

Objektno orijentirano programiranje

1. Programski jezik Java

Instalacija, kompilacija i izvršavanje jednostavnih programa,
organizacija datoteka

Creative Commons

You are free to

- **Share** — copy and redistribute the material in any medium or format
- **Adapt** — remix, transform, and build upon the material

under the following terms

- **Attribution** — You must give appropriate credit, provide a link to the license, and indicate if changes were made. You may do so in any reasonable manner, but not in any way that suggests the licensor endorses you or your use.
- **NonCommercial** — You may not use the material for commercial purposes.
- **ShareAlike** — If you remix, transform, or build upon the material, you must distribute your contributions under the same license as the original.
- <https://creativecommons.org/licenses/by-nc-sa/4.0/>



Java Development Kit (JDK) ≥ 17

- Skinuti i raspakirati odgovarajuću arhivu s <http://jdk.java.net> te podesiti putanju (detaljnije u dodatku ovim predavanjima)

- Alternativno na Ubuntu ≥ 18.04 i derivatima

```
sudo apt install default-jdk
```

- Instalaciju provjeriti pokretanjem

- `java -version`
- `javac -version`

```
mario@Asgard ~$ java -version
openjdk version "17.0.2" 2022-01-18
OpenJDK Runtime Environment Temurin-17.0.2+8 (build 17.0.2+8)
OpenJDK 64-Bit Server VM Temurin-17.0.2+8 (build 17.0.2+8)

mario@Asgard ~$ javac -version
javac 17.0.2

mario@Asgard ~$
```

jdk.java.net

GA Releases
JDK 17
JMC 8

Early-Access
Releases
JDK 19
JDK 18
Loom
Metropolis
Panama
Valhalla

Reference
Implementations
Java SE 17
Java SE 16
Java SE 15
Java SE 14
Java SE 13
Java SE 12
Java SE 11
Java SE 10
Java SE 9
Java SE 8
Java SE 7
Feedback
Report a bug
Archive

OpenJDK JDK 17.0.2 General-Availability Release

This page provides production-ready open-source builds of the [Java version 17](#), an implementation of the [Java SE 17 Platform](#) under the [Public License, version 2](#), with the [Classpath Exception](#).

Commercial builds of JDK 17.0.2 from Oracle, under a [non-open-source license](#), can be found at the [Oracle Technology Network](#).

Documentation

- [Features](#)
- [Release notes](#)
- [API Javadoc](#)

Buils

Linux/AArch64	tar.gz (sha256)	185776417 bytes
Linux/x64	tar.gz (sha256)	187033144
macOS/AArch64	tar.gz (sha256)	182209404
macOS/x64	tar.gz (sha256)	184480668
Windows/x64	zip (sha256)	186216309

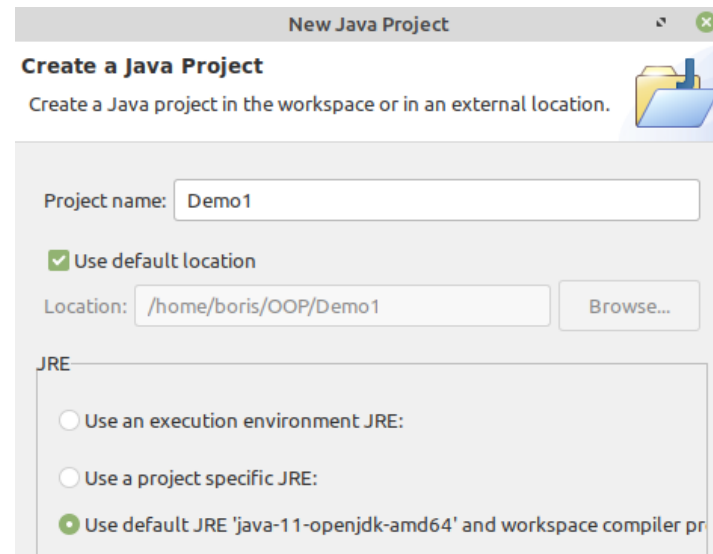
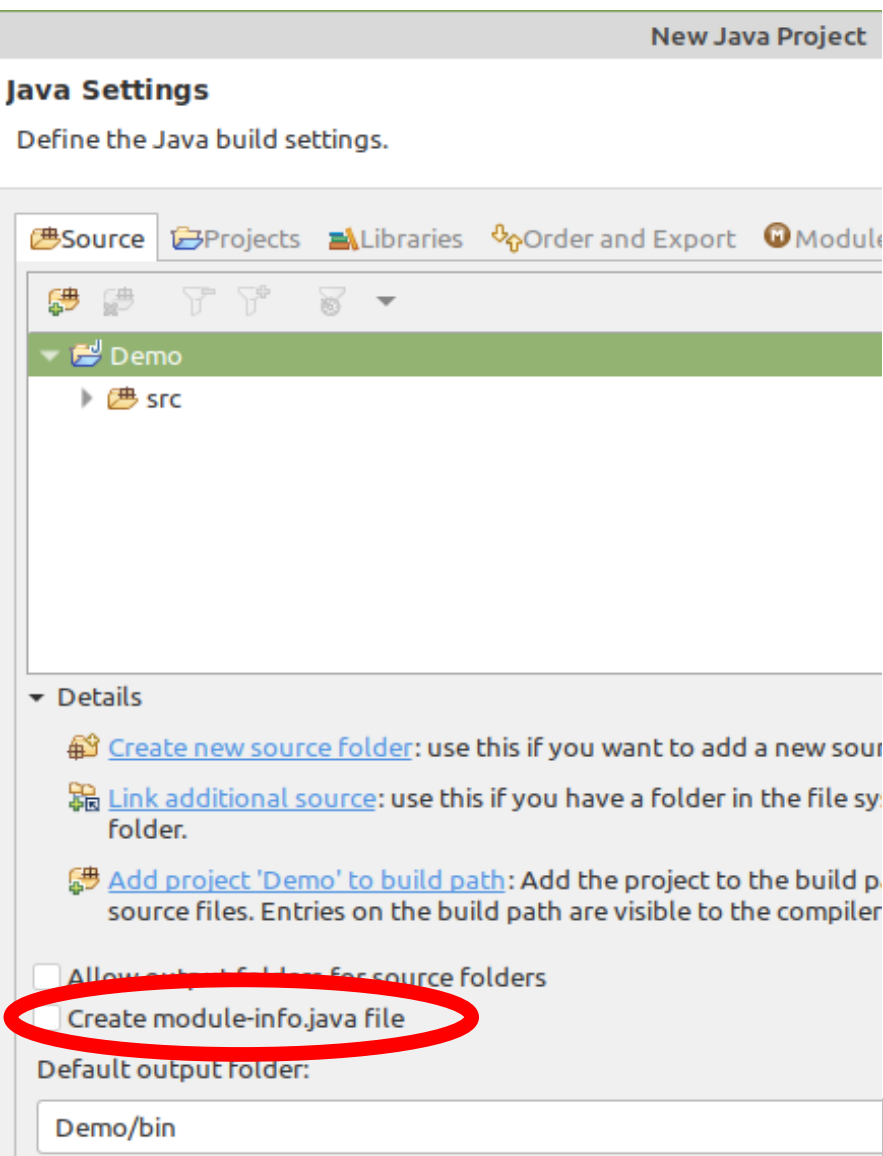
- Dokumentacija: <https://docs.oracle.com/en/java/javase/17>

Uobičajena razvojna okruženja (1)

- Eclipse IDE, Apache NetBeans, IntelliJ IDEA, Visual Studio Code, ...
 - Preporuča se korištenje IDE-a nakon uvodnih primjera
- Eclipse IDE (2021-12 R):
<https://www.eclipse.org/downloads/packages/>
 - Odabrati **Eclipse IDE for Java Developers**, skinuti i raspakirati arhivu



Napomena vezana za Eclipse IDE



Prilikom kreiranja novog projekta, potrebno je isključiti opciju `Create module-info.java`

- Moduli su novost od Java 9, ali ih ne koristimo na ovom predmetu

Uobičajena razvojna okruženja (2)

- Apache NetBeans (trenutno 12.6):
<https://netbeans.apache.org/download/index.html>
- Pretpostavka za instalaciju
 - Instaliran JDK
 - Podešena putanja (tj. sistemska varijabla PATH) za JDK
 - Podešena sistemska varijabla JAVA_HOME
- Koraci instalacije
 - Dohvatiti izvršnu verziju instalacije s gornje poveznice
 - Pokrenuti instalaciju
 - Odabrati sve ponuđene komponente osim PHP-a (Base IDE, Java SE, JAVA EE i HTML5/JavaScript)

Prvi program u Javi – Hello World

- Stvoriti novu datoteku i snimiti je kao *HelloWord.java*
 - Na Windowsima obratiti pažnju na skrivanje ekstenzija!

```
public class HelloWorld {                                     HelloWorld.java
    public static void main(String[] args){
        System.out.println("Hello world");
    }
}
```

- Dva osnovna pravila:
 1. Ime (klase) iza ključnih riječi *public* i *class* mora odgovarati imenu datoteke ispred ekstenzije *.java*
 2. U datoteci se smije nalaziti samo jedan zapis oblika *public class Ime*
 - direktna posljedica pravila #1
- O ostalim pravilima i mogućim kombinacijama naknadno

Izvršavanje programa – generalni koncepti

- Programer piše izvorni kod – procesor izvršava strojni (binarni) kod
 - Klasični pristup (npr. C)
 - (pretprocesirani) izvorni kod → (prevodilac prevodi u) simbolički strojni kod → (kojeg assembler pretvara u) objektni kod → (a povezičavač povezuje s objektnim i strojnim kodom programskih biblioteka u) → izvršni kod
 - Izvorni kod je (možda) prenosiv, ostali dijelovi su specifični za platformu
 - Python, Perl, MathLab, ...
 - interpreter za pojedini operacijski sustav interpretira (i/ili pretvara u efikasnije) instrukcije iz izvornog koda i izvršava ih
 - Izvorni kod je (vjerojatno) portabilan
 - Java, C#
 - izvorni kod → (prevodilac prevodi u) *bytecode/intermediate language* (binarni kod s instrukcijama za pojedino virtualno računalo) → *Just-In-Time* prevodilac pretvara *bytecode* u strojni kod i izvršava ga
 - Bytecode/IL je portabilan (prenosiv)

Prvi program u Javi

- Izvršni kod iz *.java* datoteka prevodi se Javinim prevodiocom - *javac*
- Prevođenjem nastaje jedna ili više datoteka s ekstenzijom *.class*
 - nisu izvršne datoteke
 - sadrže *bytecode* pisan za Javino virtualno računalo
 - sinonim: *intermediate language*
- Datoteke su portabilne
 - mogu se kopirati i izvršiti na Linuxu, Windowsima, macOS-u ili bilo kojem OS-u koje ima instaliran Java Runtime Environment (JRE)
- *java* izvršava *bytecode* iz *.class* datoteka
 - započinje se s kodom koji se nalazi u metodi *main* navedene datoteke
 - *.class* se ne navodi u naredbi za izvršavanje

/home/boris/temp/intro

 HelloWorld.class

 HelloWorld.java

```
boris@C55:~/temp/intro$ javac HelloWorld.java
boris@C55:~/temp/intro$ java HelloWorld
Hello world
```

Paketi

- Preporuča se odvojiti izvorni kod od prevedenog (byte) koda
 - obično u mapama *src* i *bin*
- Srodne klase grupiraju se u pakete
 - jednostavnije održavanje i pretraga
 - rješava potencijalne probleme istog naziva datoteka
 - npr. što ako postoji više datoteka, tj. klasa imena *HelloWorld*
 - **puno ime sastoji se od naziva paketa i naziva klase**
- Konvencija imenovanja paketa
 - koristiti mala slova
 - naziv paketa obično kombinacija obrnute internetske domene neke firme/institucije (npr. *hr.fer*) i naziva softvera
 - U primjerima koji slijede koristit će se *hr.fer.oop.tema_predavanja*

Organizacija mapa prilikom korištenja paketa

- Odabrati vršnu mapu po želji i stvoriti podmape *src* i *bin*
- Za svaki dio naziva paketa stvoriti odgovarajuću mapu



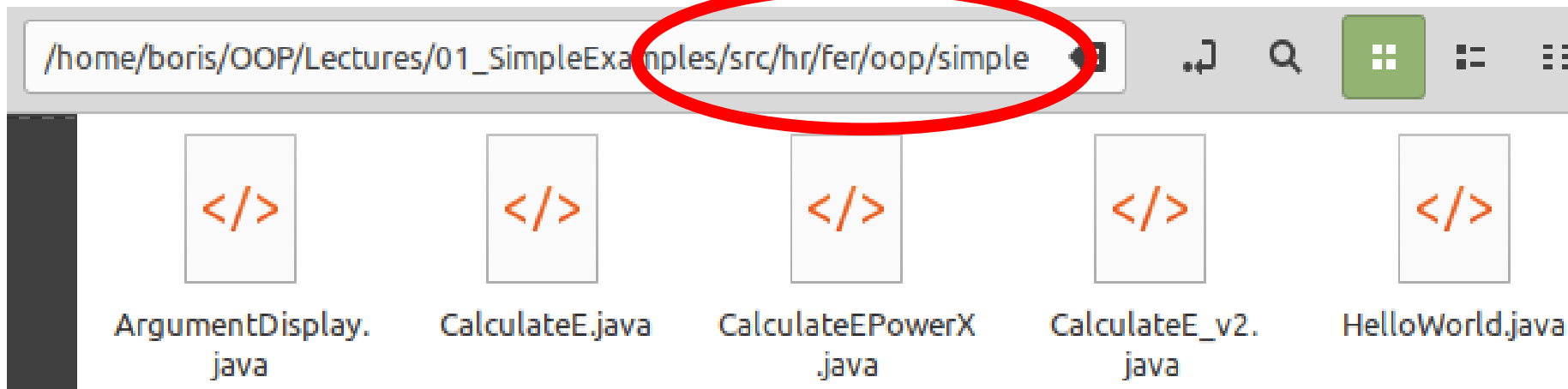
- Prevodilac pokrenuti s opcijom `-d bin`
- Uspješno prevođenje stvorit će identičnu strukturu ispod mape `bin` s `.class` datotekama

Hello World – varijanta korištenjem paketa

- Koristi se ključna riječ *package* na početku datoteke s izvornim kodom

```
package hr.fer.oop.simple;  
public class HelloWorld {  
    public static void main(String[] args){  
        System.out.println("Hello World");  
    }  
}
```

...01_SimpleExamples/src/fer/oop/simple/HelloWorld.java



Prevođenje korištenjem mapa *src* i *bin*

- U vršnoj mapi pojedinog projekta izvršiti sljedeću naredbu

```
javac -d bin  
src/hr/fer/oop/simple/HelloWorld.java
```

- Parametrom *-d bin* određuje se odredište prevedenih datoteka
- Uspješnim prevođenjem stvara se *HelloWorld.class* u mapi *...bin/hr/fer/oop/simple*
- Pri pokretanju parametrom *-cp* se navodi **vršna lokacija** prevedenih datoteka i **puni naziv** klase koja se želi pokrenuti

```
java -cp bin hr.fer.oop.simple.HelloWorld
```

 - Primijetiti da se ne navodi putanja do konkretne *.class* datoteke!

```
boris@C55:~/OOP/Lectures/01_SimpleExamples$ javac -d bin src/hr/fer/oop/simple/HelloWorld.java  
boris@C55:~/OOP/Lectures/01_SimpleExamples$ java -cp bin hr.fer.oop.simple.HelloWorld  
Hello world!
```

Osnove Javine sintakse

- Javina sintaksa bazirana na C-u:
 - Definicija/deklaracija varijabli
 - slični osnovni (primitivni tipovi)
 - *statically-typed* : sve varijable moraju se deklarirati prije korištenja
 - Pravila imenovanja varijabli slično kao u C-u
 - Blokovi omeđeni vitičastim zagradama
 - Petlje (*for*, *while*, *do-while*) i grananja (*if-else*, *switch*)
 - razlika su logički izrazi: umjesto 0 za laž i različito od nule za istinu, postoji poseban logički tip *boolean*
 - Sintaksa i definicija funkcija
 - u Javi se obično koristi pojam *metoda* umjesto *funkcija*
- Više o osnovama Java
<https://docs.oracle.com/javase/tutorial/java/nutsandbolts/index.html>

Primitivni (osnovni) tipovi u Javi

Tip	Veličina u oktetima	Pretpostavljena vrijednost	Raspon
byte	1	0	[-128, 127]
short	2	0	[-32768, 32767]
int	4	0	[-2 147 483 648, 2 147 483 647]
long	8	0L	[-9 223 372 036 854 775 808, 9 223 372 036 854 775 807]
char	2	'\u0000'	[0, 65 536] (UTF-16)
boolean	?	false	<i>true</i> ili <i>false</i>
float	4	0.0f	$\approx \pm 3.40282347E+38F$ standard IEEE 754
double	8	0.0d	$\approx \pm 1.79769313486231570E+308$ standard IEEE 754

Primjer: Izračun Eulerova broja e

$$e = 2.7182818284590452353602874713527...$$

- Može se aproksimirati računanjem sume prvih n članova Taylorovog reda

$$e^x = \sum_{i=0}^{\infty} \frac{x^i}{i!}$$

za $x = 1$

Primjer: Izračun Eulerova broja e

- Izračun napisan u posebnoj metodi

$$e = \sum_{i=0}^{\infty} \frac{1}{i!}$$

```
package hr.fer.oop.simple;
public class CalculateE {
    public static void main(String[] args) {
        double sum = ePowerX(1);
        System.out.printf("e = %.6f\n", sum);
    }
    public static double ePowerX(double x) {
        double power = 1.0;    double factorial = 1.0;
        double sum = 1.0;
        for(int i = 1; i < 10; i++) {
            power = power * x;
            factorial = factorial * i;
            sum += power/factorial;
        }
        return sum;
    }
}
```

...01_SimpleExamples/src/hr/fer/oop/simple/CalculateE.java

Odvajanje dijelova koda u zasebne datoteke

- Metoda *ePowerX* bi mogla zatrebati za neke buduće programe
 - Premještena u novu datoteku u paketu *hr.fer.oop.util*

```
package hr.fer.oop.util;
public class Taylor {
    public static double ePowerX(double x) {
        double power = 1.0;
        double factorial = 1.0;
        double sum = 1.0;
        for(int i = 1; i < 10; i++) {
            power = power * x;
            factorial = factorial * i;
            sum += power/factorial;
        }
        return sum;
    }
}
```

...01_SimpleExamples/src/hr/fer/oop/util/Taylor.java

Što se događa s prethodno napisanim programom?

- Metoda *ePowerX* nije više u istoj datoteci
 - Pripada klasi koja se zove *Taylor*
 - Klasa *Taylor* ne pripada istom paketu
 - Potrebno prevodiocu reći da je želimo uključiti (*import*)
 - Isti princip za matematičke funkcije i konstante. One pripadaju klasi *Math* (paket *java.lang* koji nije potrebno eksplicitno uključiti)

```
package hr.fer.oop.simple;
import hr.fer.oop.util.Taylor;
public class CalculateE_v2 {
    public static void main(String[] args) {
        double e = Taylor.ePowerX(1);
        System.out.printf("e = %.6f\n", e);
        double diff = Math.abs(Math.E - e);
        System.out.printf("diff = %g\n", diff);
    }
}
```

...01_SimpleExamples/src/hr/fer/oop/simple/CalculateE_v2.java

Prevođenje izvornog koda iz više datoteka (1)

- Naredba

```
javac -d bin  
src/hr/fer/oop/simple/CalculateE_v2.java  
uzrokuje pogrešku
```

```
boris@C55:~/00P/Lectures/01_SimpleExamples$ javac -d bin src/hr/fer/oop/simple/CalculateE_v2.java  
src/hr/fer/oop/simple/CalculateE_v2.java:2: error: package hr.fer.oop.util does not exist  
import hr.fer.oop.util.Taylor;  
                        ^  
src/hr/fer/oop/simple/CalculateE_v2.java:6: error: cannot find symbol  
        double e = Taylor.ePowerX(1);  
                        ^  
symbol:   variable Taylor  
location: class CalculateE_v2  
2 errors
```

Prevođenje izvornog koda iz više datoteka (2)

- *Taylor.java* je u drugoj mapi (paketu), pa je potrebno prilikom prevođenja navesti gdje će prevodilac tražiti klase navedene u import naredbama
- Koristi se parametar *sourcepath*

```
javac -sourcepath src -d bin  
src/hr/fer/oop/simple/CalculateE_v2.java
```
- Primijetiti da nismo eksplicitno navodili putanju do datoteke *Taylor.java*, već se ona mora nalaziti na odgovarajućem mjestu ispod mape *src*
import hr.fer.oop.util.Taylor → *src/hr/fer/oop/util/Taylor.java*
- Napomena: pokretanje programa se ne mijenja

```
java -cp bin hr.fer.oop.simple.CalculateE_v2
```

Komentiranje i dokumentiranje koda

- Komentiranje koda se radi iz dva razloga (i na dva načina)
 - „Obični” komentari olakšavaju razumijevanje onome tko naknadno čita program
 - `/* komentar u više redaka */`
 - `// komentar do kraja retka`
 - Komentari iz kojih se generira dokumentacija (**javadoc komentari**) za klase i metoda
 - *`/** komentar s posebnim oznakama */`*
- **Javadoc komentari** sadrže oznake oblika `@naziv vrijednost`, npr.
 - `@author ime_autora`, npr. `@author OOP`
 - `@version verzija_razreda`, npr. `@version 1.0`
 - `@param ime_argumenta opis`, npr. `@param x broj čiji sinus treba izračunati`
 - `@return opis`, npr. `@return vraća sinus zadanog broja`

Primjer izrade Javine dokumentacije

```
package hr.fer.oop.util;
```

...01_SimpleExamples/src/hr/fer/oop/util/Taylor.java

```
public class Taylor {
```

```
    /**
```

```
     * Calculates  $e^x$  for Taylor series, according to formula:
```

```
     *  $e^x = 1 + x + (x^2/(2!)) + (x^3/(3!)) + (x^4/(4!)) + \dots$ 
```

```
     * @param x argument of function  $e^x$ 
```

```
     * @return  $e^x$  calculated as sum of first 10 numbers in Taylor series.
```

```
    */
```

```
    public static double ePowerX(double x) {
```

```
        double sum = 0.0;
```

```
        ...
```

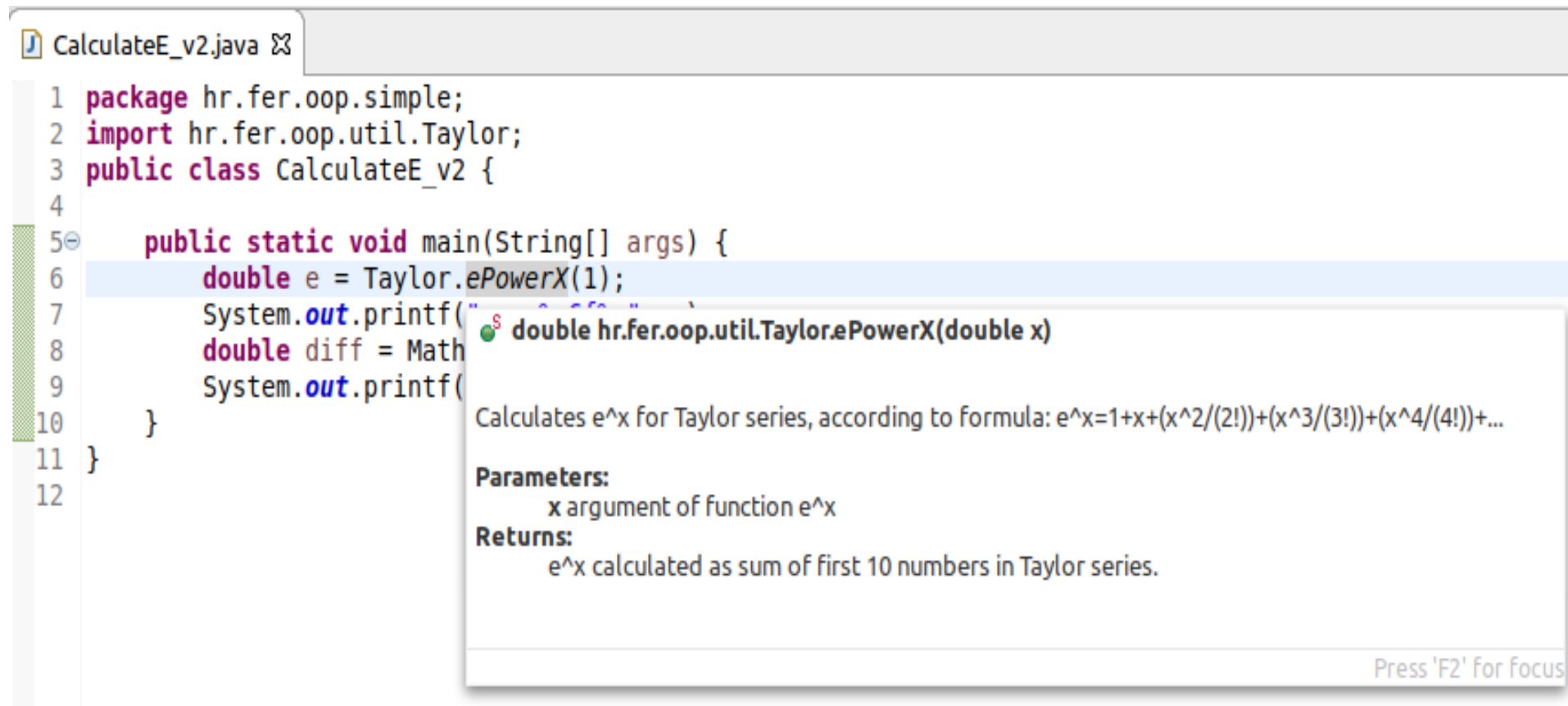
■ Naredba

```
javadoc -sourcepath src hr.fer.oop.util -d docs
```

stvara HTML datoteke za Javinu dokumentaciju klasa iz navedenog paketa

JavaDoc i IDE

- Ako klasa ima *JavaDoc* komentare, bit će prikazani unutar IDE-a na isti način kao za ugrađene Javine klase.
 - Primjer za Eclipse



The screenshot shows the Eclipse IDE with a Java file named `CalculateE_v2.java` open. The code defines a package `hr.fer.oop.simple`, imports `hr.fer.oop.util.Taylor`, and defines a public class `CalculateE_v2`. Inside the class, there is a `main` method that calls `Taylor.ePowerX(1)`. A tooltip is displayed over the `ePowerX` call, showing the signature `double hr.fer.oop.util.Taylor.ePowerX(double x)` and its JavaDoc comment: "Calculates e^x for Taylor series, according to formula: e^x=1+x+(x^2/(2!))+(x^3/(3!))+(x^4/(4!))+...". The tooltip also lists the parameter `x` as the argument of the function and the return value as `e^x` calculated as the sum of the first 10 numbers in the Taylor series.

```
1 package hr.fer.oop.simple;
2 import hr.fer.oop.util.Taylor;
3 public class CalculateE_v2 {
4
5     public static void main(String[] args) {
6         double e = Taylor.ePowerX(1);
7         System.out.printf("e = %f\n", e);
8         double diff = Math.exp(1) - e;
9         System.out.printf("diff = %f\n", diff);
10    }
11 }
12
```

double hr.fer.oop.util.Taylor.ePowerX(double x)

Calculates e^x for Taylor series, according to formula: e^x=1+x+(x²/(2!))+(x³/(3!))+(x⁴/(4!))+...

Parameters:
x argument of function e^x

Returns:
e^x calculated as sum of first 10 numbers in Taylor series.

Press 'F2' for focus

Argumenti programa

- Argumenti spremljeni u nizu stringova
 - String* je (općenito, ali neprecizno) niz znakova
 - Indeksi niza od 0 do broj_elemenata-1
 - Broj elemenata nekog niza može se dobiti s *imeniza.length*

```
package hr.fer.oop.simple;    ...01_SimpleExamples/src/.../simple/ArgumentDisplay.java
public class ArgumentDisplay {
    public static void main(String[] args) {
        int argCount = args.length;
        for(int i = 0; i < argCount; i++) {
            System.out.printf("Argument[%d] = %s%n", i, args[i]);
        }
    }
}
```

```
Argument[0] = first
Argument[1] = second
Argument[2] = this is the third
Argument[3] = and then something more
Argument[4] = and
Argument[5] = more
```

```
java -cp bin hr.fer.oop.simple.ArgumentDisplay first
second "this is the third" "and then something more"
"and" "more"
```

Konverzija *Stringa* u numerički tip

- Ako neki *String* sadrži samo znamenke i decimalnu točku, onda se odgovarajućim metodama iz njega može „izvući” broj i pohraniti u varijablu odgovarajućeg numeričkog tipa
 - Tipični primjeri

```
int x = Integer.parseInt("123") – x je int s vrijednošću 123
Double.parseDouble("3.14") – vraća double s vrijednošću 3.14
```
 - Neuspješno „parsiranje” uzrokuje pogrešku
 - Preciznije, pojavljuje se iznimka prilikom izvršavanja koja prekida normalno izvršavanje programa
 - O iznimkama naknadno u predavanju broj 7
 - Primjer: `Integer.parseInt("12w")` će uzrokovati prekid (normalnog) izvršavanja programa
- Parsiranje ne mijenja sadržaj *Stringa*!
- Pogledajte opise metoda na <https://docs.oracle.com/en/java/javase/17> pod [API Documentation](#)

Argument programa za vrijednost x u e^x

```
package hr.fer.oop.simple;
import hr.fer.oop.Taylor;
public class CalculateEPowerX {
    public static void main(String[] args) {
        if (args.length != 1) {
            System.out.println(
                "The program needs an integer value x to calculate e^x.");
            System.exit(1); //exit program with error code 1
        }
        int x = Integer.parseInt(args[0]);
        double result = Taylor.ePowerX(x);
        System.out.printf("e^%d = %.6f%n", x, result);
        double diff = Math.abs(Math.pow(Math.E, x) - result);
        System.out.printf("diff = %g%n", diff);
    }
}
```

...01_SimpleExamples/src/hr/fer/oop/simple/CalculateEPowerX.java