Razvoj softvera

dr.sc. Emir Mešković

I predavanje

Sadržaj predavanja

- Java osnove
  - Uvod
  - Java platforma
  - Tipovi podataka u Javi
  - Kontrola toka
  - Nizovi
  - Stringovi
  - Korisnički unos

Uvod

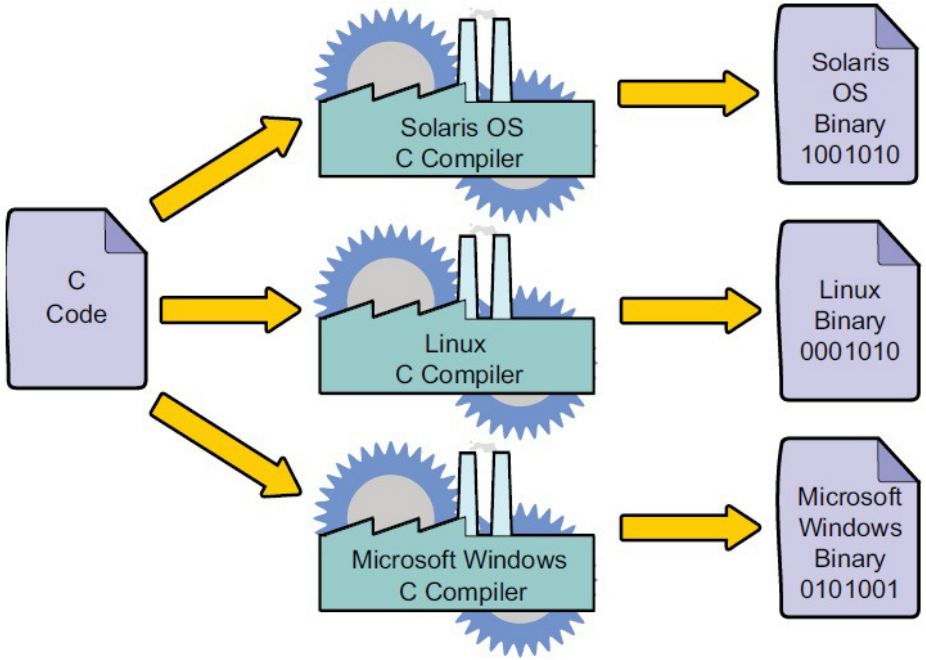
- Java istorijat
  - 1991 Sun Microsystems Projekat pod kodnim imenom 'Green' sa ciljem dizajna platforme za male korisničke uređaje.
    - U okviru projekta razvijen programski jezik Oak modeliran kao jednostavnija verzija C++.
    - Oak kompajler je generirao kod koji se izvršavao na hipotetičkom procesoru (tzv virtual machine) koji se implementirao u obliku interpretera.
    - Oak kasnije dobija ime Java
  - 1996 verzija 1.0 fokusirana kao razvojni jezik za Internet. Ograničene mogućnosti, slabe performanse

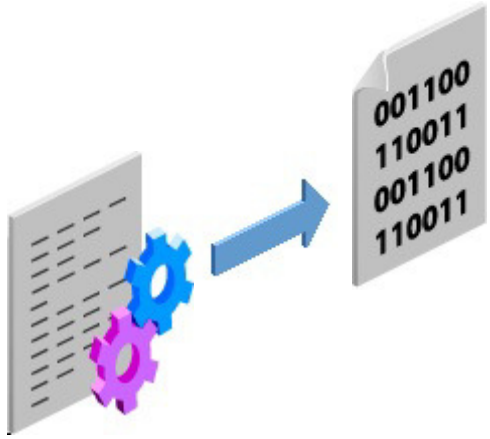
Uvod

- Java istorijat
  - 1998 verzija 1.2 Java za tri platforme
    - Micro Edition mali uređaji kao mobilni telefoni
    - Standard Edition standardni desktop
    - Enterprise Edition za servere
    - Revizija kompletne platforme uključujući sve bitnije biblioteke. Uveden Swing GUI.
  - 2000 – 2002 verzije 1.3 i 1.4 sa poboljšanim performansama i novim dodacima u standardnoj biblioteci
  - 2004 Java 5.0 (ili 1.5)
    - Generics, autoboxing
  - 2006 – 2015 verzije 6 i 7 poboljšanja o pogledu rukovanja izuzecima, diamond operator, switch sa stringovima
  - 2016 – Java 8 kao najvažnija proširenja uvodi lambda izraze i Stream API kao podršku za operacije nad streamovima elemenata

Računarski program

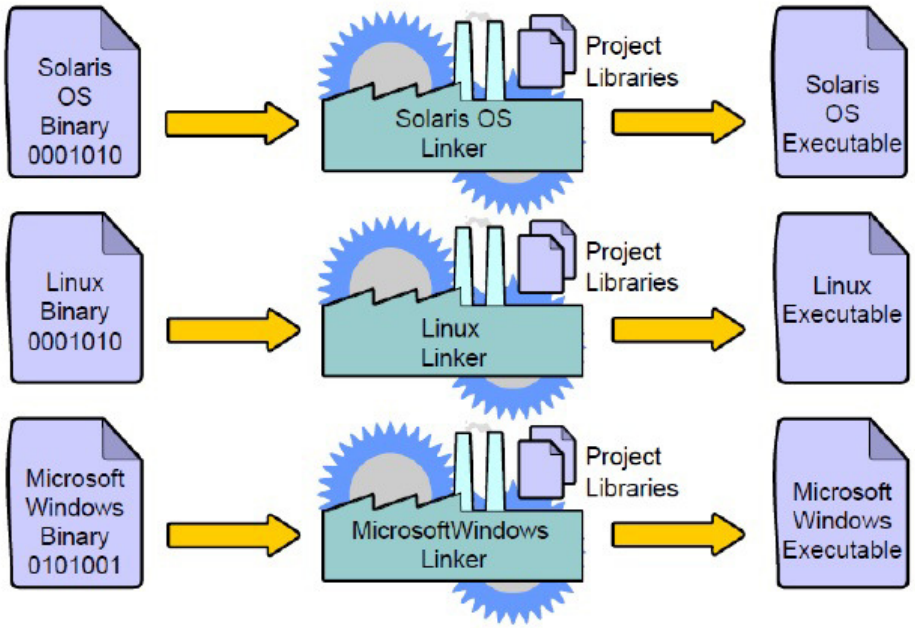
- Skup instrukcija koje se izvršavaju na računaru ili drugom digitalnom uređaju
  - Na mašinskom nivou program se sastoji od binarnih instrukcija (1-ca i 0-la) – mašinski kod
  - Većina programa je napisana u programskom jeziku visokog nivoa ("čitljivi kod") – mora biti preveden u mašinski kod



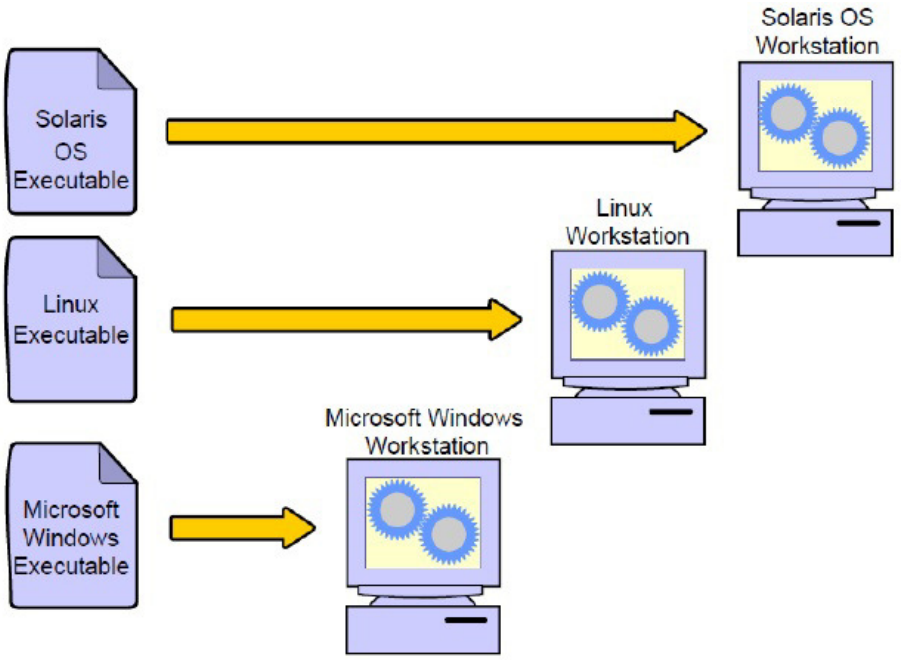


Prevođenje koda visokog nivoa u mašinski kod

Računarski program

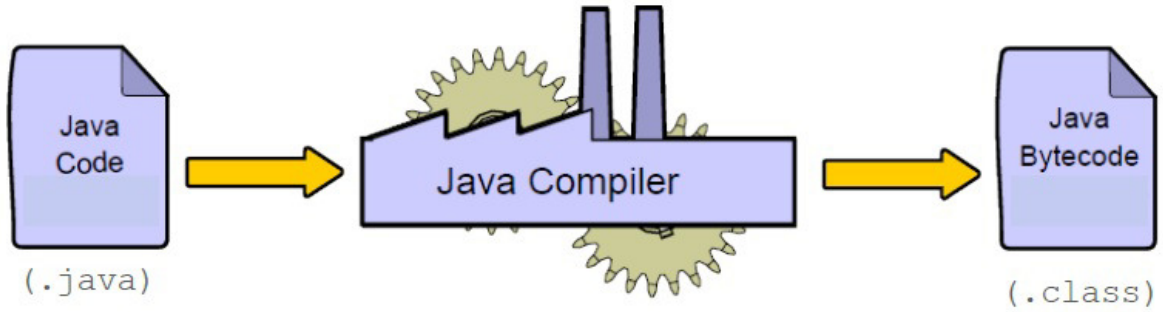


Povezivanje sa platformski specifičnim bibliotekama

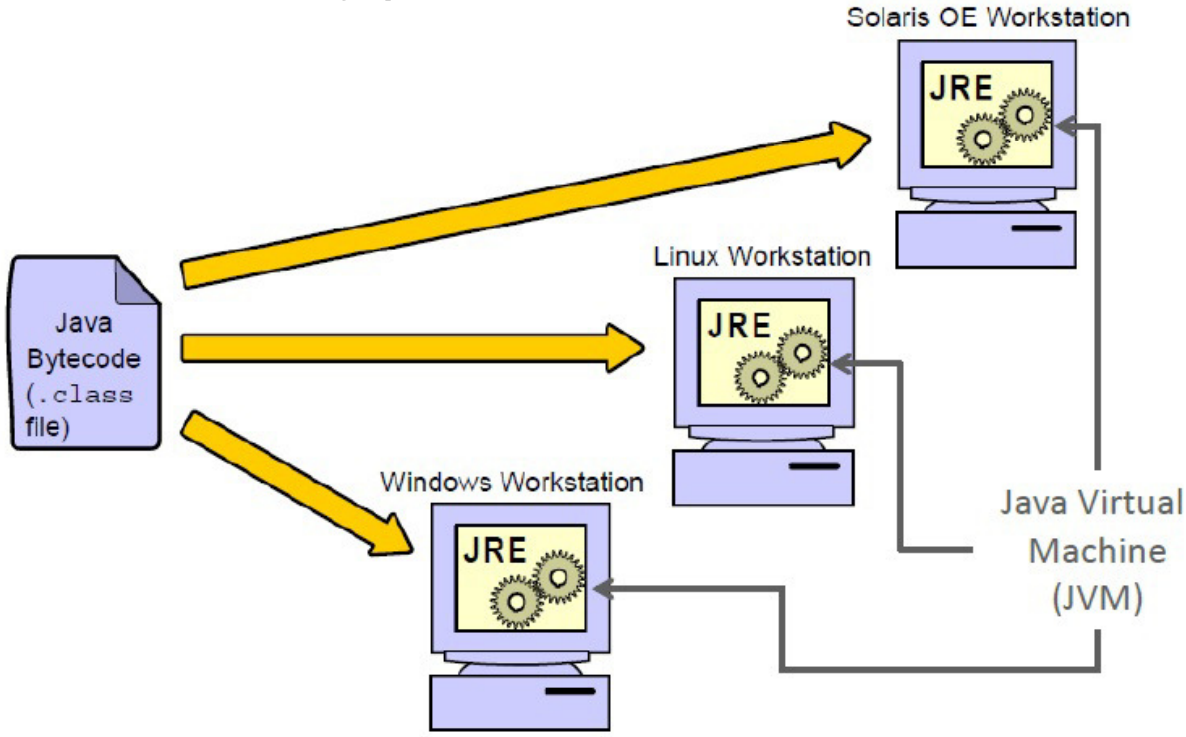


Platformski ovisni programi

Java program



Java je platformski neovisna



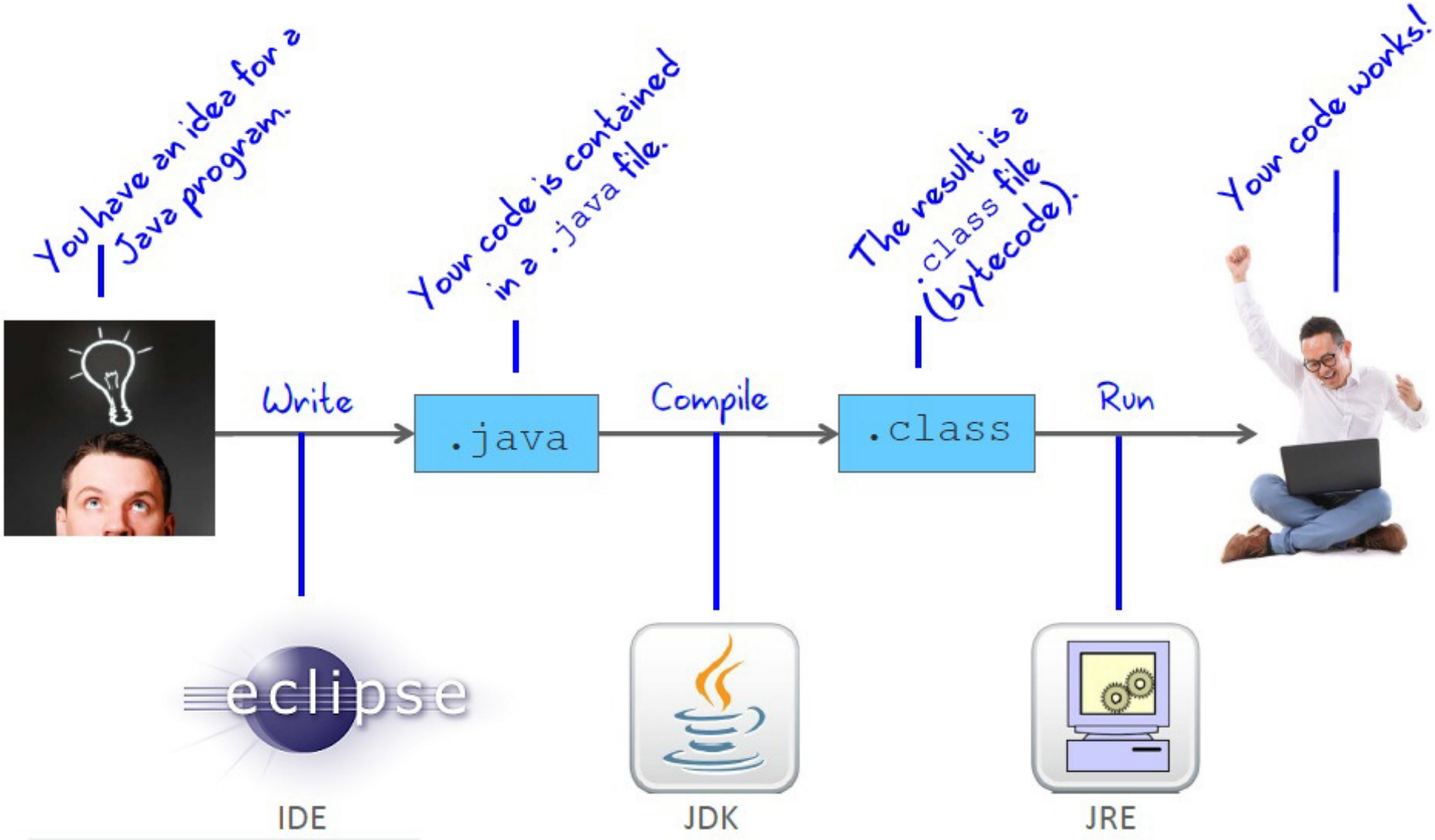
Java programi se izvršavaju u JVM

Java platforma

- Java je platforma koja se sastoji od
  - okruženja za izvršavanje aplikacija **JRE** (*Java Runtime Environment*) koje sadrži
    - **JVM** (*Java Virtual Machine*) program koji izvršava specijalizirani kod za abstraktni procesor(tzv *java bytecode*).
      - Prve verzije JVM-a samo su interpretirale bytecode a novije verzije prevode bytecode u mašinski kod prije izvršavanja (*Just In Time compilation* **JIT)**
    - Velike biblioteke koje se nanovo mogu koristiti u razvoju novih aplikacija (strukture, algoritmi, GUI, mreže itd)

  - Okruženje za razvoj aplikacija **JDK** (*Java Developers Kit*) koje sadrži
    - kompajler *javac koji prevodi programe* napisane u Java programskom jeziku u bytecode za JVM.
      - *javac* je napisan u programskom jeziku Java

Programiranje u Javi



Java programski jezik

- Java je OOP jezik baziran na C++ modelu
- Java ne podržava
  - Predprocesor (zaglavlja)
    - Deklaracija i implementacija java klasa vrši se unutar istog fajla
  - Višestruko naslijeđivanje
  - Manuelnu alokacija memorije (pointeri, destruktori)
    - JVM brine o dealokaciji dinamičke memorije preko garbage collectora (**GC**)
  - Preopterećenje operatora (*operator overloading*)
- Sličnost sa C++
  - Kod se implementira unutar blokova definiranim sa zagradama {}.
  - Identiče komande za kontrolu toka (*if, while, for, switch itd)*
  - Klase se definišu na sličan način

Prvi Java program


```
public class Test {
        public static void main(String[] args){
                System.out.println("Java, pozdrav!");
        }
}
```

- Kod je snimljen u fajlu `Test.java`
  - Program se može kompajlirati pomoću JDK-a tipkajući u komandnoj liniji u direktoriju u kojem je snimljen kod:
    - `javac Test.java`
  - ili putem IDE-a (npr Eclipse)
  - Nakon uspješnog kompaliranja proizvodi se bytecode fajl i to sa imenom `Test.class`
- Za izvršenje programa potrebno je startati program u JVM-u tipkajući u komandnoj liniji
  - `java Test`
  - Ili putem IDE-a

Obzervacije

- Kompletan program sadržan je unutar klase `Test`
  - Sve u Java-i mora biti definirano unutar klase uključujući i funkcije koje moraju biti metodi neke klase
- `main()` metod:
  - označava početak izvršavanja Java programa i u programu postoji samo jedan `main()` metod
  - mora biti definiran kao `public static void`
  - mora uzimati niz objekata tipa `String` (tj. `String[]`)
- Definicija klase u Java-i se ne terminira sa ; kao u C++
- Naredbe u Java-i se završavaju sa tačka-zarez (;)
- Prazna mjesta u kodu (*whitespaces*)
  - Prostor između riječi, prazne linije ili uvlake ispred linija koda
  - Ne utiču na izvršavanje, ali povećavaju čitljivost i održavanje koda

Obzervacije

- Za komunikaciju sa konzolom radi ispisa koristi se objekat `out` definiran unutar klase `System`
  - `System.out.println(obj)` ispisuje `obj` na izlazu pa prelazi u novu liniju
  - `System.out.print(obj)` ispisuje `obj` na izlazu
- Komentari
  - Jednolinijski komentar
    - Počinje sa dvije kose crte (//) i prostire se do kraja linije
  - Višelinijski komentar
    - Počinje sa kosa crta-zvjezdica (/*), a završava sa zvjezdica-kosa crta (*/)

Primitivni tipovi

- Cijelobrojni tipovi
  - `byte` – 8 bit (-128 do +127)
  - `short` – 16 bit (-32768 do +32767)
  - `int` – 32 bit (–2147483648 do 2147483647)
  - `long` – 64 bit (–9223372036854775808 do 9223372036854775807)
- Karakteri
  - `char` – 16 bit, unicode kodirani a ne ASCII
- Realni brojevi
  - `float` – 4 byte (-3.4 x 10^{38} do +3.4 x 10^{38})
  - `double` – 8 byte (-1.7 x 10^{308} do 1.7 x 10^{308})
- Logičke varijable
  - `boolean` (`true` ili `false`)

Primitivni tipovi

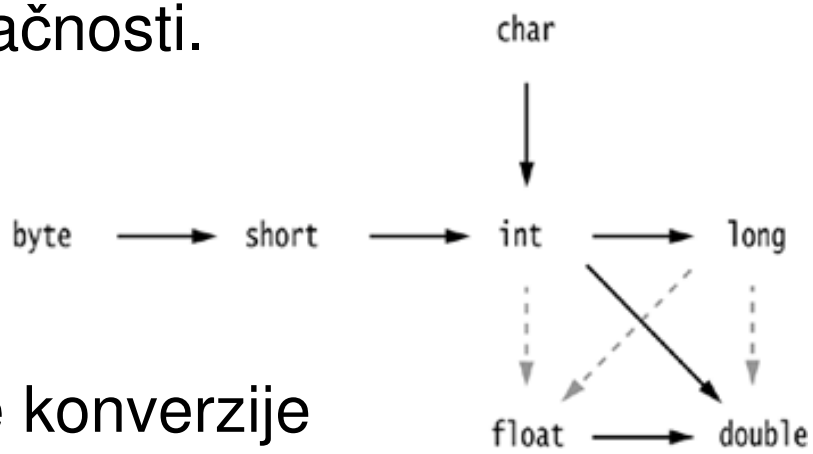
- Java je strogo tipizirani programski jezik
- Java varijable definiraju se isto kao C++ npr
  - `int a;`
  - `double m;`
  - `long f,p;`
- Nakon definiranja varijable ista se prije upotrebe mora inicijalizirati.
  - `int a;`
  - `System.out.println(a);` // nije ispravno
- Varijable se mogu inicijalizirati i prilikom definiranja
  - `int a = 10;`

Primitivni tipovi

- Varijable se mogu definirati i u hexadecimalnoj i binarnoj (od Java SE7) formi prefiksom `0x` i `0b` npr
  - `byte m = 0x10;  //16`
  - `byte n = 0b101;  //5`
- Za inicijalizaciju tipa `float` obavezno je potrebno dodati oznaku `f` nakon vrijednosti tj.
  - `float br = 3.56f;`
- Tip `char` se može inicijalizirati na načine
  - `char p = 'A';`
  - `char p = 65;`
  - `char p = '\u0041';`
- Ako se varijabla deklariše sa ključnom riječi `final` njena vrijednost je nepromjenjiva (*immutable*)
  - `final double PI = 3.141592;`
  - `PI = 3.14; // nije dozvoljeno`

Konverzija između tipova

- Konverzija između tipova, kao i u C++, može biti *implicitna i eksplicitna.*
- Sljedeći grafik pokazuje moguće validne *implicitne* konverzije između tipova gdje isprekidana linija naznačava mogući gubitak tačnosti.



- Primjer implicitne konverzije
  - `int n = 123456789; float f = n;`
- Ukoliko binarni operator operiše na dva različita tipa vrši se implicitna konverzija tipa koji sadrži manje informacije u drugi tip koji učestvuje u operaciji.

Konverzija između tipova

- Voditi računa kod implicitne konverzije:
  - `byte br = 128; //greška`
  - `byte br1 = 127;`
    `br1++; //OK -128`
  - `long velikiBroj = 1234567890; // greška`
  - `int br1 = 55555, br2 = 66666;`
    `long rez;`
    `rez = br1 * br2; // 3703629630 greška`
  - `int br1=53, br2=47;`
    `byte br3 = br1 + br2; //greška`
  - `int br1 = 7, br2 = 2;`
    `double rez = br1/br2; // netačno`
  - `float f = 23.5; //greška`

Konverzija između tipova

- Konverzije u kojima je moguć gubitak informacije u Java-i je moguće izvesti samo putem eksplicitne konverzije i to putem *cast* operatora.
- Varijabla se konvertira u željeni tip postavljanjem imena tog tipa u zagrade ispred varijable (slično kao u C-u) npr.
  - `double x = 4.56;`
  - `int p = (int) x;`
- Nije moguće izvršiti implicitnu, niti eksplicitnu konverziju između `boolean` i bilo kojeg numeričkog tipa. Ako je to iz nekog razloga potrebno uraditi onda se koristi izraz
  - `boolean b = true; int k = b ? 1 : 0;`

Operatori

- Aritmetički - operišu na numeričkim vrijednostima isto kao C++
  - `+, -, *, /, %`
  - `++, --`
  - `+=, -=, *=, /=, %=`
- Poređenje - porede dvije vrijednosti i vraćaju tip `boolean`
  - `<, >, >=, <=, ==, !=`
- Logički - rade striktno sa tipovima `boolean` i vraćaju `boolean`
  - `&&, ||, !`
- Operator `?:` je kondicioni operator sa identičnom upotrebom kao u C++
- Nije moguće preopteretiti operatore i Java ne podržava operatore `->`, *, & za rad sa memorijom

Kontrola toka

- Kontrola toka
  - `if, if-else, switch`
  - `for, while, do-while`
  - `break, continue`
- rade identično u Java-i
- Gornji izrazi operišu na blokovima koda koji su definirani pomoću zagrada {}
- Varijable definirane unutar blokova imaju lokalni karakter tj. moguće im je pristupati samo unutar bloka u kojem su definirane.
- Java ne podržava `goto` izraz

Kontrola toka

- `boolean` izrazi vraćaju vrijednost `true` ili `false` i mogu se dodijeliti varijablama tipa `boolean`
  - `int x = 5;`
  - `boolean isFive = x == 5;`
- `boolean` izrazi mogu sadržavati varijable i konstante
- Relacijski operatori se koriste u `boolean` izrazima koji se koriste za izračunavanje `if/else` naredbi
- Kontrole toka u Javi omogućavaju odlučivanje koja naredba se izvršava sljedeća, a ove odluke se zasnivaju na `boolean` izrazima
- Relacijski operatori poput == su dobri za poređenje primitiva, ali ne i Stringova (i drugih objekata)
  - Porede vrijednosti primitiva, odnosno lokacije objekata u memoriji

Kontrola toka

- Za kombinaciju više `boolean` izraza u jedan, koriste se logički operatori &&, || i !
- && i || su kratkospojni logički operatori
  - Ukoliko je prvi izraz s lijeve strane *false* (*true*), nema potrebe za izračunavanjem ostalih izraza ako se koristi && (||) operator
- `switch` naredbu treba koristiti kada se testira:
  - Jednakost (a ne opseg)
  - Jedna vrijednost
  - Za fiksne poznate vrijednosti tokom vremena kompajliranja
  - `int`, `short`, `byte`, `char` ili `String`

Kontrola toka

- U `for` petlji svaki segment zaglavlja je opcioni, međutim:
  - ako se izostavi inicijalizacija – moguće da ne postoji brojač petlje
  - ako se izostavi uslov – uslov petlje se smatra tačnim (*true*) – beskonačna petlja
  - ako se izostavi ažuriranje – brojač petlje zadržava istu vrijednost.
  - U skladu s navedenim `for(;;)` – ispravna beskonačna petlja
- Varijabla koja se deklariše u `for` petlji je lokalna varijabla i ne može joj se pristupiti izvan `for` petlje
  - Ako je varijabla istog imena deklarisana prije `for` petlje javiće se greška
  - Varijable istog imena mogu postojati u različitim `for` petljama koje nisu ugniježdene

Kontrola toka

- `while` petlja se vrlo često koristi za implementaciju korisničkog unosa sa komandne linije:

```
public static void main(String[] args) {
   Scanner console = new Scanner(System.in);
   int sum = 0;
   System.out.println("Unesite broj (-1 za kraj): ");
   int num = console.nextInt();
   while (num != -1) {
     sum = sum + num;
     System.out.println("Unesite broj (-1 za kraj): ");
     num = console.nextInt();
   }
   System.out.println("Suma iznosi " + sum);
}
```

break i continue u petljama


```
while(uslov){
    naredba1;
    naredba2;
    break;
    naredba3;
}
naredba;
```

Kontrola se prenosi na naredbu izvan petlje

[naredba izvan while petlje]

```
while(uslov){
    naredba1;
    naredba2;
    continue;
    naredba3;
    naredba4;
}
naredba;
```

Kontrola se prenosi na uslov petlje

Ove naredbe se preskaču u trenutnoj iteraciji

[naredba izvan while petlje]

```
for(i = 0; i < 10; i++){
    naredba1;
    naredba2;
    continue;
    naredba3;
    naredba4;
}
```

Kontrola se prenosi na izraz ažuriranja

Ove naredbe se preskaču u trenutnoj iteraciji

Nizovi

- Svi Java tipovi koji ne spadaju u primitivne tipove, spadaju u reference pa tako i nizovi elemenata.
- Java niz je kolekcija elemenata istog tipa. Niz se deklarira specificiranjem tipa elemenata koji se nalaze u nizu nakon čega se obavezno upisuje [] npr
  - `int[] a;`
  - `double[] b;`
- Gornje deklaracije definišu varijable `a` i `b` kao nizove cijelih tj realnih brojeva ali ne vrše alokaciju memorije.
  - `a` i `b` su samo reference na nizove datih tipova
- Za alokaciju prostora za elemente niza koristi se operator `new` i to
  - `int[] a; a = new int[15]; // ili`
  - `double[] b = new double[5];`

Nizovi inicijalizacija

- U novonastalom nizu elemenata, inicijalne vrijednosti elemenata se postavljaju na default vrijednosti u skladu sa njihovim tipovima:
  - `boolean – false`
  - `char - '\u0000‘`
  - `byte, int, short, long, float, double – 0`
  - Tip reference – `null`
- Elemente niza moguće je inicijalizirati eksplicitno pomoću liste elemenata unutar zagrada {} i to
  - `int[] a = {2, 5, 10};`
- Niz se indeksira kao u C++ navođenjem indeksa unutar []. Indeksiranje počinje od indeksa 0.
  - `int[] a = new int[4];`
  - `a[0] = 4;`
  - `int b = a[2];`

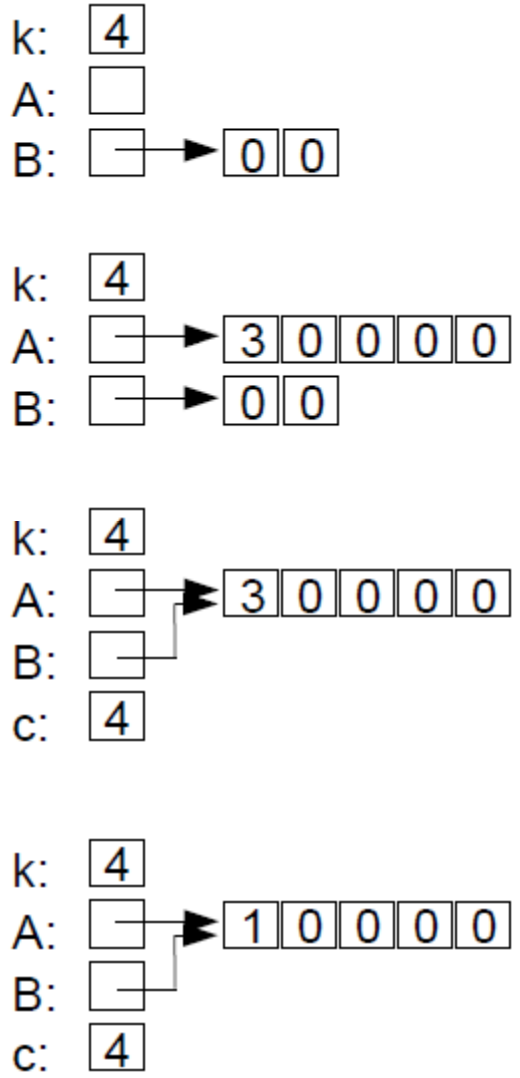
Java varijable i memorija


```
int k = 4;
int[] A;
int[] B = new int[2];

A = new int[5];
A[0] = 3;

B = A;
int c = k;

B[0] = 1;
```



- Varijable tipa referenci samo pokazuju na lokaciju na kojoj se nalazi objekat.
- Objekat za kojeg nema aktivnih referenci JVM automatski uklanja (garbage collection)

Nizovi kopiranje

- Funkcija
  - `System.arraycopy(a, aInd, b, bInd, br);`
- kopira `br` elemenata niza `a` od indeksa `aInd` u niz `b` od indeksa `bInd`. Npr

```
int[] a = {0, 1, 23, 4}, b = new int[3];
System.arraycopy(a,1,b,0,3);
for (int i = 0; i < b.length; ++i)
  System.out.println(b[i]);
```

- Dimenzije moraju biti kompatibilne i elementi nizova moraju biti istog tipa.
- Broj elemenata u nizu se može dobiti korištenjem `length` svojstva niza, npr. `b.length`

Prolazak kroz niz

- Iteriranje, ili prolazak kroz niz, podrazumijeva pristup svakom elementu niza
- Korištenjem `for` petlje i `length` svojstva niza kao uslova kraja petlje

```
String imena[]=new String[“Dino”, “Deni”, “Damir”];
for (int idx = 0; idx < imena.length; idx++){
 System.out.println(imena[idx]);
}
```

- Korištenjem `for-each` petlje
  - U svakoj iteraciji petlje sljedeći element niza se dohvaća i pohranjuje u iteracijsku varijablu čiji tip mora biti isti kao i tip elemenata niza

```
for(String ime: imena){
 System.out.println(ime);
}
```

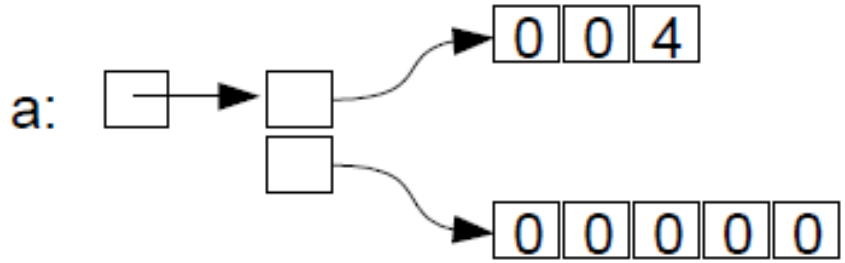
ArrayIndexOutOfBoundsException

- Niz ima fiksnu dužinu i njegovi indeksi su u intervalu [0, n-1], gdje je n broj elemenata niza
- Ukoliko se navede indeks koji je negativan ili veći ili jednak od veličine niza, JVM izbacuje `ArrayIndexOutOfBoundsException`
- Ova greška se javlja samo u vremenu izvršavanja (*run-time*), što znači da je Java kompajler ne provjerava tokom kompajliranja
- Ako se ova greška ne obradi, program se završava

```
Array length: 7
Exception in thread "main"
java.lang.ArrayIndexOutOfBoundsException: 10
        at arraysdemo.ArraysDemo.main(ArraysDemo.java:21)
Java Result: 1
```

Višedimenzionalni nizovi

- Višedimenzionalni nizovi se definiraju dodavanjem dodatnih [] za svaku novu dimenziju niza. Npr za 2D niz
  - `int[][] a = new int[2][3];`
  - `int[][] b = { {0,2}, {5,4} };`
- Niz ne mora biti pravougaonog oblika npr
  - `int[][] a;`
  - `a = new int[2][];`
  - `a[0] = new int[3];`
  - `a[1] = new int[5];`
  - `a[0][2] = 4;`



- Java višedimenzionalni nizovi nisu u kontinualnom bloku memorije.

Tekstualni primitivni tip

- Jedini primitivni tekstualni tip podatka je `char`, koji se koristi za pohranu jednog karaktera (16 bita)
  - `char velicina = 'M';`
- Karakteri se mogu nanizati kako bi se kreirale rečenice
  - Neefikasno:

```
char slovo1 = 'H';
char slovo2 = 'e';
char slovo3 = 'l';
char slovo4 = 'l';
char slovo5 = '0';
System.out.println(slovo1 + slovo2 + slovo3 + slovo4 +
  slovo5);
```

  - Efikasnije:

```
String pozdrav = "Hello world!";
System.out.println(pozdrav);
```

Karakteri vs Stringovi

- `char` je manijenjen za pojedinačne karaktere
  - Koristi se jednostruki navodnik
  - `char velicina1 = ‘S’;`
  - `char velicina2 = ‘M’;`
  - `char velicina3 = ‘L’;`
- `char` ne može rukovati sa više karaktera
  - `char velicina4 = ‘XL’;`
  - `char velicina5 = “XXL”;`
- Stringovi rukuju sa više karaktera
  - `String velicina6 = “XXXL”;`
- `char` je ključna riječ za primitivni tip podatka, piše se malim slovima i obojena je u IDE-u
- Stringovi su objekti a ne primitivi i tipovi objekata se pišu sa prvim velikim slovom

Klasa String

- Za rad sa sekvencama unicode karaktera Java obezbjeđuje u standardnoj biblioteci klasu `String`.
- Sekvenca karaktera između navodnika je instanca klase `String`.
  - `String grad = "Rim";`
- Nakon kreiranja obekat grad više nije moguće promjeniti tj `String` objekti su *immutable*.
- Operator + je definiran na način da povezuje dva stringa. Ostali tipovi se konvertiraju u string prije nadovezivanja
  - `String a = "pozdrav", b = "\tsvima"; int c = 5;`
  - `String r = a + b + c; // r = "pozdrav\tsvima5"`

Klasa String

- Zbog implicitne konverzije tipova prilikom spajanja stringova, voditi računa kada se koristi izračunavanje:
  - `String totalPrice = “Total: $” + 3 + 2 + 1;`
  - `System.out.println(totalPrice);  //Total: $321`
  - Ali:
  - `String totalPrice = “Total: $” + (3 + 2 + 1);`
  - `System.out.println(totalPrice);  //Total: $6`
- Specijalni karakteri u stringovima
  - Karakter kojem prethodi *backslash* (\) je *escape* sekvenca i ima posebno značenje kompajleru
  - Tab (\t), backspace (\b), novi red (\n), carriage return (\r), formfeed (\f), jednostruki navodnik (\’), dvostruki navodnik (\”), backslash (\\)

Klasa String

- Neki metodi
  - `int length();` vraća broj karaktera u stringu
  - `char charAt(int k);` vraća k-ti karakter u stringu
  - `boolean equals(Object S);` vraća tačno ako je string identičan stringu `S`
  - Operator == testira da li su stringovi pohranjeni na istoj lokaciji u memoriji - za poređenje sadržaja koristiti metod `equals`
  - `String substring(int p);`
  - `String substring(int p, int k);`
    - Vraća substring stringa između indeksa p i kraja stringa, tj indeksa p i k-1 indeksa.

Primjer


```
class Test
{
   public static void main(String[] args)
   {
     String a = "Pocetak";
     String b = "Posta";
     if (a.substring(0,2) == b.substring(0,2))
            System.out.println("isti1");
     if (a.substring(0,2).equals(b.substring(0,2)))
            System.out.println("isti2");
   }
}
```

Korisnički unos

- Postoji nekoliko načina za unos podataka od strane korisnika u Javi:
  - Swing-ov JOptionPane
  - Scanner klasa
  - JavaFX (nasljednik Swing-a)

- Najjednostavniji način je korištenjem `JOptionPane` dijaloškog okvira
  - Unos se pohranjuje kao tip `String`
  - `String unos = JOptionPane.showInputDialog("Unesite nešto:");`
  - Ekvivalentno sa: `String unos = "Ovo je string";`
  - Mora se uključiti `javax.swing.JOptionPane` (`import` opcija će biti obrađena nešto kasnije)
  - Dijaloški okvir se može prilagoditi sa nešto složenijim oblicima `showInputDialog` metoda

Konverzija String tipa u numerički tip

- Bilo šta da korisnik unese u dijaloški okvir biće pohranjeno kao String tip podatka
- Ukoliko se zahtjeva brojčani unos, String se mora konvertovati u numerički tip podatka
- Konverzija String u int:
  - `int intVar = Integer.parseInt(“100”);`
- Konverzija String u double:
  - `double doubleVar = Double.parseDouble(“2.72”);`
- U ovakvim slučajevima se često može javiti `NumberFormatException` ukoliko unesena vrijednost ne može biti parsirana
  - `int intVar1 = Integer.parseInt(“Psići!”);`
- Obradu takvih grešaka i wrapper klase za sada treba zanemariti

Korisnički unos Scanner-om

- `Scanner` objekat otvara ulazni tok (*stream*) za prikupljanje unosa:
  - `System.in` priprema `Scanner` za unos sa konzole
- Nakon završetka rada sa `Scanner`-om treba ga zatvoriti
- Prilikom čitanja sa ulaznog toka `Scanner` traži tokene
- Tokeni su razdvojeni delimiterom
  - Podrazumijevani delimiter je prazno mjesto (*space*)
- Metode `Scanner` klase za čitanje sa ulaznog toka:
  - `nextInt()` – čita sljedeći token kao int
  - `nextDouble()` – čita sljedeći token kao double
  - `next()` – čita sljedeći token kao String
  - `nextLine()` – čita sljedeću liniju u fajlu
  - `findInLine("StringToFind")` – traži sljedeće pojavljivanje datog uzorka, ignorišući delimitere

Korisnički unos Scanner-om

- Greške koje se mogu pojaviti prilikom prihvata korisničkog unosa:
  - `InputMismatchException`
    - Javlja se kada unos ne može biti parsiran kao očekivani tip
  - `IllegalStateException`
    - Javlja se kada se ulaznom toku pristupa nakon što je zatvoren
  - `NullPointerException`
    - Javlja se ukoliko ne postoji ulazni tok iz kojeg se čita, npr. pogrešno napisan naziv fajla