Operatorji in izrazi

Programiranje 2, Tomaž Dobravec

Aritmetični operatorji

- unarni operator (en sam operand): – in +

```
int a=-1;
int b=+15;     // isto kot int b=15;
```

- binarni operator (dva operanda) +, -, *, /, %

```
int a=8, b=2, c;

c = (a + b) * 3;   // c = 30;
c = a / b;         // 4 (celostevilsko deljenje)
c = 33 % 5;        // 3 (ostanek po deljenju s 5)

float x = 3, y = 2, z;
z = x / y;         // z = 1.5 (realno deljenje)
```

Aritmetični operatorji - posebnosti

- Java ne pozna operatorjev `//` in `**`

Znak `//` se v Javi uporablja za začetek komentarja

- Operator `/` v javi upošteva kontekst:
  - če sta oba operanda celi števili, java izvede celoštevilsko deljenje
  - sicer: izvede se deljenje v realni aritmetiki

Primer:

```
System.out.println(10/3);
```
... izpiše se `3`

```
System.out.println(10.0/3);
```
... izpiše se `3.33333`

- Za računanje potence, uporabimo metodo `pow()` razreda `Math`

```
Math.pow(2, 10) == 2^10
```

Relacijski in logični operatorji

Relacijski operatorji

`<`	manjše
`<=`	manjše ali enako
`>`	večje
`>=`	večje ali enako
`==`	enako
`!=`	različno

Logični operatorji

`!`	negacija
`&&`	konjunkcija (logični in)
`||`	disjunkcija (logični ali)

Python namesto
`!` `&&` `||`
uporablja
`not` `and` `or`

Relacijski in logični operatorji

- Logični izrazi se preverjajo od leve proti desni. Izračun vrednosti logičnih izrazov se konča takoj, ko je znana končna vrednost.

```
PRIMER:  if (a==2 && zmanjsajStevec() > 0) {…}
```

- Pravilo ARL: Aritmetični operatorji imajo prednost pred relacijskimi, relacijski pa pred logičnimi

`i < a - 1`	**isto kot**	`i < (a-1)`
`a < 5 && b > 3`	**isto kot**	`(a<5) && (b>3)`

Operatorja ++ in --

Python ne pozna operatorjev ++ in --

```
                ... isto kot ...
i++;                                i = i + 1;
i--;                                i = i - 1;

x = n++;                            x = n; n = n + 1;
y = i--;                            y = i; i = i - 1;

t[i++]=a;                           t[i] = a;  i = i + 1;
```

```
                ... isto kot ...
++i;                                i = i + 1;
--i;                                i = i - 1;

x = ++n;                            n = n + 1; x = n;
y = --i;                            i = i - 1; y = i;

t[++i]=a;                           i = i + 1; t[i] = a;
```

Operatorji +=, -=, *=, /= in %=

```
            ... isto kot ...
i+=5;                          i = i + 5;
i-=2;                          i = i - 2;
i/=2;                          i = i / 2;
i*=2;                          i = i * 2;
i%=2;                          i = i % 2;
```

Tridelni pogojni operator ?

- Operator `pogoj ? izraz1 : izraz2` najprej ovrednoti `pogoj`; če je ta resničen (`true`), operator vrne vrednost `izraz1` sicer `izraz2`

- Namesto

```
if (pogoj)
  rezultat = izraz1;
 else
  rezultat = izraz2;
```

Tridelni operator v Pythonu:

```
izraz1 if pogoj else izraz2
```

lahko pišemo tudi

```
rezultat = pogoj ? izraz1 : izraz2;
```

Primer uporabe:

```
System.out.printf("%s", x < 37 ? "OK" : "VROCINA!");
```

Bitni operatorji

Java pozna naslednje bitne operatorje:

&	in	AND
|	ali	OR
^	ekskluzivni ali	XOR
<<	predznačen pomik bitov v levo	LSHIFT
>>	predznačen pomik bitov v desno	RSHIFT
>>>	nepredznačen pomik bitov v desno	
~	eniški komplement	NOT

Bitni operatorji

- Primer delovanja bitnih operatorjev:

11&5=1	11|5=15	11^5=14	~11=20
01011	01011	01011	~01011
&00101	|00101	^00101	
=00001	=01111	=01110	=10100

to velja, če je velikost podatkovnega tipa 5 bitov;

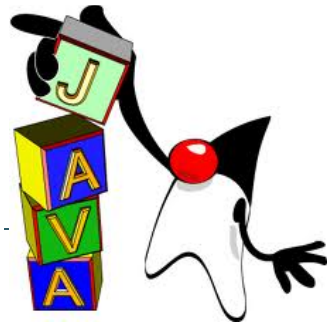
velikost pod. tipa 8 bitov → ~11=11110100=244

- Množenje in deljenje z 2 s pomočjo bitnih operatorjev:

```
a = b << 1;     // isto kot a=b*2
a = b << 3;     // isto kot a=b*8
a = b >> 1;     // isto kot a=b/2
a = b >> 10;    // isto kot a=b/1024
```

Naloga

Bitni operatorji



`operatorji/Biti.java`

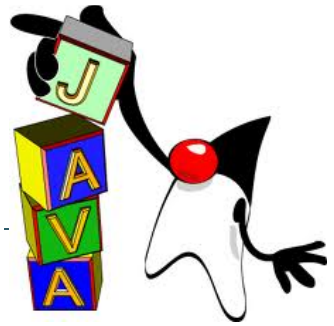
- Napiši program, ki izpiše število prižganih bitov v prebranem številu.

Primer izvajanja programa:

```
$ javac Biti.java
$ java Biti 3243424
Stevilo prizganih bitov v 3243424 je 11
```

Naloga

Dvojiški zapis



operatorji/Dvojisko.java

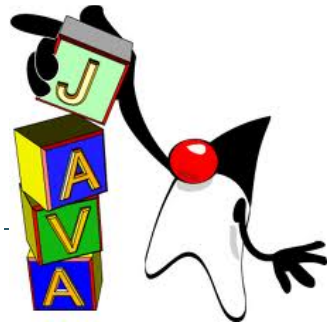
Napiši metodi

```
static String vDvojisko(int x)
static int vDesetisko(String d)
```

za pretvorbo med desetiškim in dvojiškim številskim sistemom. Metodi naj pri pretvarjanju uporabita bitne operatorje in operator `?`.

Naloga

Množica znakov



operatorji/Mnozica.java

Celoštevilski tip `int` lahko uporabimo za implementacijo množice z `31` elementi takole:

element `i` je v množici `x` ⇔ v `x` prižgan `i`-ti bit

Naloga: napiši vse metode, ki jih uporablja spodnji program, da boš dobil pravilen izpis, kot je prikazano na desni strani spodaj.

```
int mnozica1 = getPraznaMnožica();
mnozica1 = dodajElement(mnozica1, 'a','i', 'e', 'z');
System.out.printf("Mnozica1=%s\n", toString(mnozica1));

int mnozica2 = getPraznaMnožica();
mnozica2 = dodajElement(mnozica2, 'a', 'f','z', 'b');
System.out.printf("Mnozica2=%s\n", toString(mnozica2));

int presek = presek(mnozica1, mnozica2);
int unija  = unija (mnozica1, mnozica2);
System.out.printf("presek(%s, %s)=%s\n",
  toString(mnozica1), toString(mnozica2), toString(presek));
System.out.printf("inija (%s, %s)=%s\n",
  toString(mnozica1), toString(mnozica2), toString (unija));
```

```
Mnozica1=[a,e,i,z]
Mnozica2=[a,b,f,z]
presek([a,e,i,z], [a,b,f,z])=[a,z]
inija ([a,e,i,z], [a,b,f,z])=[a,b,e,f,i,z]
```

Dodaj tudi metode `brisiElement()`, `preklopiElement()` in `jeElement()`.

Prireditveni stavek

- **Prireditveni stavek vrne vrednost.**
- Rezultat prirejanja `b = a` je `a`.

Uporaba:

- `c = b = a;`
- ```
  while ((c=fis.read()) != -1) {
      …
  }
  ```
  ```

Prioriteta in asociativnost operatorjev

Vprašanje 1: Se bo v izrazu

```
d = a + b * c;
```

najprej izračunal seštevek ali zmnožek?

Odgovor: najprej zmnožek, saj ima * večjo prioriteto kot +.

Prioriteta in asociativnost operatorjev

Vprašanje 2: Se bo v izrazu

```
d =  8 / 4 / 2;
```

najprej delilo z 2 ali s 4?

Odgovor: najprej s 4, saj je / levo-asociativen.

Prioriteta in asociativnost operatorjev

Operatorji	Asociativnost
(), [], .	L
!, ~, ++, --, +, -, (type), new	D
*, /, %	L
+, -	L
<<, >>, >>>	L
<, <=, >=, >, instanceof	L
==, !=	L
&	L
^	L
|	L
&&	L
||	L
?:	D
=, + =, − =, * =, / =, %=, &=, ^=, | =, <<=, >>=, >>>=	D

Tabela: Prioriteta in asociativnost operatorjev v *javi*