

Tema: Ndërtimi i një makine llogaritëse
në Java

Përshëndetje të gjithëve!

Në këtë artikull do të shpjegojmë ndërtimin e një aplikacioni si **Makina Llogaritëse**, në Java.

Makina jonë llogaritëse do të kryejë katër veprimet kryesore; mbledhje, zbritje, shumëzim dhe pjesëtim. Për këtë krijojmë klasën `FunksionetLlogaritese` me katër metodat të cilat kryejnë këto veprime.

```
package CalculatorApp;

public class FunksionetLlogaritese {

    public static double mbledhje(double num1, double num2) {
        return num1 + num2;
    }

    public static double zbritje(double num1, double num2) {
        return num1 - num2;
    }

    public static double shumezim(double num1, double num2) {
        return num1 * num2;
    }

    public static double pjesetim(double num1, double num2) {
        return num1 / num2;
    }
}
```

Analizojmë një prej metodave të quajtur **mbledhje**.

```
public static double mbledhje(double num1, double num2) {
    return num1 + num2;
}
```

Kemi **access modifier public**, që përcakton se kjo metodë mund të aksesohet kudo në kodin tonë. Më pas kemi **keyword static**, e cila përdoret për një variabël ose një metodë që është e njëjtë për çdo instancë të një klase. Tre veçori të metodave statike janë:

- Një metodë statike i përket klasës dhe jo objektit të një klase.
- Një metodë statike mund të thirret pa pasur nevojë të krijojmë një instancë të një klase.
- Një metodë statike mund të ketë akses në *data members* statike dhe mund të ndryshojë vlerën e tyre.

Keyword double përcakton tipin e të dhënës të cilën metoda do të bëj **return**. Tipi i të dhënave që bëhen *return* nga një metodë duhet të jetë në përputhje me **return type** të specifikuar nga metoda. Për shembull, nëse *return type* i disa metodave është *boolean*, ne nuk mund të bëjmë *return* një *double*. **Return** përdoret për të dalë nga një metodë, me ose pa një vlerë. Me pak fjalë, për të përcaktuar se çfarë vlere do të japë kjo metodë pasi thirret. Emri i metodës është **mbledhje()** dhe merr dy parametra të tipit *double* (num1 dhe num2). Brenda bllokut të metodës kemi vetëm një *return statement* që përcakton se çfarë do të kthejë kjo metodë. Në rastin tonë kthejmë shumën e num1 + num2.

Më pas krijojmë klasën **MainClass{}**. Në këtë klasë kemi metodën *main* të cilën e kemi shpjeguar dhe më parë. Në metodën *main* kemi deklaruar katër variabla nga të cilat tre janë të tipit *double* (num1, num2, rezultati) dhe një e tipit *char* (operatori).

```
package CalculatorApp;

import java.util.Scanner;

public class MainClass {

    public static void main(String[] args) {

        double num1;
        double num2;
        char operatori;
        double rezultati = 0;

        Scanner sc = new Scanner(System.in);
        System.out.println("Zgjidh operatorin {+, -, *, /} \n");
        operatori = sc.next().charAt(0);

        System.out.println("Vendos numrin e pare: \n");
        num1 = sc.nextDouble();

        System.out.println("Vendos numrin e dyte: \n");
        num2 = sc.nextDouble();

        if(Character.compare(operatori, '/') == 0 && num2 == 0) {
            System.out.println("Numri i dyte nuk mund te jete 0. Vendos perseri: \n");
            num2 = sc.nextDouble();
        }

        switch (operatori) {
            case '+':
                rezultati = FunksionetLlogaritese.mbledhje(num1, num2);
                break;
            case '-':
                rezultati = FunksionetLlogaritese.zbritje(num1, num2);
                break;
            case '*':
                rezultati = FunksionetLlogaritese.shumezim(num1, num2);
                break;
            case '/':
                rezultati = FunksionetLlogaritese.pjesetim(num1, num2);
                break;
            default:
                System.out.println("Operatori i zgjedhur eshte i gabuar");
                break;
        }

        sc.close();
        System.out.println(num1 + " " + operatori + " " + num2 + " = " + rezultati);
    }
}
```



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

Në rreshtin më sipër krijojmë një objekt të klasës **Scanner**, e cila është e përcaktuar në paketën (package) **java.util.scanner**. Klasa Scanner na lejon të marrim inputs nga konsola (console). **System.in** e kalojmë si parametër për t'i treguar kompiluesit të Java se input-i i sistemit do të sigurohet përmes console (tastierës).




```
System.out.println("Zgjidh operatorin [+ , - , * , /] \n");  
operatori = sc.next().charAt(0);
```

Me anë të dy rreshtave të mësipërme të kodit si fillim printojmë në console tekstin “Zgjidh operatorin [+ , - , * , /]”.

Në rreshtin e dytë variablit operatori i japim vlerën e vendosur nga përdoruesi. Klasa Scanner ka disa metoda si **nextInt()**, **nextDouble()**, **nextBoolean()** etj. por nuk ka një metodë *nextChar()*. Për këtë arsye përdorim `sc.next().charAt(0)`; Funkzioni **next()** kthen si String token-in/fjalën e vendosur si input dhe funksioni **charAt(0)** kthen karakterin e parë në String.

Në rreshtin `num1 = sc.nextDouble()`; variablit **num1** i japim vlerën double të vendosur nga përdoruesi në *console* dhe e njëjta logjikë vlen dhe për **num2**.



```
if(Character.compare(operatori, '/') == 0 && num2 == 0) {  
    System.out.println("Numri i dyte nuk mund te jete 0. Vendos perseri: \n");  
    num2 = sc.nextDouble();  
}
```

Me anë të kësaj pjese kodi bëjmë kontrollin për veprimin e pjesëtimit ku emëruesi nuk duhet të jetë 0. **if()** është një **decision-making statement**. Përdoret për të vendosur nëse një pjesë kodi

duhet të ekzekutohet ose jo. Nëse kushti brenda kllapave () është **true** atëherë blloku i kodit ekzekutohet. Nëse kushti është **false**, blloku i kodit nuk ekzekutohet. Në rastin tone kemi dy kontrolle të lidhura me && (si e kemi shpjeguar më parë, duhet që të dyja të jenë true në mënyrë që të ekzekutohet blloku i kodit brenda kllapave { }).

`Character.compare(operatori, '/')` → metoda ***compare(char x, char y)*** e klasës ***Character***, përdoret për të krahasuar dy vlera *char* në mënyrë numerike.

Metoda ***compare(char x, char y)*** kthen:

- Vlerën 0 nëse $x == y$
- Vlerë më të vogël se 0 nëse $x < y$
- Vlerë më të madhe se 0 nëse $x > y$

Në rastin tone duam të kontrollojmë nëse operatori i zgjedhur prej përdoruesit është '/' (pjesëtimi) prandaj krahasojmë vlerën e variablit *operatori* me karakterin '/

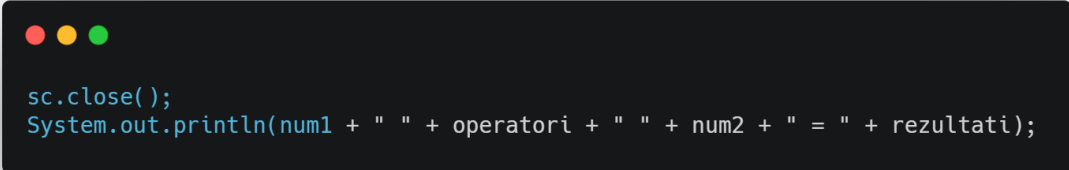
`Character.compare(operatori, '/') == 0`, pra nëse vlera e kthyer nga funksioni `compare` i klasës *Character* është e barabartë me 0 (ka vlerën 0) do të thotë se përdoruesi ka zgjedhur operatorin e pjesëtimit. Më pas kontrollojmë nëse *num2* (numri i dytë i vendosur prej përdoruesit) është 0. Nëse plotësohen të dyja këto kushte (janë true) ekzekutohet pjesa e kodit brenda kllapave {}. Si fillim printojmë tekstin "Numri i dyte nuk mund te jete 0. Vendos perseri:", dhe më pas na jepet përsëri mundësia për të vendosur numrin e dytë.

Për të vendosur se cila metodë do të ekzekutohet nga klasa *FunksionetLlogaritëse* përdorim `switch` statement. ***switch (operatori)*** -> këtu vendosim variablin *operatori* vlerën e të cilit e marrim si *input* nga përdoruesi, dhe kontrollojmë nëse vlera e variablit ***operatori*** është e njëjtë me një nga vlerat e *case*. Nëse ndodh kështu atëherë do të ekzekutohet ajo pjesë kodi brenda *case* që përshtatet. Për shembull; nëse përdoruesi zgjedh veprimin e shumëzimit, operatori do të ketë vlerën '*'. Kjo vlerë kontrollohet së pari me *case '+'*. Meqë nuk janë të njëjta kalojmë në rastin e dytë, *case '-'* e kështu me radhë derisa gjendet, në rastin tonë është *case '*'*. Ekzekutohet kodi brenda *case '*'* ku thirret

```
switch (operatori) {
    case '+':
        rezultati = FunksionetLlogaritese.mbledhje(num1, num2);
        break;
    case '-':
        rezultati = FunksionetLlogaritese.zbritje(num1, num2);
        break;
    case '*':
        rezultati = FunksionetLlogaritese.shumezim(num1, num2);
        break;
    case '/':
        rezultati = FunksionetLlogaritese.pjesetim(num1, num2);
        break;
    default:
        System.out.println("Operatori i zgjedhur eshte i gabuar");
        break;
}
```

metoda ***shumezim()*** nga klasa *FunksionetLlogaritëse*. Siç e shihni për të thirrur këtë metodë, meqë është *statike* nuk na u desh të krijojmë një objekt të klasës *FunksionetLlogaritëse*, por e thirrëm direct duke përdorur ***emriKlasës.metoda()***. I kalojmë si parametra variablat ***num1*** dhe ***num2*** dhe na kthen si vlerë shumëzimin e tyre që i vendoset variablit rezultati. ***break; statement*** përdoret

për të përfunduar një *loop*. Në rastin tonë duke qenë se gjetëm case-in e duhur ndërpritet kërkimi dhe kalohet në rreshtat e kodit jashtë **switch statement**. *Switch statement* ka dhe një *case* të quajtur **default** e cila ekzekutohet nëse asnjë case tjetër nuk përket me vlerën e variablit të vendosur në fillim. Pra, nëse përdoruesit do kishte zgjedhur si input për operatorin ‘%’ do të kalonim në *default case* dhe do të printohej në console teksti “Operatori i zgjedhur eshte i gabuar”, dhe më pas do të ekzekutohej **break; statement**.



```
sc.close();
System.out.println(num1 + " " + operator + " " + num2 + " = " + rezultati);
```

Në dy rreshtat e fundit të metodës main kemi `sc.close()` e cila përdoret për të mbyllur scanner-in dhe gjithashtu printojmë tekstin “ **vlera e num1 operatori vlera e num2 = vlera e rezultatit**”.

Përfundim

Në këtë artikull mësuam se si të ndërtojmë një makinë llogaritëse me katër veprime matematikore. Për ta zhvilluar këtë ndërtoam dy klasa *FunksionetLlogratiese* dhe *MainClass*.

Nëse doni ta zhvilloni më tepër këtë aplikacion mund të ndërtoni një ndërfaqe grafike duke përdorur JavaFX dhe mund të ndryshoni kodin në mënyrë që të kryhen veprime me më tepër se dy numra.