Prosti tipovi podataka

Tipovi realnih brojeva

- Uzimaju vrednosti iz skupa racionalnih brojeva
  - `float` – jednostruka preciznost (4 bajta)
  - `double` – dvostruka preciznost (8 bajtova)
- Zapis realnih literala:
  - Standardni zapis – decimalni broj, tačka, decimalni broj
    Primer: `583.45`
  - Naučni zapis – mantisa + eksponent
    Primer: `5.8345E2` ($5.8345 \cdot 10^2$)
  - Na kraju može da stoji oznaka tipa (`D, d, F, f`)
    Primeri: `583.45D  5.8345E2F`
    (ako se oznaka ne navede podrazumeva se tip `double`)

Tipovi realnih brojeva

- **Operatori** (kao i kod celobrojnih tipova) – relacioni operatori, aritmetički operatori, operatori nad bitovima, operatori dodele, uslovni operator, "*cast*" operatori, operator konkatenacije stringova

- Tri **posebne konstante** (nastaju samo kao rezultat izvršavanja nekih matematički nedefinisanih operacija) :

  `-Infinity` (minus beskonačno)

  `Infinity` (beskonačno)

  `NaN` (*'Not a number'* – nije broj)

- Klase `Float` i `Double` – korisne metode i način predstavljanja realnih tipova preko klasa

Tipovi realnih brojeva: primeri

Ispravni realni izrazi	Neispravni realni izrazi
`2e13f + 7 * 9.8`	`++3.14 // mora biti promenljiva`
`x++ //x je float ili double`	

Rezultati nekih matematički nedefinisanih operacija

```
System.out.println(-1.0 / 0.0 );   // štampa: -Infinity
System.out.println( 0.0 / 0.0 );   // štampa: NaN
System.out.println( 1.0 / 0.0 );   // štampa: Infinity
```

Promenljive

- **Promenljiva** predstavlja jedan podatak u memoriji računara, kojem se pristupa preko imena (identifikatora)
- Svaka promenljiva ima:
  - Ime (po kojem se promenljiva prepoznaje)
  - Vrednost (čuva se u memoriji)
  - Tip podataka (određuje vrednosti koje promenljiva može imati)
- **Imena** se grade od slova, cifara i znakova `_` i `$` , po pravilima zapisivanja identifikatora sa prošlog predavanja
- **Tip podataka** može biti bilo koji prosti tip, ili referencijalni tip (klasa, niz, nabrojivi tip – o njima više kasnije)

Deklaracija i inicijalizacija

- U Javi, svaka promenljiva se pre korišćenja mora deklarisati, na primer:

```
int br;          // Ime: br, tip: int
double x;        // Ime: x, tip: double
String s;        // Ime: s, tip: String
boolean nasao; // Ime: nasao, tip: boolean
```

- Moguće je promenljivama dodeliti početnu vrednost prilikom deklaracije:

```
int br = 42;                  double x = 5.5;
String s = "Neki tekst";    boolean nasao = true;
```

- A moguće je i deklarisati/inicijalizovati više promenljivih odjednom:

```
int br, n, i, j;
double x = 1.0, y = 2.0, z = 3.0;
```

- Ako se pri deklaraciji ne inicijalizuje, promenljiva prostog tipa imaće podrazumevanu (*default*) vrednost tog tipa koja odgovara broju 0, odnosno `(imeTipa)0`: `0, 0L, 0.0, '\0', false...`

  (sem ako je u pitanju lokalna promenljiva, o njima više kasnije)

Deklaracija i inicijalizacija: preporuke

- Dobra je praksa inicijalizovati promenljive prilikom deklaracije (ukoliko je moguće i ima smisla)
- Imena promenljivih treba birati tako da opisuju sadržaj promenljive, odnosno ono što promenljiva predstavlja u rešenju problema
  - Primer: promenljivu koja sadrži konačnu ocenu studenta na ispitu je preporučljivo imenovati `ocena`, ili još bolje `konacnaOcena`, a ne recimo `o`, `x` ili `ejfvbewjhrb`
- Preporučuje se da se svaka promenljiva deklariše u posebnom redu
  - Primer: preferira se

```
double ispit = 0.0;
double test = 10.0;
double konacnaOcena = 0.0;
```

u odnosu na

```
double ispit = 0.0, test = 10.0, konacnaOcena = 0.0;
```

Štampanje

- Vrednost promenljive (odnosno izraza) može se štampati pozivom standardnih metoda

  `System.out.print()`

  `System.out.println()`
- Kao argument u zagradama prosleđuje se vrednost koja se štampa, a koja može biti bilo kog tipa podataka
- Metode štampaju odgovarajuću reprezentaciju date vrednosti kao `String`
- Razlika između metoda je što `System.out.println()` dodaje znak za kraj reda (`'\n'`) na kraj ispisa

Štampanje: primer

```
class Stampa {
  public static void main(String[] arg) {
    int broj = 42;
    char znak = 'A';
    double ocena = 10.0;
    System.out.println(broj);
    System.out.print("znak = ");
    System.out.println(znak);
    // + kao konkatenacija stringova
    System.out.println("ocena = " + ocena);
  }
}
```

- Izlaz:

```
42
znak = A
ocena = 10.0
```

Operatori

- **Relacioni operatori (operatori poređenja)**
  - `==` (jednakost)
  - `!=` (nejednakost)
  - `<` (manje)
  - `<=` (manje ili jednako)
  - `>` (veće)
  - `>=` (veće ili jednako)

Primeri

```
x == 4
slovo != 'd'
45 < duzina
x >= y
```

Relacioni operatori: primer (1/3)

```
class RelacioniOperatori {
  public static void main(String[] args) {
    int i = 37;
    int j = 42;
    int k = 42;
    System.out.println("Vrednosti promenljivih:");
    System.out.println("    i = " + i);
    System.out.println("    j = " + j);
    System.out.println("    k = " + k);
    // Vece
    System.out.println("Vece:");
    System.out.println("    i > j = " + (i > j)); // false
    System.out.println("    j > i = " + (j > i)); // true
    System.out.println("    k > j = " + (k > j)); // false
    // Vece ili jednako
    System.out.println("Vece ili jednako:");
    System.out.println("    i >= j = " + (i >= j)); // false
    System.out.println("    j >= i = " + (j >= i)); // true
    System.out.println("    k >= j = " + (k >= j)); // true
```

Relacioni operatori: primer (2/3)

```
        // Manje
        System.out.println("Manje:");
        System.out.println("    i < j = " + (i < j)); // true
        System.out.println("    j < i = " + (j < i)); // false
        System.out.println("    k < j = " + (k < j)); // false
        // Manje ili jednako
        System.out.println("Manje ili jednako:");
        System.out.println("    i <= j = " + (i <= j)); // true
        System.out.println("    j <= i = " + (j <= i)); // false
        System.out.println("    k <= j = " + (k <= j)); // true
        // Jednako
        System.out.println("Jednako:");
        System.out.println("    i == j = " + (i == j)); // false
        System.out.println("    k == j = " + (k == j)); // true
        // Nejednako
        System.out.println("Nejednako:");
        System.out.println("    i != j = " + (i != j)); // true
        System.out.println("    k != j = " + (k != j)); // false
    }
}
```

Relacioni operatori: primer (3/3)

Izlaz:

```
Vrednosti promenljivih:
    i = 37
    j = 42
    k = 42
Vece:
    i > j = false
    j > i = true
    k > j = false
Vece ili jednako:
    i >= j = false
    j >= i = true
    k >= j = true
```

```
Manje:
    i < j = true
    j < i = false
    k < j = false
Manje ili jednako:
    i <= j = true
    j <= i = false
    k <= j = true
Jednako:
    i == j = false
    k == j = true
Nejednako:
    i != j = true
    k != j = false
```

Operatori

- **Aritmetički operatori**
  - `+` (unarni plus)
  - `-` (unarni minus)
  - `*` (množenje)
  - `/` (deljenje)
  - `%` (ostatak pri deljenju)
  - `+` (sabiranje)
  - `-` (oduzimanje)

Primeri

```
-a + 2 * (b % 3)
5 / b / 3 - 1.0

int a = 10;
System.out.println(++a); // 11
System.out.println(a);   // 11
int b = 10;
System.out.println(b++); // 10
System.out.println(b);   // 11
```

  - `++` (prefiksni ili postfiksni operator uvećanja)
  - `--` (prefiksni ili postfiksni operator smanjenja)

Aritmetički operatori: primer (1/3)

```
class AritmetickiOperatori {
  public static void main(String[] args) {
    int i = 37;
    int j = 42;
    double x = 27.475;
    double y = 7.22;
    System.out.println("Vrednosti promenljivih:");
    System.out.println("    i = " + i);
    System.out.println("    j = " + j);
    System.out.println("    x = " + x);
    System.out.println("    y = " + y);
    // Sabiranje
    System.out.println("Sabiranje:");
    System.out.println("    i + j = " + (i + j));
    System.out.println("    x + y = " + (x + y));
    // Oduzimanje
    System.out.println("Oduzimanje:");
    System.out.println("    i - j = " + (i - j));
    System.out.println("    x - y = " + (x - y));
```

Aritmetički operatori: primer (2/3)

```
        // Mnozenje
        System.out.println("Mnozenje:");
        System.out.println("    i * j = " + (i * j));
        System.out.println("    x * y = " + (x * y));
        // Deljenje
        System.out.println("Deljenje:");
        System.out.println("    i / j = " + (i / j));
        System.out.println("    x / y = " + (x / y));
        // Ostatak pri deljenju
        System.out.println("Ostatak pri deljenju:");
        System.out.println("    i % j = " + (i % j));
        System.out.println("    x % y = " + (x % y));
        // Mesanje tipova
        System.out.println("Mesanje tipova:");
        System.out.println("    j + y = " + (j + y));
        System.out.println("    i * x = " + (i * x));
    }
}
```

Aritmetički operatori: primer (3/3)

Izlaz:

```
Vrednosti promenljivih:
    i = 37
    j = 42
    x = 27.475
    y = 7.22
Sabiranje:
    i + j = 79
    x + y = 34.695
Oduzimanje:
    i - j = -5
    x - y = 20.255000000000003
```

```
Mnozenje:
    i * j = 1554
    x * y = 198.36950000000002
Deljenje:
    i / j = 0
    x / y = 3.805401662049862
Ostatak pri deljenju:
    i % j = 37
    x % y = 5.815000000000002
Mesanje tipova:
    j + y = 49.22
    i * x = 1016.575
```

Aritmetički operatori ++ i −−

- Unarni operatori:
  - Uvećanja (inkrementacija): ++
  - Umanjenja (dekrementacija): −−

- Koriste se uz promenljivu nekog brojevnog tipa i uvećavaju, odnosno umanjuju vrednost promenljive za 1

- Primer:

```
++broj
```

radi isto što i

```
broj = broj + 1
```

Aritmetički operatori ++ i --

- Operatori ++ i -- mogu da se koriste u prefiksnom i postfiksnom obliku
  - Primeri: prefiksni oblik `++broj` `--broj`
    postfiksni oblik `broj++` `broj--`
- Efekat na vrednost promenljive je isti
- Razlika je u vrednosti celog izraza (koji uključuje ime promenljive i operator)
  - Postfiksni operator: vrednost izraza je originalna vrednost promenljive
  - Prefiksni operator: vrednost izraza je uvećana/umanjena vrednost promenljive

Aritmetički operatori ++ i −−: primer

```
class IncDecOperatori {
  public static void main(String[] args) {
    int i = 10;
    int j = 3;
    int k;
    k = j++ + i;
    System.out.println("k = " + k);
    k = ++j + i;
    System.out.println("k = " + k);
  }
}
```

- Izlaz:

```
k = 13
k = 15
```

Aritmetički operatori ++ i −−: preporuka

- Treba paziti da izrazi u kojima se koriste operatori ++ i −− budu kratki i jasni

```
class IncDecOperatori2 {
  public static void main(String[] args) {
    int i = 10;
    int j = 3;
    int k = +j--+--i-++j-i--+i;
    System.out.println("k = " + k);
  }
}
```

- Izlaz:

```
k = 8
```

Operatori

- **Operatori nad bitovima**
  - `~` negacija (invertovanje) bitova
  - `<<` pomeranje bitova ulevo
  - `>>` pomeranje bitova udesno vodeći računa o predznaku broja
  - `>>>` pomeranje bitova udesno ne vodeći računa o predznaku broja

Primeri

```
System.out.println(~0);          // štampa -1
System.out.println(~7);          // štampa -8
System.out.println(1 << 3);      // štampa 8
System.out.println(-1 << 3);     // štampa -8
System.out.println(17 >> 3);     // štampa 2
System.out.println(-17 >> 3);    // štampa -3
System.out.println(17 >>> 3);    // štampa 2
System.out.println(-17 >>> 3);   // štampa 536870909
System.out.println(1 << 35);     // štampa 8
```

Operatori

- **Logički operatori**

!	logička negacija		
&&	logička konjunkcija	**&**	konjunkcija nad bitovima
||	logička disjunkcija	**|**	disjunkcija nad bitovima
		^	ekskluzivna disjunkcija nad bitovima

- Operator `!` je unaran, dok su svi ostali binarni
- Operatori s leve strane primenljivi su isključivo na operandima tipa `boolean`
- Operatori s desne strane su istovremeno i logički operatori (primenljivi na tip `boolean`), i operatori koji rade na bitovima (primenljivi na brojevnim tipovima)

Logički operator !

- Neka je `a` operand (promenljiva ili izraz) tipa `boolean`
- Istinitosna tablica za operator `!`:

`!a`

a	Rezultat
true	false
false	true

- Operator `!` daje suprotan rezultat u odnosu na operand

Logički operator ! : primer

```
class TestNOT {
  public static void main(String[] args) {
    boolean a = true;
    boolean b = false;
    System.out.println(!a);
    System.out.println(!b);
  }
}
```

- Izlaz:

```
false
true
```

Logički operatori

- Neka su `a` i `b` operandi (promenljive ili izrazi) tipa `boolean`
- Istinitosne tablice za operatore `&&` i `&`, odnosno `||` i `|`

`a && b, a & b`

a	b	Rezultat
true	true	true
true	false	false
false	true	false
false	false	false

`a || b, a | b`

a	b	Rezultat
true	true	true
true	false	true
false	true	true
false	false	false

Logički operatori

- Za tip `boolean` operatori `&&` i `&`, odnosno `||` i `|` daju iste rezultate

- Postoje važne **razlike**:
  - `&&` i `||` se primenjuju samo na tip `boolean`
  - `&` i `|` se mogu primeniti na svim brojevnim tipovima, i tada rade nad pojedinačnim bitovima, na primer:

    ```
    System.out.println(7 & 9); // 1
    System.out.println(7 | 9); // 15
    ```

  - `&&` i `||` se evaluiraju **lenjim izračunavanjem** (*short-circuiting*), što znači da ako se vrednost izraza može zaključiti na osnovu prvog operanda, drugi operand se neće ni izračunavati; kod `&` i `|` se uvek izračunavaju svi operandi (**vredno izračunavanje**)

Logički operatori && i &: primer

```
class TestAND {
  public static void main(String[] args) {
    int i = 0;
    int j = 10;
    boolean test;
    test = (i > 10) && (j++ > 9);
    System.out.println(i);
    System.out.println(j);
    System.out.println(test);
    test = (i > 10) & (j++ > 9);
    System.out.println(i);
    System.out.println(j);
    System.out.println(test);
  }
}
```

- Izlaz:

```
0
10
false
0
11
false
```

Logički operatori | | i |: primer

```
class TestOR {
  public static void main(String[] args) {
    int i = 0;
    int j = 10;
    boolean test;
    test = (i < 10) || (j++ > 9);
    System.out.println(i);
    System.out.println(j);
    System.out.println(test);
    test = (i < 10) | (j++ > 9);
    System.out.println(i);
    System.out.println(j);
    System.out.println(test);
  }
}
```

- Izlaz:

```
0
10
true
0
11
true
```

Logički operator ^

- Istinitosna tablica za operator ^

a ^ b

a	b	Rezultat
true	true	false
true	false	true
false	true	true
false	false	false

- Operator ekskluzivne disjunkcije ^ daje rezultat `true` ako je tačno jedan operand `true`, a drugi `false`
- Pošto je neophodno izračunati vrednosti oba operanda da bi se došlo do rezultata, lenjo izračunavanje nema smisla (i zato ne postoji operator ^^)

Logički operator ^: primer

```
class TestXOR {
  public static void main(String[] args) {
    boolean a = true;
    boolean b = true;
    System.out.println(a ^ b);
    a = true; b = false;
    System.out.println(a ^ b);
    a = false; b = true;
    System.out.println(a ^ b);
    a = false; b = false;
    System.out.println(a ^ b);
  }
}
```

- Izlaz:

```
false
true
true
false
```

Operatori

- **Operatori dodele**

`=`	dodela
`+=`	operator dodele sa prethodnom primenom operatora `+`
`-=`	operator dodele sa prethodnom primenom operatora `-`
`*=`	operator dodele sa prethodnom primenom operatora `*`
`/=`	operator dodele sa prethodnom primenom operatora `/`
`%=`	operator dodele sa prethodnom primenom operatora `%`
`<<=`	operator dodele sa prethodnom primenom operatora `<<`
`>>=`	operator dodele sa prethodnom primenom operatora `>>`
`>>>=`	operator dodele sa prethodnom primenom operatora `>>>`
`&=`	operator dodele sa prethodnom primenom operatora `&`
`|=`	operator dodele sa prethodnom primenom operatora `|`
`^=`	operator dodele sa prethodnom primenom operatora `^`

Operatori dodele

- Osnovni operator = koristi se u obliku:
  `promenljiva = vrednost`
- Promenljiva sa leve strane znaka = mora biti već deklarisana (ili se operator koristi prilikom deklaracije)
- Vrednost sa desne strane može biti neki **literal** (npr. `2`), **promenljiva**, **poziv metoda**, odnosno u opštem slučaju **izraz** koji kombinuje literale, promenljive i pozive metoda
- Način izvršavanja: prvo se izračuna vrednost izraza sa desne strane =, zatim se ta vrednost dodeli promenljivoj sa leve strane
- Takođe, cela konstrukcija `promenljiva = vrednost` predstavlja izraz koji ima vrednost jednaku dodeljenoj vrednosti

  Primer:
  ```
  int i;
  int j = (i = 22) + 8;
  ```
  - Ovu mogućnost u principu treba izbegavati, jer može dovesti do grešaka koje se lako prave a teško uočavaju

Operatori dodele

- Operatori oblika `op=` , gde je `op` neki od navedenih operatora (+, –, ...) koriste se u obliku:
  `promenljiva op= vrednost`

- Izvršavaju se isto kao
  `promenljiva = promenljiva op vrednost`

- Primer:

```
int i = 2;
i += 2; // i = i + 2;
i *= 3; // i = i * 3;
i %= 5; // i = i % 5;
```

Operatori

- **Specijalni operatori**

`?:`	uslovni operator
`(`***`imeTipa`***`)`	eksplicitna konverzija tipa
`+`	konkatenacija stringova

Primeri

```
int i;
i = (int)3.14; // konverzija iz tipa double u tip int
System.out.println(i); // štampa 3

System.out.println("Novi " + "Sad"); // štampa: Novi Sad
System.out.println("Broj " + i);     // štampa: Broj 3
// Bitno je da je jedan operand tipa String, drugi će biti
// automatski konvertovan u String
```

Uslovni operator ? :

- Jedini **ternarni operator** u Javi
  - Potrebno mu je proslediti tri operanda
- Koristi se u obliku:

  `izraz1 ? izraz2 : izraz3`

  gde je `izraz1` tipa `boolean`, a ostali izrazi mogu biti bilo kog tipa (ne obavezno istog)
- Izvršava se na sledeći način:
  - Izračuna se vrednost `izraz1`
  - Ako je ta vrednost `true`, vrednost celog izraza dobija se izračunavanjem vrednosti `izraz2`
  - U protivnom, vrednost celog izraza dobija se izračunavanjem vrednosti `izraz3`
- Izračunavanje `izraz2` i `izraz3` se radi po potrebi, tj. “lenjo”

Uslovni operator ? : – primer

```
class UslovniOperator {
  public static void main(String[] args) {
    String status;
    int bodovi = 80;
    status = (bodovi >= 55) ? "Polozio" : "Nije polozio";
    System.out.println(status);
    int i = 0;
    int j = 22;
    System.out.println((i < 10) ? "Manji od 10" : j++);
    System.out.println("j = " + j);
  }
}
```

- Izlaz:

```
Polozio
Manji od 10
j = 22
```

Operatori

- **Ostali operatori**

  `instanceof` – pripadnost referencijalnom tipu

  `.` (tačka) – pristup članu klase, paketa...

  `[]` (uglaste zagrade) – pristup elementu niza

  `new` – kreiranje instance klase

- Većinu ovih operatora detaljnije ćemo obraditi kasnije

Operatori: prioritet

- Svi operatori razvrstani su po prioritetu, tako da je za svaki dobro formiran izraz tačno poznato kojim se redosledom izračunavaju vrednosti operanada
- Prioritet operatora se menja korišćenjem zagrada (i)
- Primer: vrednost izraza

```
6 % 2 * 5 + 4 / 2 + 88 - 10
```

izračunava se kao da su zagrade stavljene na sledeći način:

```
((((6 % 2) * 5) + (4 / 2)) + 88) - 10
```

Redosled izračunavanja može se potpuno promeniti:

```
6 % ((((2 * 5) + 4) / (2 + 88) - 10))
```

Operatori: prioritet

Operator	Komentar
`. [] new pozivMetoda()`	Operatori najvećeg prioriteta
`-- ++`	Postfiksni operatori
`(imeTipa) ~ ! -- ++ + -`	Unarni operatori. Prefiksni operatori
`* / %`	Množenje, deljenje, ostatak
`+ -`	Sabiranje, konkatenacija i oduzimanje
`<< >> >>>`	Pomeranje bitova
`< > <= >= instanceof`	Relacioni operatori
`== !=`	Ispitivanje jednakosti
`&`	Konjunkcija
`^`	Ekskluzivna disjunkcija
`|`	Disjunkcija
`&&`	Logička konjunkcija
`||`	Logička disjunkcija
`?:`	Uslovni operator
`= *= /= %= += -= <<= >>= >>>= &= ^= |=`	Operatori dodele imaju najmanji prioritet

Konverzija prostih tipova

- Često je u programu (tj. izrazima) potrebno "mešati" vrednosti različitih tipova podataka
  - Ovo je rađeno i u mnogim primerima koje smo videli, kao što je recimo izraz `2e13f + 7 * 9.8`
- Ista vrednost (npr. broj 7, posmatran kao broj a ne Java literal) se različito predstavlja u memoriji (na nivou bitova) u različitim brojevnim tipovima podataka
  - Kao `int`: `00000000000000000000000000000111`
  - Kao `float`: `01000000111000000000000000000000`
- S obzirom da vrednost izraza može biti samo jednog tipa, neophodno je da se vrednosti različitih tipova podataka **konvertuju**, tj. prebace iz jedne reprezentacije u drugu

Konverzija prostih tipova

- Razlikujemo dve vrste konverzije prostih tipova:
  - Proširujuće proste konverzije
  - Sužavajuće proste konverzije
- **Proširujuće proste konverzije** su:
  - iz tipa **`byte`** u tipove **`short`**, **`int`**, **`long`**, **`float`** ili **`double`**,
  - iz tipa **`short`** u tipove **`int`**, **`long`**, **`float`** ili **`double`**,
  - iz tipa **`char`** u tipove **`int`**, **`long`**, **`float`** ili **`double`**,
  - iz tipa **`int`** u tipove **`long`**, **`float`** ili **`double`**,
  - iz tipa **`long`** u tipove **`float`** ili **`double`**,
  - iz tipa **`float`** u tip **`double`**.
- Ne dolazi do gubitka informacija, jer važi sledeće:
  - tipovi su međusobno kompatibilni,
  - ciljni tip je veći od izvornog tipa

Konverzija prostih tipova

- **Sužavajuće proste konverzije** su:
  - iz tipa **`byte`** u tip **`char`**
  - iz tipa **`short`** u tipove **`byte`** ili **`char`**
  - iz tipa **`char`** u tipove **`byte`** ili **`short`**
  - iz tipa **`int`** u tipove **`byte`**, **`short`** ili **`char`**
  - iz tipa **`long`** u tipove **`byte`**, **`short`**, **`char`** ili **`int`**
  - iz tipa **`float`** u tipove **`byte`**, **`short`**, **`char`**, **`int`** ili **`long`**
  - iz tipa **`double`** u tipove **`byte`**, **`short`**, **`char`**, **`int`**, **`long`** ili **`float`**
- Može doći do gubitka informacija, pri konverziji se odseca decimalni deo (konverzija realni-celi) i viši bajtovi broja (konverzije celi-celi)
- Ove konverzije programer mora eksplicitno naznačiti korišćenjem cast operatora
  - Primer: `int n = (int)53.7;`

    (bez cast operatora kompajler bi prijavio grešku)

Konverzija prostih tipova

- Proširujuću prostu konverziju po pravilu Java kompajler može da reguliše automatski, bez intervencije programera, ali treba biti svestan njenog postojanja i pravila
  - Pogrešna očekivanja mogu dovesti do grešaka u kodu
  - Nepotrebna konverzija može usporiti program

- Primer: `2e13f + 7 * 9.8`
  - Literal `7` je tipa `int`, a literal `9.8` tipa `double`, pa je podizraz `7 * 9.8` tipa `double` i vrši se konverzija iz `int` u `double`
  - Literal `2e13f` je tipa `float`, i konvertuje se u `double` da bi tip celog izraza bio tipa `double`
  - Da je izraz bio zapisan `2e13 + 7.0 * 9.8` ne bi bilo konverzije

Konverzija prostih tipova

Pravila za automatsko unapređenje (promociju) tipova koja se primenjuju u izrazima:

- **Pravilo unarne numeričke promocije** (primena unarnih operatora)**:** vrednost tipa `byte`, `short` ili `char` se menja proširujućom prostom konverzijom u tu istu vrednost tipa `int`
- **Pravilo binarne numeričke promocije** (primena binarnih operatora)**:**
  - Ako je jedan operand tipa `double` onda se i drugi konvertuje u tip `double`
  - Inače, ako je jedan operand tipa `float` onda se i drugi konvertuje u tip `float`
  - Inače, ako je jedan operand tipa `long` onda se i drugi konvertuje u tip `long`
  - Inače, ako oba operanda već nisu tipa `int`, onda se konvertuju u tip `int`

Konverzija prostih tipova

Primer:

```
short s = 42;
s = -s;
s = s + 1;
```

- Šta nije u redu sa datim kodom?

- Zbog automatske promocije tipova, mora se upotrebiti cast operator:

```
s = (short)-s;
s = (short)(s + 1);
```

zato što su izrazi `-s` i `s + 1` promovisani u tip `int`

Primer Java programa

(Idealne) etape pri rešavanju nekog problema pisanjem programa:

- Potpuno **razumevanje i shvatanje** definisanog problema
- **Uočavanje objekata** i njihovih osobina i konstrukcija apstraktnog modela problema koji se rešava
- **Formiranje** apstraktnog modela i algoritma
- **Realizacija** apstraktnog modela u željenom programskom jeziku, “kodiranje” algoritma
- **Unošenje** programa u jedan ili više fajlova pomoću nekog editora
- **Prevođenje** programa
- **Izvršavanje i testiranje** programa

Primer Java programa

- **Problem**: Izračunavanje godišnjih faktora inflacije u periodu od 10 godina za godišnje stope inflacije od 7% i 8%.
- Globalni algoritam:
  - **inicijalizacija vrednosti**;
    ispis zaglavlja,
  - **ponavljanje** (sve dok se ne dostigne maksimalna godina):
    uzmi narednu godinu;
    izračunaj faktore;
    odštampaj izračunate vrednosti;

Izlaz:

```
Year:    7%        8%
   1     1.07      1.08
   2     1.144     1.166
   3     1.225     1.259
   ...
   10    1.967     2.158
```

Primer Java programa

```
class Inflation {
  public static void main(String[] args) {
    int year;
    double factor7 = 1.0;
    double factor8 = 1.0;
    System.out.println("Year:\t7%\t8%");
    for (year = 1; year <= 10; year++) { // year = year + 1
      factor7 = factor7 * 1.07;
      factor8 = factor8 * 1.08;
      System.out.print(year);
      System.out.print("\t");
      System.out.print((double)((int)(factor7 * 1000.0)) / 1000.0);
      System.out.print("\t");
      System.out.println((double)((int)(factor8 * 1000.0)) / 1000.0);
    }
  }
}
```