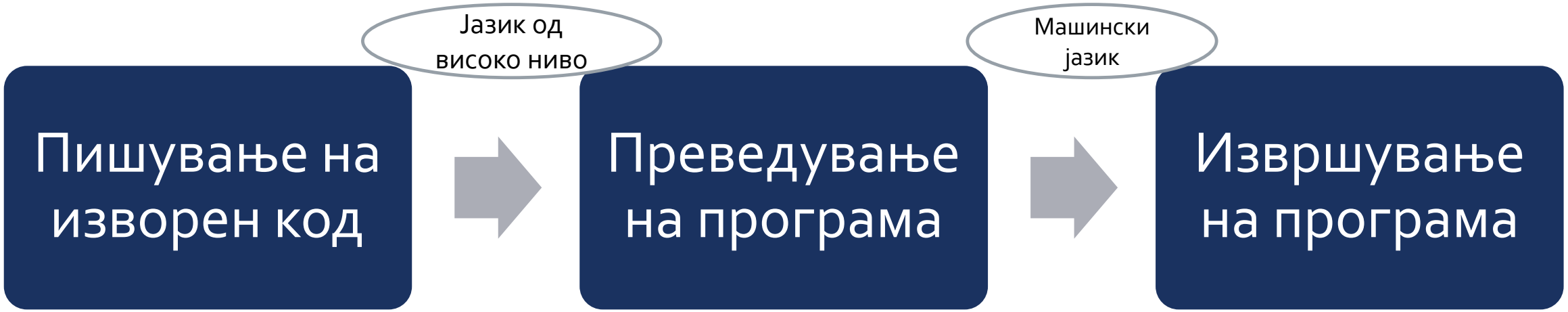
ПОДАТОЧНИ СТРУКТУРИ И АНАЛИЗА НА АЛГОРИТМИ

ЧАС 1: ВОВЕД ВО JAVA

АУДИТОРИСКИ ВЕЖБИ

БОЈАНА ВЕЛИЧКОВСКА

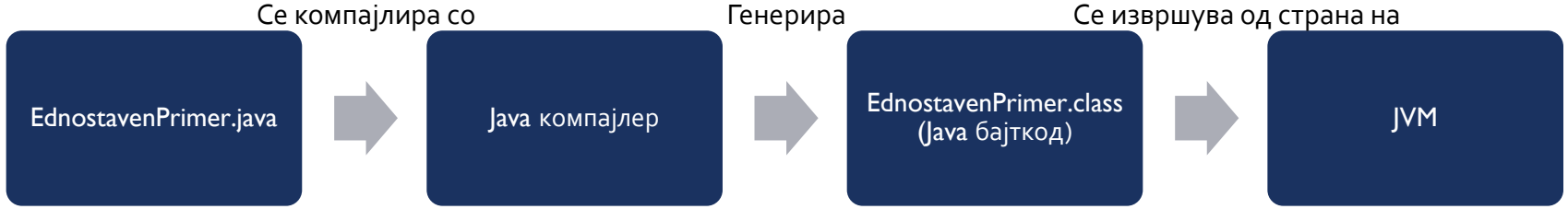
ПРОГРАМСКИ ЦИКЛУС



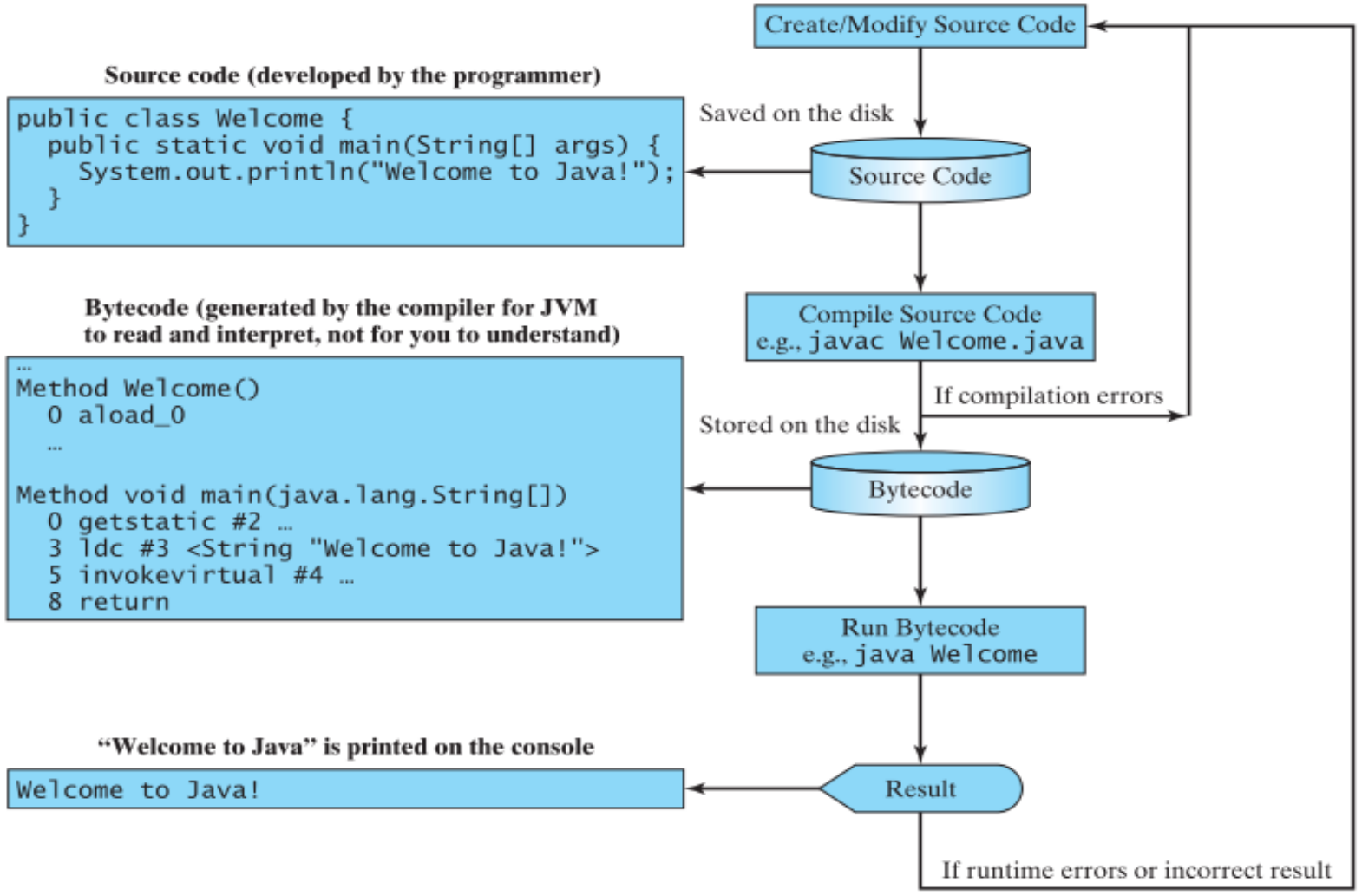
ПИШУВАЊЕ, КОМПАЈЛИРАЊЕ И ИНТЕРПРЕТИРАЊЕ

Во текстуален едитор:

- Изворниот код го пишуваме во текстуална датотека
- Програмите напишани во Java имаат **.java** наставка
- Команда за компајлирање: **javac**
  - > javac EdnostavenPrimer.java
- Компајлерот ги пријавува сите грешки кои би можеле да настанат при компајлирање
  - Грешките мора да се поправат и програмата треба да се искомпајлира одново
- Ако помине компајлирањето, на излез се добива Java бајткод (со .class наставка)
- Команда за извршување: **java**
  - > java EdnostavenPrimer

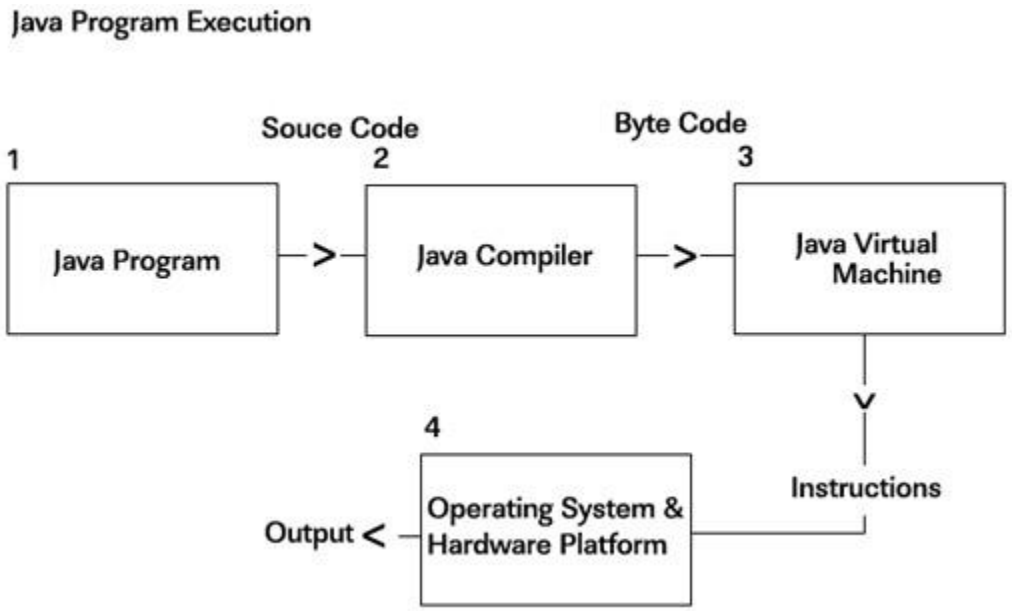


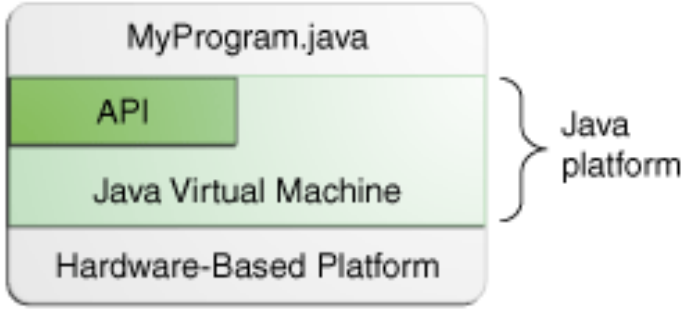
ПИШУВАЊЕ, КОМПАЈЛИРАЊЕ И ИНТЕРПРЕТИРАЊЕ



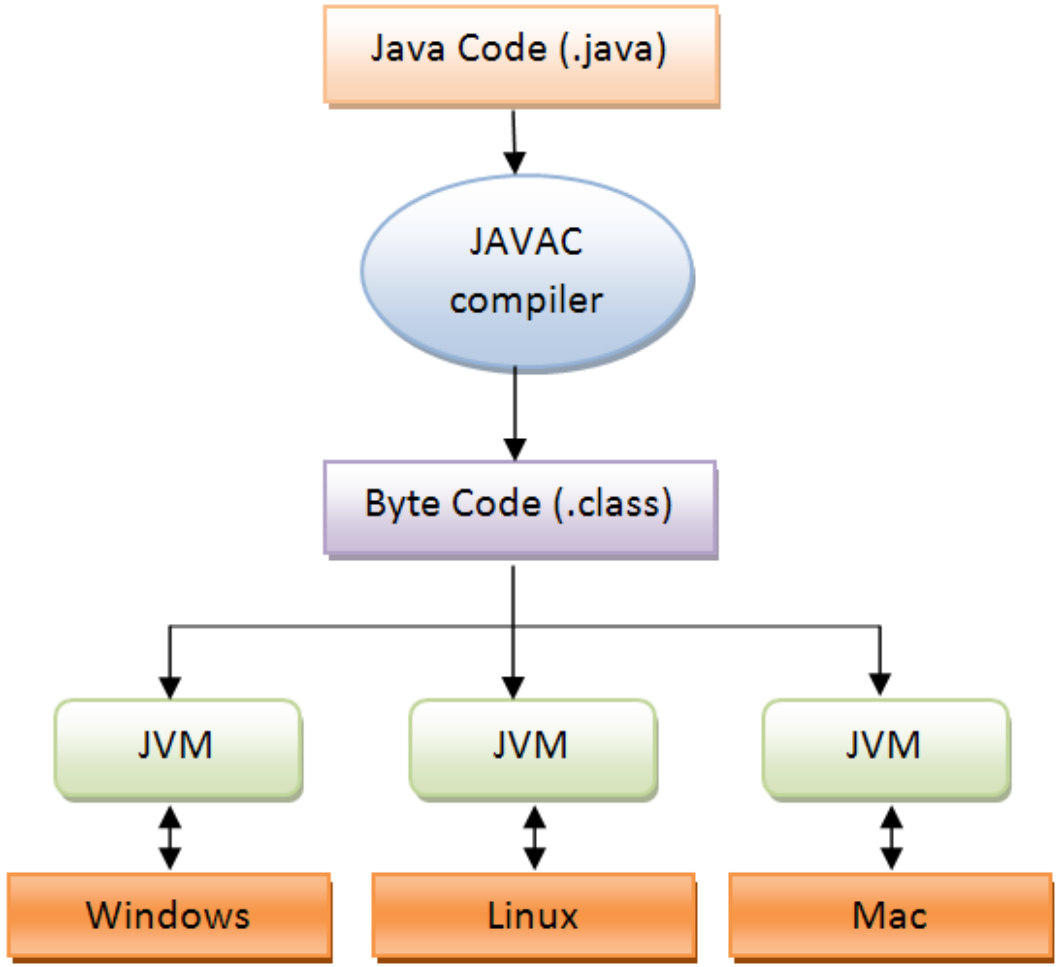
JAVA ТЕХНОЛОГИЈА

- Java програмски јазик
  - Јазик од високо ниво
  - Разбирлив за човекот
- Java бајткод
  - Јазик од ниско ниво
  - Множество од основни инструкции
  - Независен од платформа
- Java виртуелна машина (JVM)
  - Програма која интерпретира Java бајткод
  - Имплементации постојат за повеќе платформи (Windows, Linux, Mac)
- Java API
  - Збир од готови софтверски компоненти групирани во библиотеки





JAVA ПЛАТФОРМСКА НЕЗАВИСНОСТ



ОБЈЕКТНО – ОРИЕНТИРАНО ПРОГРАМИРАЊЕ

- Ја упростува комплексноста при програмирањето
- Големите проблеми ги дели на помали делови, со кои полесно се управува
- Се заснова на хиерархија од класи
  - Класа е структура која ги дефинира податоците и методите (функциите) кои управуваат со тие податоци.
  - Класа е нов податочен тип

ИЗВОРЕН КОД ВО JAVA (ПРИМЕР)

```
// Овој изворен код е именуван како EdnostavenPrimer.java
public class EdnostavenPrimer
{
   // Кодот печати на екран едноставна реченица
    public static void main(String[] args)
     {
        System.out.println(“Prva Java programa");
     }
}
```

КОМЕНТАРИ

- Игнорирани од страна на Java компајлерот
- // - коментар во една линија
- /* … */ - коментар во повеќе линии
- /** … */ –коментари кои и дозволуваат на програма наречена Javadoc автоматски да генерира софтверска документација

```
/*
*  Ova e komentar vo povekje linii
*/
/ /  Ova e komentar vo edna linija
```

ОБЈЕКТИ

- Објект – инстанца од класа
  - Чува податоци и обезбедува методи за пристап и промена на податоците
- Податочни членови - инстанцнирани променливи (состојба)
  - Чуваат податоци за објектот
  - Примитивни типови или објекти од други класи
- Методи (однесување)
  - Операции кои се извршуваат над податоците
  - Конструктори, процедури, функции
  - Го дефинираат однесувањето на објектот

ДЕКЛАРИРАЊЕ НА КЛАСИ

```
public class Counter
{
    protected  int count;
                /*    default  constructor   */
    Counter()  {count=0;}
                /*  get  метод  */
    public  int getCount()  { return  count;  }
                /*  set метод  */
    public  void  incrementCount() { count++;  }
                /*  set метод  */
    public  void  decrementCount()  {count--;  }
}
```

ПАКЕТИ

- Група од класи дефинирани во ист директориум

```
package package_name;
```

- Вклучување на надворешни пакети или класи

```
import packageName.className;

import  (packageName).*;
```

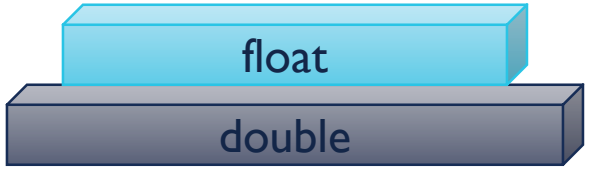
Java вклучува множество од пакети во својот API(Application Programming Interface)

ТИПОВИ НА ПОДАТОЦИ

- Податочниот тип определува:
  - како вредностите за даден податок се сместуваат во меморијата и колку меморија зафаќаат (зависно од платформата),
  - множеството вредности за податокот
  - операциите што може да се извршат со или над неговите вредности.

Цели броеви
-1, 0, 134, -765432

Знаци
'a', 'C', '\n', 'f'



Реални броеви
1.23, 1.23e3, -23.3546

Стрингови
"niza", "feit"

ОСНОВНИ (ПРИМИТИВНИ) ПОДАТОЧНИ ТИПОВИ

- boolean – true/false
- char – 16-битни Unicode знаци
- byte – 8-битни цели броеви
- short – 16-битни цели броеви
- int – 32-битни цели броеви
- long - 64-битни цели броеви
- float - 32-битни реални броеви
- double - 64-битни реални броеви

* Големината на податочниот тип зависи од процесорот кој ја извршува програмата

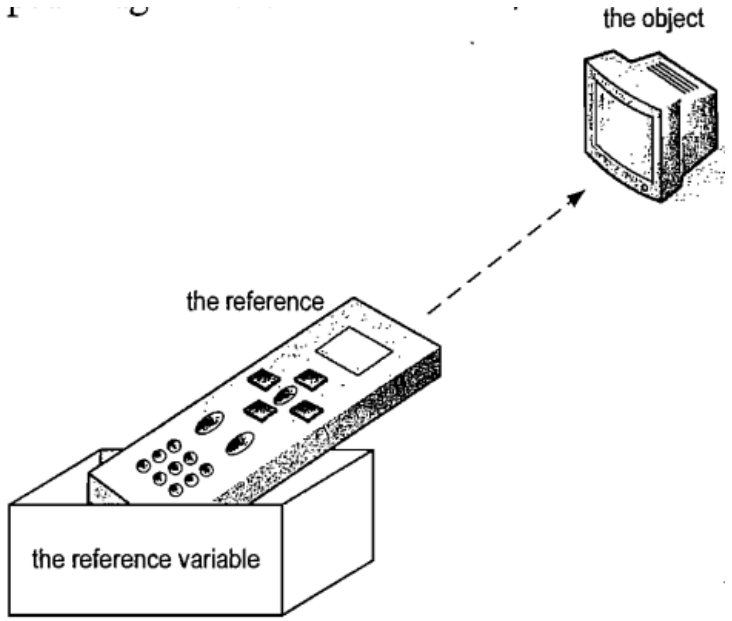
ОСНОВНИ (ПРИМИТИВНИ) ПОДАТОЧНИ ТИПОВИ

```
public class Base{
    public  static  void  main  (String[]  args)  {
    boolean  flag = true;
    char  ch = 'A';
    byte  b = 12;
    short  s = 24;
    int  i = 257;
    long  l = 890l;  /* забележете ја употребата на "l" на крај */
    float  f = 3.1415F;  /* забележете ја употребата на "F" на крај */
   double  d = 2.1828;
    System.out.println("flag = " +  flag);
// + е конкатенатор на стрингови
    System.out.println("ch = "  +  ch);
    System.out.println("b = "  +  b);
    System.out.println("s = "  +  s);
    System.out.println("i = " +  i);
    System.out.println("l = " +  l);
    System.out.println("f = "  +  f);
    System.out.println("d = " +  d);
}
}
```

КРЕИРАЊЕ НА ОБЈЕКТИ

- Се креираат со помош на операторот new и повик на конструкторот за тој објект
- Враќаат референца (меморискa адреса) кон новокреираниот објект
- Не смее да е abstract, static или final

```
public static void main(String[] args) {
        Counter c;
        Counter d = new Counter();
        c = new Counter();
        System.out.println("c count = " + c.count);
        d = c;
    }
```



- Променливите кои не добиваат вредност во конструкторот, автоматски се поставуваат на 0, null или false

НУМЕРИЧКИ ОБЈЕКТИ

- Java креира обвиткувачка класа за сите нумерички основни типови на податоци
- На тој начин може да се користат како објекти

Base Type	*Class Name*	*Creation Example*	*Access Example*
byte	Byte	n = new Byte((byte)34);	n.byteValue()
short	Short	n = new Short((short)100);	n.shortValue()
int	Integer	n = new Integer(1045);	n.intValue()
long	Long	n = new Long(10849L);	n.longValue()
float	Float	n = new Float(3.934F);	n.floatValue()
double	Double	n = new Double(3.934);	n.doubleValue()

ТЕКСТУАЛНИ ОБЈЕКТИ

- Java дефинира специјална класа за работа со текстуални објекти, String
  - String s1 = “Java”;
  - String s2 = new String (“Welcome to Java”);
  - char [ ] chArray = {‘H’, ‘i‘, ‘!’};

    String s3 = new String (chArray);
- Секој знак во рамки на стрингот може да биде референциран со индексот за позцијата (првиот знак е на позиција нула)
- Конкатенирање на стрингови – со операторот +

  String s = “dino” + “saur”;

КЛАСАТА STRING

- Методи:
  - boolean equals (String s);

  Операторот == проверува дали референците за двата стринга покажуваат кон ист објект, не ја проверува еднаквоста на нивната содржина
  - int compareTo (String s);
  - boolean startsWith(String prefix);
  - boolean endsWith(String suffix);
  - int length()
  - char charAt (int index);
  - String concat (String s1); //истото се постигнува со операторот +
  - String substring (int beginInd, int endInd);

КЛАСАТА STRING

- String toLowerCase ();
- String toUpperCase ();
- String trim(); //ги отстранува празните места од двете страни
- String [ ] split (String delimiter); //враќа низа од стрингови одвоени според delimiter
- replace, replaceFirst, replaceAll
- indexOf, lastIndexOf

КОНВЕРЗИЈА МЕЃУ STRING И CHAR [ ]

- char [ ] chars = “Java”.toCharArray();

//chars = {‘J’, ‘a’, ‘v’, a’};

- String str = new String(new char [ ] {‘j’, ‘a’, ‘v’, ‘a’, ‘1’, ‘3’, ‘0’, ’1’});

// str = “jaa1301”;

НЕСООДВЕТНОСТ НА ТИПОВИ НА ПРОМЕНЛИВИ И КАСТИРАЊЕ

- String pet = 5; //грешка

  test.java.2: incompatible types

  found: int

  required: java.lang.String

- int a = 2; //a = 2

  double b = 2; //b = 2.0

  int a = 15.5 //грешка

  int a = (int) 15.5; //a = 15, кастирање

  double b = 2/5; //b = 0.0

  double b = (double)2/5; //b = 0.4

КАСТИРАЊЕ

- Кастирање е операција која дозволува промена на тип на променлива
- Синтакса:

(nov_tip) izraz

```
double dl = 3.2;
double d2 = 3.9999;
int il = (int)dl; // il ima vrednost 3
int i2 = (int)d2; // i2 ima vrednost 3
double d3 = (double)i2; // d3 ima vrednost 3.0
 il = 3;
 i2 = 6;
dresult = (double )il / (double )i2; // dresult ima vrednost 0.5
dresult = il / i2; // dresult ima vrednost 0.0
```

ИМПЛИЦИТНО КАСТИРАЊЕ

```
int iresult, i = 3;
double dresult, d = 3.2;
dresult = i / d; // dresult е 0.9375. Истиот е кастиран во тип double
iresult = i / d; // губење на прецизност → грешка при компајлирање
iresult = (int) i / d; // iresult е 0, децималниот дел се губи
```

- Имплицитното кастирање секогаш важи за стрингови
  - Кога стринг се конкатенира со каков било друг објект или основен податочен тип, тој се конвертира во стринг

ЕКСПЛИЦИТНО КАСТИРАЊЕ

- Експлицитно кастирање на стрингови не е дозволено

```
String  s = (String) 4.5;  // погрешно!
String  t  =  “Value  =“  +  (String)  13;  // погрешно!
String  u = 22;  // погрешно!
```

- За конверзија во стринг се користи методот **toString()**

```
String s  = “ ” +  4.5;  // точно, но не е добар стил на програмирање
String  t  =  “Value  =  “  +  13;  //  во ред
String  u =  Integer.toString(22);  // во ред
```

ТИПОВИ НА ПРОМЕНЛИВИ - ПРОШИРУВАЊЕ

- Кастирање помеѓу char и int
  - char ch = (char)65.25; // декадно 65 се доделува на ch

    System.out.println(ch); // ch е знакот A
  - int i = (int)'A'; // Кодот на знакот A во Unicode табелата се доделува на i

    System.out.println(i); // i сега е 65
- Сите аритметички операции може да се извршат над char променливи
- char операнд автоматски се кастира во број, ако другиот операнд е број или знак
- Ако другиот операнд е стринг, тогаш знакот се прилепува на стрингот

ТИПОВИ НА ПРОМЕНЛИВИ - ПРОШИРУВАЊЕ

- int i = '2' + '3'; // (int)'2' е 50, а (int)'3' е 51

```
System.out.println("i е " + i);          // i е 101
int j = 2 + 'a';                         // (int)'a' е 97
System.out.println("j е " + j);          // j е 99
System.out.println(j + " е Unicode za znakot“ + (char)j);
System.out.println(“Den" + '2');
```

- Излез:

```
i е 101
j е 99
99 е Unicode za znakot c
Den2
```

ТИПОВИ НА ПРОМЕНЛИВИ - ПРОШИРУВАЊЕ

- При конкатенација на стрингови, ако некој од членовите не е стринг, автоматски е конвертиран во таков

```
// Конкатенација на три стринга
String message = "Welcome " + "to " + "Java";
// Стрингот Chapter се конкатенира со бројот 2
String s = "Chapter" + 2; // s станува Chapter2
// Стрингот Supplement се конкатенира со знакот B
String s1 = "Supplement" + 'B'; // s1 станува SupplementB
```

СПЕЦИЈАЛНИ ЗНАЦИ

- \t таб

  \n нов ред

  \\ коса црта

  \' единечни наводници

  \" двојни наводници
- System.out.println("He said \"Java is fun\"");

КОЈ Е ИЗЛЕЗОТ ОД СЛЕДНИВЕ НАРЕДБИ?

```
System.out.println("1" + 1);
System.out.println('1' + 1);
System.out.println("1" + 1 + 1);
System.out.println("1" + (1 + 1));
System.out.println('1' + 1 + 1);
```

Unicode вредноста на ‘1’ е 49

ОПЕРАТОРОТ ТОЧКА

- Се користи за пристап до членовите на класата за даден објект (методи и променливи)
- Ако постојат повеќе методи со исто име **(преоптоварување)**, се употребува оној метод кој има ист број и тип на аргументи како оние кои се проследуваат при повикот

```
oven.cookDinner();

oven.cookDinner(food);

oven.cookDinner(food,seasoning);
```

- Не е дозволено два методи со исто име, број и тип на аргументи да враќаат различен тип

МОДИФИКАТОРИ НА ПРОМЕНЛИВИ

- Модификатори кои ја контролираат видливоста на променливите:
- public - секој може да пристапи
- protected – може да пристапат само методи од истиот пакет (класи дефинирани во еден пакет) или подкласите
- private - пристап имаат само методите од класата во која е дефинирана променливата
- Ако не е дефиниран пристап, се смета дека е friendly, односно до променливата може да пристапат методите од класи во истиот пакет

МОДИФИКАТОРИ НА ПРОМЕНЛИВИ

- Модификатори кои ја контролираат употребата на променливите:
- static – променлива која се поврзува со класата, а не со инстанца од класата
  - Чуваат глобални информации за класата
  - Постојат дури и ако не е креирана инстанца од класата
- final – променлива на која мора да и се додели почетна вредност и понатаму истата не смее да се менува
  - Ако е променлива од основен тип, тогаш е константа
  - Ако е референца кон објект, тогаш мора да покажува на истиот објект

МОДИФИКАТОРИ НА ПРОМЕНЛИВИ

```
public class Gnome {
    // Инстанцирани променливи:
public String name;
public int age;
public Gnome gnomeBuddy;
private boolean magical = false;
protected double height = 2.6;
public static final int MAX_HEIGHT = 3;
```

МОДИФИКАТОРИ НА ПРОМЕНЛИВИ

```
// Конструктори:
Gnome(String nm, int ag, Gnome bud, double hgt) {
   name = nm;
   age = ag;
   gnomeBuddy = bud;
   height = hgt;
}
Gnome() {
name = "Rumple!!";
age = 204;
gnomeBuddy = null;
height = 2.1;
}
```

МОДИФИКАТОРИ НА ПРОМЕНЛИВИ

```
/ Методи:
public static void makeKing (Gnome h) {
h.name = "King " + h.getRealName();
h.magical = true; // Само Gnome класата може да пристапи до magical
}
public void makeMeKing () {
name = "King " + getRealName();
magical = true; }
public boolean isMagical() { return magical; }
public void setHeight(int newHeight) { height = newHeight; }
public String getName() {return "I won't tell "; }
public String getRealName() { return name; }
public void renameGnome(String s) {name = s;}
}
```

ЕНУМЕРАЦИЈА

- Тип на податок кој прима само точно дефинирани вредности од дадено множество на имиња
- Се декларираат со клучниот збор **enum**
- **modifier enum name { val1 , val2, ... , valn-1 };**

public enum Day { MON, TUE, WED, THU, FRI, SAT, SUN };

- Ако податокот го употребиме во текстуален израз, **Java** го користи неговото име
- Со помош на методот **valueOf** се добива вредноста на податокот која е иста како и неговиот стринг

ЕНУМЕРАЦИЈА

```
public enum Day {MON, TUE, WED, THU, FRI, SAT, SUN};
public static void main(String[] args) {
Day d = Day.MON;
System.out.println("Initially d is " + d);
d = Day.WED;
System.out.println("Then it is " + d);
Day t = Day.valueOf("WED");
System.out.println("I say d and t are the same: " + (d == t));
}
```

МЕТОДИ

modifiers type name(type0 parameter0, ..., typen-l parametern_l) {

// method body. . .

}

- Модификатори:
  - public – секој може да го повика
  - protected – може да го повикаат само методи во ист пакет или подкласи на дадена класа
  - private – може да го повикаат само методи од иста класа
  - Ако не е специфициран тип, се подразбира дека е **friendy**, односно може да биде повикан само од објекти во класи кои се наоѓаат во ист пакет

МЕТОДИ

- Дополнителни модификатори:
  - abstract – нема тело
  - final – не може да биде препокриен од **подкласа**
  - static – поврзан е со класата, а не со нејзина инстанца (се користи за промена на вредност на статички променливи)
    - main методот е статичен
- Процедура – метод кој не враќа ништо (од void тип е)
- Функција – метод кој враќа вредност

ОПЕРАТОРИ

- Аритметички: +,-,*,/,%,++,--
- Релациони: <,<=,>,>=,==,!=
  - == и != може да се употребуваат и за споредба на објекти
- Логички: &&, ||, !
- Операторот + се користи и за конкатенирање (спојување) на стрингови

```
String rug = "carpet";
String dog = "spot";
String mess = rug + dog;
String answer = mess + “will cost me ” + 5 + “ hours! “;
```

КОНТРОЛНИ СТРУКТУРИ

- if…else

```
if (snowLevel < 2) {
goToClass();
comeHome();
}
else if (snowLevel < 5) {
goSledding();
haveSnowballFight();
}
else
stayAtHome();
```

КОНТРОЛНИ СТРУКТУРИ

- switch

```
switch (d) {
case MON: System.out.println("This is tough."); break;
case TUE: System.out.println("This is getting better."); break;
case WED: System.out.println("Half way there."); break;
case THU: System.out.println("I can see the light."); break;
case FRI: System.out.println("Now we are talking."); break;
default: System.out.println("Day off! "); break;
}
```

КОНТРОЛНИ СТРУКТУРИ

- while, do…while

```
public  void  waterCarrots () {
Carrot  current  = garden.findNextCarrot();
while  (!waterCan.isEmpty())  {
water (current,  waterCan);
current = garden.findNextCarrot();
}
}
```

КОНТРОЛНИ СТРУКТУРИ

- for

```
public void eatApples (Apples  apples)  {
numApples  =  apples.getNumApples();
for (int  x= 0;  x < numApples;  x++)  {
eatApple(apples.getApple (x));
spitOutCore();
}
}
```

BREAK

- break – излегува од најблискиот switch, while, do…while, for
- break со лабела – излегува од циклусот означен со лабела

```
int i;
tuka: for(i=0;i<10;i++)
{
    for(int j=1;j<2;j++)
        if(i==3)
            break tuka;
}
System.out.println(i);
```

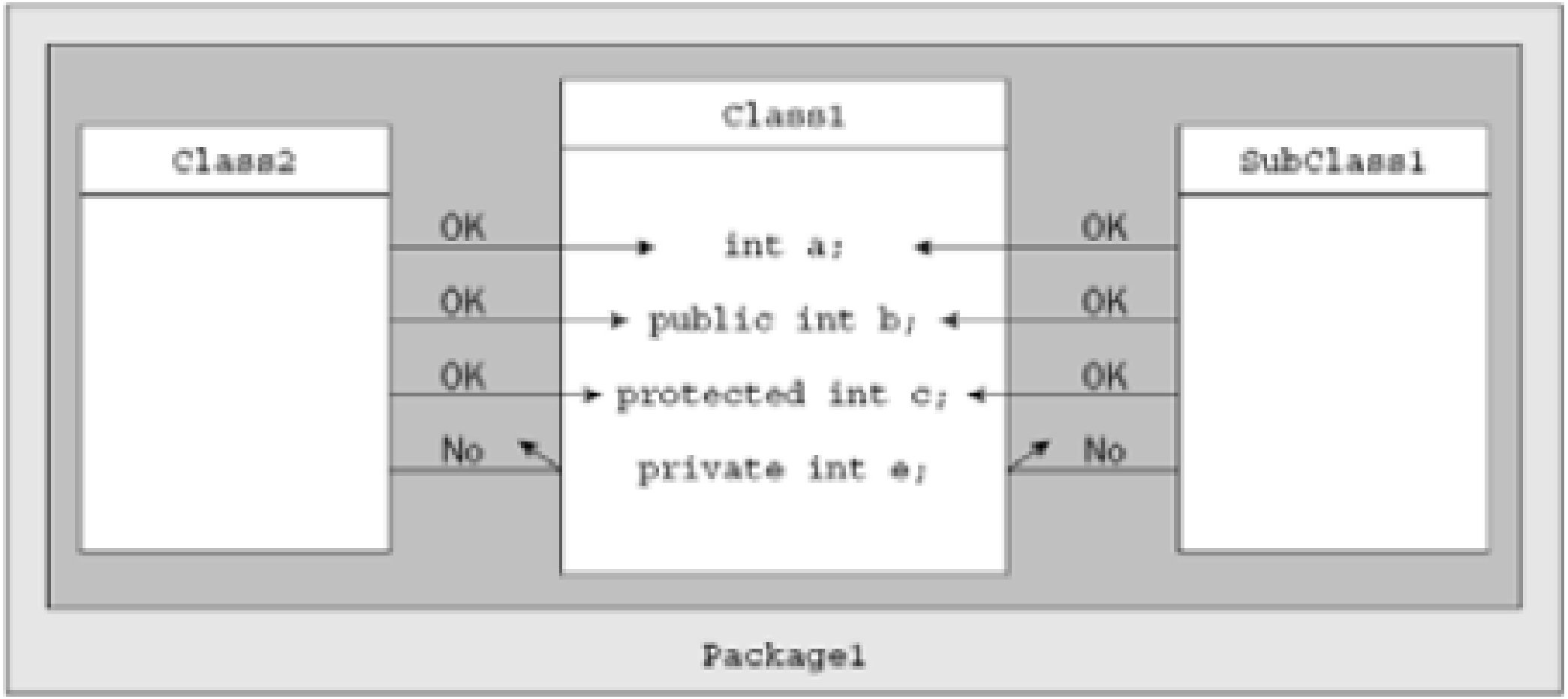
CONTINUE

- Се користи само во циклуси (while, do…while, for)
- Може да се користи и лабела
- Се прескокнуваат останатите наредби во рамки на тековната итерација

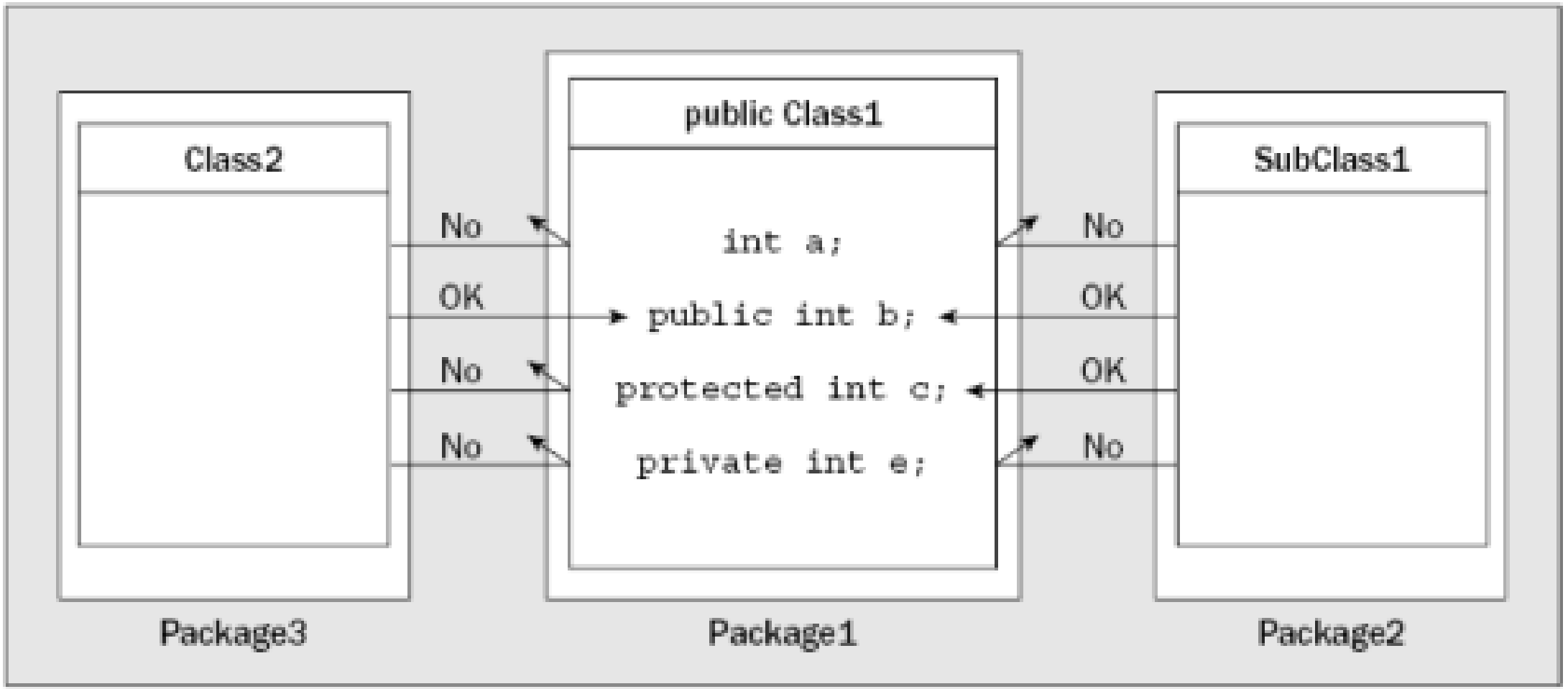
МОДИФИКАТОРИ НА КЛАСИ

- Опционални клучни зборови кои претходат на клучниот збор **class**
- **public** класата може да биде инстанцирана или проширена во рамки на истиот пакет или во рамки на класи кои ја вклучуваат (**import**) класата
  - **public** класите се декларираат во посебна датотека **classname.java**, каде "**classname**" е името на класата
  - Ако класата не е **public**, тогаш се смета дека е **friendly** (може да биде инстанцирана или проширена во рамки на истиот пакет)
- **abstract** класата има абстрактни методи (методи кои немаат тело)
  - Интерфејс – класа која содржи само абстрактни методи и нема полиња
- **final** класата не може да биде подкласа

МОДИФИКАТОРИ ЗА ПРИСТАП

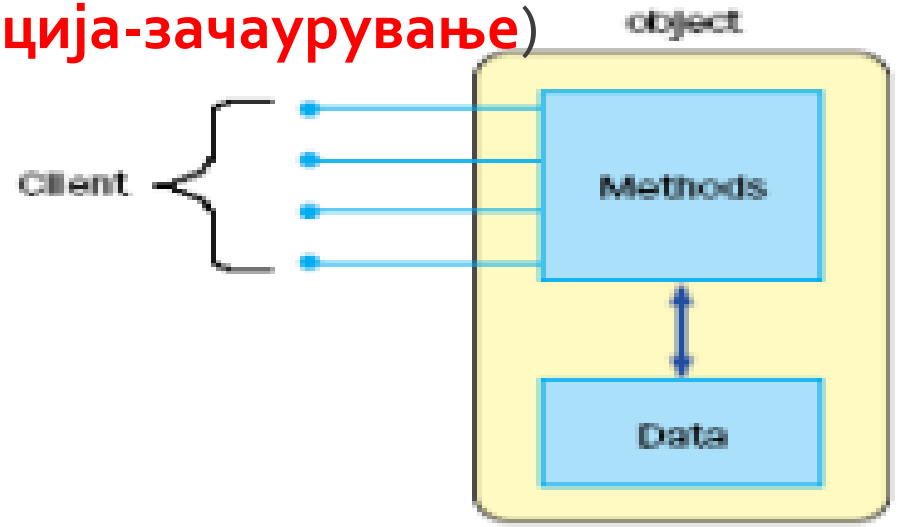


МОДИФИКАТОРИ ЗА ПРИСТАП



ЗАДАЧА 1 (МОДИФИКАТОРИ ЗА ПРИСТАП)

- Да се напише класа Coins во која треба да се дефинира метод flip() кој генерира случаен број (0 или 1). Да се напише метод isHeads() кој враќа точно доколку падне глава. Да се напише метод toString() кој враќа стринг "Heads" или "Tails".
- Да се напише класа CountFlips која симулира вртење на паричка 500 пати и ги брои главите и опашките.
- **Забелешки**: кодот на клиентот(CountFlips) не смее директно да пристапи до членовите на објектот од класата Coins (**енкапсулација-зачаурување**)



ЗАДАЧА 1 (МОДИФИКАТОРИ ЗА ПРИСТАП)

```
import java.util.Random;
class Coin {
        private final int HEADS = 0;
        private final int TAILS = 1;
        private int face;
public Coin () { flip(); }
public void flip () { face = (int) (Math.random() * 2); }
public boolean isHeads () {return (face == HEADS); }
```

ЗАДАЧА 1 (МОДИФИКАТОРИ ЗА ПРИСТАП)

```
public String toString() {
        String faceName;
        if (face == HEADS) faceName = "Heads";
        else faceName = "Tails";
        return faceName;
} }
```

ЗАДАЧА 1 (МОДИФИКАТОРИ ЗА ПРИСТАП)

```
public class CountFlips {
    public static void main (String[] args) {
        final int NUM_FLIPS = 500;
        int heads = 0, tails = 0;
        Coin myCoin = new Coin();
        for (int count=1; count <= NUM_FLIPS; count++) {
            myCoin.flip();
            if (myCoin.isHeads()) heads++;
            else tails++;
        }
    System.out.println ("The number of heads: " + heads);
    System.out.println ("The number of tails: " + tails);
} }
```

TOSTRING МЕТОД ВО JAVA

- toString() методот врши конверзија на објект во неговата текстуална репрезентација
- Печатењето на кој било објект во позадина го повикува методот toString().
- Препокривањето на овој метод овозможува објектот да се испечати во посакуваниот облик

TOSTRING МЕТОД ВО JAVA

```
class Student{
 int rollno;
 String name;
 String city;
 Student(int rollno, String name, String city){
 this.rollno=rollno;
 this.name=name;
 this.city=city;   }
 public static void main(String args[]){
   Student s1=new Student(101,"Raj","lucknow");
   Student s2=new Student(102,"Vijay","ghaziabad");
   System.out.println(s1);  //kompajlerot povikuva s1.toString()
   System.out.println(s2);  //kompajlerot povikuva  s2.toString()
 } }
```

Output:Student@1fee6fc

Student@1eed786

TOSTRING МЕТОД ВО JAVA

```
...
public String toString(){        //prepokrivanje na metodot toString()
  return rollno+" "+name+" "+city;
 }
 public static void main(String args[]){
   Student s1=new Student(101,"Raj","lucknow");
   Student s2=new Student(102,"Vijay","ghaziabad");

   System.out.println(s1);   //kompajlerot povikuva s1.toString()
   System.out.println(s2);   // kompajlerot povikuva s2.toString()
 }
}
```

Output:101 Raj lucknow

102 Vijay ghaziabad

ЗАДАЧА 3 (ПРЕОПТОВАРЕНИ КОНСТРУКТОРИ)

```
public int roll () { faceValue = (int) (Math.random() * numFaces) + 1;
                     return faceValue; }

public int getFaceValue () { return faceValue; } }
```

ЗАДАЧА 3 (ПРЕОПТОВАРЕНИ КОНСТРУКТОРИ)

```
public class SnakeEyes {
    public static void main (String[] args) {
        final int ROLLS = 500;
        int snakeEyes = 0, num1, num2;
        Die die1 = new Die();
        Die die2 = new Die(20);
            for (int roll = 1; roll <= ROLLS; roll++) {
                num1 = die1.roll();
                num2 = die2.roll();
        if (num1 == 1 && num2 == 1) snakeEyes++; }
System.out.println ("Ratio: " + (float)snakeEyes/ROLLS);
} }
```

НИЗИ

- **Декларирање**
  - element_type [] array_name = {iniLvaL0,iniLvaL1, ...,iniLvaLN-l};

```
int []  primes = {2,  3,  5,  7,  11,  13,  17,  19,  23,  29};
```

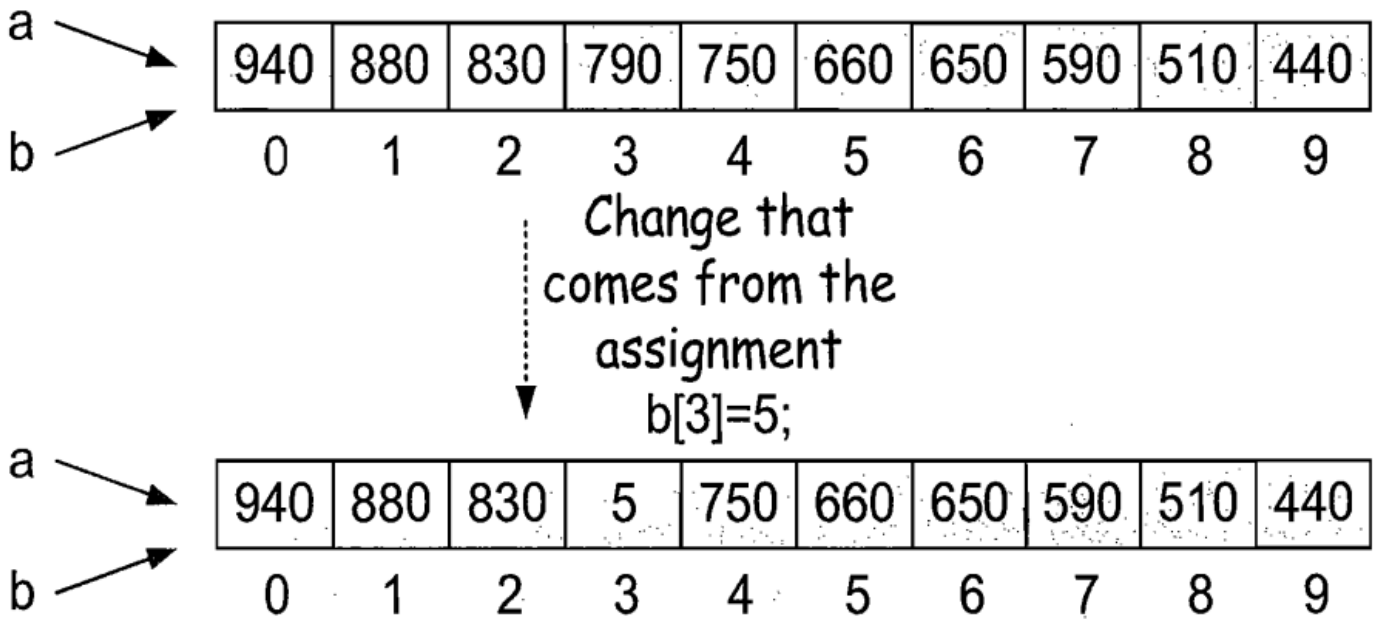
  - element_type [] array_name; //сите вредности се иницијализирани на нула

```
array_name=new elementJype[length]
double []  a;
/ /      ... various  steps  ...
a =  new  double[10];
for  (int  k=0;  k < a.length;  k++)  {  //length ја враќа должината на низа од било каков тип
a[k]  =  1.0;
}
```

НИЗИ

int [] a = {940, 880, 830, 790}; int [] b;

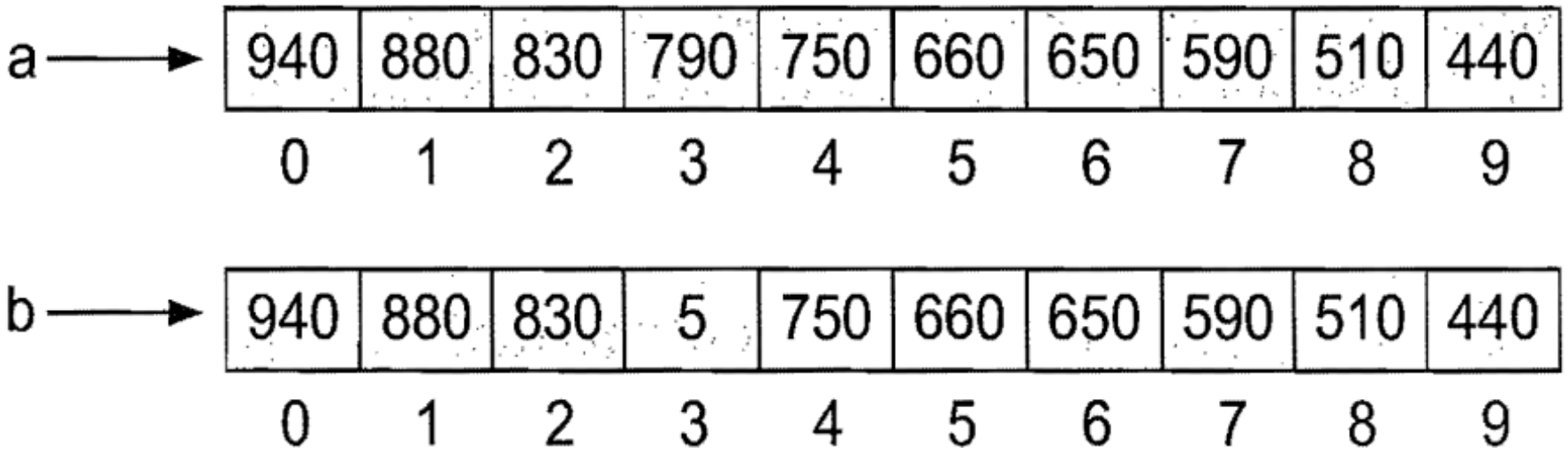
b=a; b[3]=5;



НИЗИ

- Клонирање на низи – генерирање на идентична копија
  - b = a.clone();

b[3]=5;



ФУНКЦИИ ЗА ИЗЛЕЗ

- Функции за печатење на екран:
  - System.out.print(); - курсорот останува во истиот ред
  - System.out.println(); - курсорот се придвижува во нов ред

```
System.out.print("Java  values:  ");
System.out.print(3.1415);
System.out.print(“ ,“);
System.out.print(15);
System.out.println("  (double, char, int) . ");
```

ФУНКЦИИ ЗА ВЛЕЗ

- Креирање на објект за читање вредности од тастатура:
  - new Scanner(System.in);

```
Scanner input = new Scanner(System.in);
```

- Методи за читање:

nextByte()	Чита byte цел број
nextShort()	Чита short цел број
nextInt()	Чита int цел број
nextLong()	Чита long цел број
nextFloat()	Чита float реален број
nextDouble()	Чита double реален број
next()	Чита стринг до првото празно место
nextLine()	Чита цела линија (стринг до првиот нов ред)

ФУНКЦИИ ЗА ВЛЕЗ

```
import java.util.Scanner;

public class InputExample {
    public static void main(String args[]) {
        Scanner s = new Scanner(System.in);
        System.out.print("Enter your age in years: “);
        double age = s.nextDouble();
        System.out.print("Enter your maximum heart rate: ");
        double rate = s.nextDouble();
        double fb = (rate - age) * 0.65;
        System.out.println(“Your target fat burning heart rate is " + fb + ". ");
} }
```

ФУНКЦИИ ЗА ВЛЕЗ

- BufferReader
  - Чита низа од знаци (од датотека, тастатура,...)

```
InputStreamReader isr = new InputStreamReader(System.in);
BufferedReader br = new BufferedReader(isr);
String line = "";
while ((line = br.readLine()) != null)
strLine += line + "~"; //edited isr.close();
```