statički, što znači da se mogu pozvati preko imena klase, odnosno ne mora se praviti objekat klase pre poziva.

Sledeći primer pokazuje kako se koriste neki metodi ove klase. Program pronalazi rešenja kvadratne jednačine ($ax^2 + bx + c$), pod pretpostavkom da su oba rešenja realna.

```
import java.util.Scanner;
public class Quadratic{
   public static void main (String[] args) {
      int a, b, c;  // ax^2 + bx + c
      double diskriminanta, koren1, koren2;

      Scanner scan = new Scanner (System.in);

      System.out.print ("Unesite koeficijent uz x na kvadrat: ");
      a = scan.nextInt();

      System.out.print ("Unesite koeficijent uz x: ");
      b = scan.nextInt();
```

```
        System.out.print ("Unesite konstantu: ");
        c = scan.nextInt();

        // Računaju se koreni
        // Pretpostavlja se pozitivna diskriminanta

        diskriminanta = Math.pow(b, 2) - (4 * a * c);
        koren1 = ((-1 * b) + Math.sqrt(diskriminanta)) / (2 * a);
        koren2= ((-1 * b) - Math.sqrt(diskriminanta)) / (2 * a);

        System.out.println ("Koren #1: " + koren1);
        System.out.println ("Koren #2: " + koren2);
    }
}
```

Klasa Scanner koju smo ovde koristili za unos podataka sa tastature je nova klasa koja postoji samo u verziji Java 1.5. Kao što se vidi iz koda nalazi se u paketu java.util.

Koreni kvadratne jednačine se računaju po formuli $x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$.

Klase za formatiranje brojeva

Za formatiranje brojeva se koriste klase NumberFormat i DecimalFormat. Ove klase se nalaze u paketu java.text.

Klasa NumberFormat se koristi za generalno formatiranje brojeva. Instanca ove klase se obično ne pravi preko operatora new. Umesto toga se instanca dobija pozivom nekog od statičkih metoda ove klase. To su na primer, metode getCurrencyInstance() ili getPercentInstance().

Ove metode vraćaju objekte koji se dalje koriste za formatiranje brojeva. Metod getCurrencyInstance omogućava formatiranje novčanih vrednosti, a getPercentInstance omogućava formatiranje procentualnih vrednosti.

Posle poziva ovih metoda, dobija se objekat klase NumberFormat, a formatiranje se vrši pozivom njegovih metoda format.

Sintaksa pomenutih metoda je data u sledećoj tabeli:

String format(double broj)	Vraća se string koji sadrži zadati broj, formatiran prema postavljenom obrascu.
static NumberFormat getCurrencyInstance()	Vraća se instanca objekta NumberFormat, koja predstavlja trenutni format za prikazivanje novčanih vrednosti.
static NumberFormat getPercentInstance()	Vraća se instanca objekta NumberFormat koja predstavlja trenutni format za prikazivanje procentualnmih vrednosti.

Sledeći primer pokazuje kako se koriste metodi klase NumberFormat.

```
import java.util.Scanner;
import java.text.NumberFormat;

public class Purchase{
   //-----------------------------------------------------------------
```

```
    //  Izračunava se cena kupljenih stavki na osnovu vrednosti
    //  koje korisnik unese
    //-----------------------------------------------------------------
    public static void main (String[] args)   {
       final double PORESKA_STOPA = 0.18;  // porez 18%

       int kolicina;
       double medjuzbir, porez, ukupnaCena, jedinicnaCena;

       Scanner scan = new Scanner (System.in);

       NumberFormat fmt1 = NumberFormat.getCurrencyInstance();
       NumberFormat fmt2 = NumberFormat.getPercentInstance();

       System.out.print ("Unesite kolicinu: ");
       quantity = scan.nextInt();

       System.out.print ("Unesite jedinicnu cenu: ");
       jedinicnaCena = scan.nextDouble();

       medjuzbir = kolicina * jedinicnaCena;
       porez = medjuzbir * PORESKA_STOPA;
       ukupnaCena = medjuzbir + porez;

       // Štampanje izlaza sa odgovarajućim formatima
System.out.println ("Medjuzbir: " + fmt1.format(medjuzbir));
       System.out.println ("Porez: " + fmt1.format(porez) + " od "
                           + fmt2.format(PORESKA_STOPA));
       System.out.println ("Ukupno: " + fmt1.format(ukupnaCena));
    }
}
```

Izlaz iz ovog programa može biti:

```
Unesite kolicinu:20
Unesite jedinicnu cenu: 3.5
Medju zbir: $70.00
Porez: $4.20 od 6%
Ukupno: $74.20
```

Na računaru na kojem je dobijen ovakav izlaz je podešen SAD novčani sistem.

Za razliku od klase NumberFormat, klasa DecimalFormat se instancira na tradicionalan način. preko operatora new. Konstruktor ove klase prima string koji predstavlja obrazac koji se koristi za formatiranje. Nakon toga se metod format primenjuje na konkretnu vrednost. Ako želimo da promenimo postavljeni obrazac, možemo upotrebiti metod applyPattern. Signatura ovih metoda je prikazana u sledećoj tabeli:

DecimalFormat (String pattern)	Konstruktor. Kreira se novi objekat sa zadatim obrascem.
void applyPattern(String pattern)	Primenjuje se zadati obrazac na objekat DecimalFormat
String format(double number)	Vraća se string koji sadrži zadati broj, formatiran prema tekućem obrascu.

Za definisanje obrasca se koriste različiti karakteri, koji imaju specijalna značenja. Na primer, 0 znači bilo koji broj, koji se obavezno prikazuje, # predstavlja bilo koji broj, ali se 0 ne prikazuje i sl. Detalji o ovim obrascima se mogu naći u dokumentaciji klase DecimalFormat. Na primer obrazac "0.###" ukazuje na to da se sa leve strane decimalne tačke treba da odštampa najmanje jedna cifra. Ako je celobrojni deo broja 0 onda treba da se odštampa ta nula. Decimalni deo se zaokružuje na tri cifre.

U narednom primeru se računa obim i površina kruga, za koji je korisnik uneo poluprečnik. Nevažeće nule u rezultatu, na primer 78.540, se ne štampaju.

```
import java.util.Scanner;
import java.text.DecimalFormat;

public class Krug{
   //-----------------------------------------------------------------
   //  racuna se povrsina i obim kruga za zadati poluprecnik
   //-----------------------------------------------------------------
   public static void main (String[] args){
      int radius;
      double povrsina, obim;

      Scanner scan = new Scanner (System.in);

      System.out.print ("Unesite poluprecnik kruga: ");
      radius = scan.nextInt();

      povrsina = Math.PI * Math.pow(radius, 2);
      obim = 2 * Math.PI * radius;

      // Zaokruzuje na tri decimalna mesta
      DecimalFormat fmt = new DecimalFormat ("0.###");

      System.out.println ("Povrsina kruga: " + fmt.format(povrsina));
      System.out.println ("Obim kruga: " + fmt.format(obim));
   }
}
```

Izlaz iz programa je:

```
Unesite poluprecnik kruga: 5
Povrsina kruga: 78.54
Obim kruga: 31.416
```

Klase omotači za primitivne tipove

Već smo pomenuli da se u Javi za predstavljanje podataka koriste primitivni tipovi i objekti. U nekim slučajevima može biti problem to što postoje dve kategorije podataka. Na primer, mogli bismo napraviti objekat koji služi kao kontejner za različite tipove drugih objekata. U određenim situacijama bismo mogli poželeti da u takav kontejner stavimo celobrojnu vrednost. U tom slučaju teba da ubacimo primitivnu vrednost u njen omotač.

Klasa omotač predstavlja konkretan primitivni tip. Na primer, klasa Integer predstavlja celobrojnu vrednost. Objekat koji se napravi na osnovu ove klase sadrži jedan ceo broj. Konstruktori klasa omotača kao argument, prihvataju primitivnu vrednost koju treba da uskladište. Na primer:

```
Integer godine + new Integer(40);
```

Nakon ove deklaracije objekat godine predstavlja broj 40, ali kao objekat. On se može koristiti svuda gde je u programu potreban objekat, a ne primitivni tip.

Za svaki primitivni tip postoji odgovarajuća Java klasa. Sve ove klase se nalaze u paketu java.lang. U narednoj tabeli su prikazane sve klase omotači:

Primitivni tip	Klasa omotač
byte	Byte
short	Short
int	Integer
long	Long
float	Float
double	Double
char	Character
boolean	Boolean
void	Void

U ovim klasama se nalaze različiti metodi koji se koriste za rukovanje primitivnim tipovima. Tu su metodi za konverziju iz jednog u drugi primitivni tip, ili za konverziju stringa u odgovarajući primitivni tip i sl. Neki od metoda klase Integer su dati u sledećoj tabeli.

double doubleValue()	Vraća vrednost celog broja iz instance, ali pretvorenu u tip double
static int parseInt(String str)	Vraća celobrojnu vrednost koja odgovara zadatom stringu
Integer (int value)	Pravi se novi objekat Integer, na osnovu zadatog celog broja.

Treba imati na umu da u ovim klasama ne postoje metodi koji omogućavaju promenu vrednosti koja je jednom zadata. To su nepromenljivi objekti. Kada na primer, jednom napravite novu instancu klase Integer, sa vrednošću 40, kao u prethodnom primeru, više je ne možete promeniti.