

JAVA TUTORIAL

GENERAL

Java is a high-level programming language originally developed by Sun Microsystems and released in 1995. Java runs on a variety of platforms, such as Windows, Mac OS, and the various versions of UNIX. This tutorial gives a complete understanding of Java.

This lecture will take you through simple and practical approach while learning Java Programming language.

AUDIENCE

This lecture has been prepared for the beginners to help them understand the basic to advanced concepts related to Java Programming language.

PREREQUISITIES

Before you start doing practice with various types of examples given in this reference, I'm making an assumption that you are already aware about what is a computer program and what is a computer programming language.

COMPILE/EXECUTE JAVA PROGRAMS

For most of the examples given in this tutorial you will find Try it option, so just make use of it and enjoy your learning.

Try following example using **Try it** or **Example** option available at the top right corner of the below sample code box:

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

RESOURCES

Java tutorials by Oracle:

<http://www.oracle.com/technetwork/java/download-138590.html>

Top free Java ebooks:

<http://www.mkymong.com/featured/top-5-free-java-ebooks/>

Java Examples:

<http://www.java-examples.com/>

JAVA – OVERVIEW

Java programming language was originally developed by Sun Microsystems which was initiated by James Gosling and released in 1995 as core component of Sun Microsystems' Java platform (Java 1.0 [J2SE]).

As of December 2008, the latest release of the Java Standard Edition is 6 (J2SE). With the advancement of Java and its widespread popularity, multiple configurations were built to suite various types of platforms. Ex: J2EE for Enterprise Applications, J2ME for Mobile Applications.

Sun Microsystems has renamed the new J2 versions as Java SE, Java EE and Java ME respectively. Java is guaranteed to be Write Once, Run Anywhere.

Java is:

OBJECT ORIENTED

In Java, everything is an Object. Java can be easily extended since it is based on the Object model.

PLATFORM INDEPENDENT

Unlike many other programming languages including C and C++, when Java is compiled, it is not compiled into platform specific machine, rather into platform independent byte code. This byte code is distributed over the web and interpreted by virtual Machine (JVM) on whichever platform it is being run.

SIMPLE

Java is designed to be easy to learn. If you understand the basic concept of OOP Java would be easy to master.

SECURE

With Java's secure feature it enables to develop virus-free, tamper-free systems. Authentication techniques are based on public-key encryption.

ARCHITECTURAL-NEUTRAL

Java compiler generates an architecture-neutral object file format which makes the compiled code to be executable on many processors, with the presence of Java runtime system.

PORTABLE

Being architectural-neutral and having no implementation dependent aspects of the specification makes Java portable. Compiler in Java is written in ANSI C with a clean portability boundary which is a POSIX subset.

ROBUST

Java makes an effort to eliminate error prone situations by emphasizing mainly on compile time error checking and runtime checking.

MULTITHREADED

With Java's multithreaded feature it is possible to write programs that can do many tasks simultaneously. This design feature allows developers to construct smoothly running interactive applications.

INTERPRETED

Java byte code is translated on the fly to native machine instructions and is not stored anywhere. The development process is more rapid and analytical since the linking is an incremental and light weight process.

HIGH PERFORMANCE

With the use of Just-In-Time compilers, Java enables high performance.

DISTRIBUTED

Java is designed for the distributed environment of the internet.

DYNAMIC

Java is considered to be more dynamic than C or C++ since it is designed to adapt to an evolving environment. Java programs can carry extensive amount of run-time information that can be used to verify and resolve accesses to objects on run-time.

HISTORY OF JAVA:

James Gosling initiated the Java language project in June 1991 for use in one of his many set-top box projects. The language, initially called Oak after an oak tree that stood outside Gosling's office, also went by the name Green and ended up later being renamed as Java, from a list of random words.

Sun released the first public implementation as Java 1.0 in 1995. It promised Write Once, Run Anywhere (WORA), providing no-cost run-times on popular platforms.

On 13 November 2006, Sun released much of Java as free and open source software under the terms of the GNU General Public License (GPL).

On 8 May 2007, Sun finished the process, making all of Java's core code free and open-source, aside from a small portion of code to which Sun did not hold the copyright.

TOOLS YOU WILL NEED:

HARDWARE & SOFTWARE

For performing the examples discussed in this tutorial, you will need a Pentium 200-MHz computer with a minimum of 64 MB of RAM (128 MB of RAM recommended).

Windows 95/98/2000/XP operating system.

DEVELOPMENT TOOLS – MINIMUM 😊

- Java JDK 5
- Microsoft Notepad or any other text editor

DEVELOPMENT TOOLS WE WILL USE

NETBEANS 7.4

For tutorials and for Exercises we will use NetBeans, version 7.4.

NetBeans is an advanced, full-featured programming environment. It supports not only JAVA, but also C, C++, Assembler, Fortran, PHP and other programming languages.

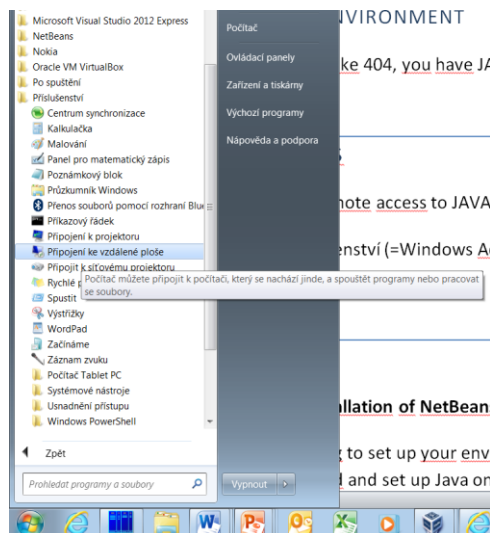
LABORATORY ENVIRONMENT

In our laboratories like 404, you have JAVA environment preinstalled. Please, check your desktop, where you find NetBeans icon.

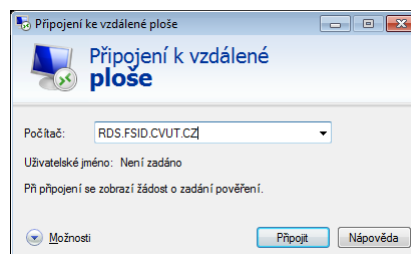
REMOTE ACCESS

You also can use remote access to JAVA environment.

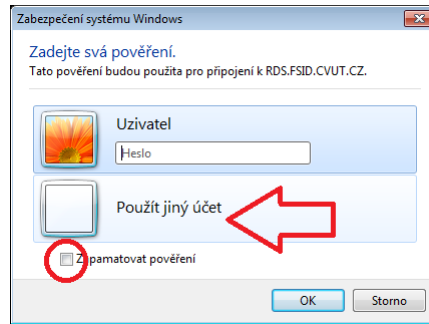
Run Start → Příslušenství (=Windows Accessories) → Připojení ke vzdálené ploše (=Remote Desktop System)



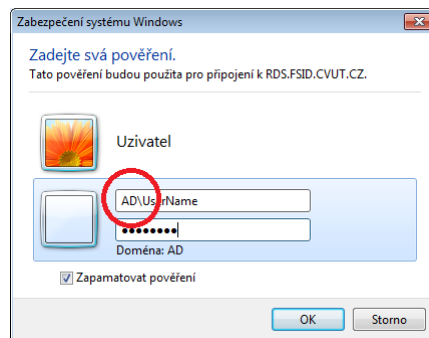
and you will get login window. Please fill „RDS.FSID.CVUT.CZ“ and press „Připojit“.



On the next window, select „Použit jiný účet“ (=select another account) and check „Zapamatovat pověření“ (=remember login).



For login, write „AD\“ followed by your login name and password, like in the following example:



Now, you should see standard desktop with NetBeans icon.

INSTALATION

Unfortunately, installation of NetBeans is not simple. It is quite a time-consuming job.

If you are still willing to set up your environment for Java programming language, then this section guides you on how to download and set up Java on your machine. Please follow the following steps to set up the environment.

Java SE is freely available from this [link](#). So you download a version based on your operating system.

Follow the instructions to download Java and run the .exe to install Java on your machine. Once you installed Java on your machine, you would need to set environment variables to point to correct installation directories:

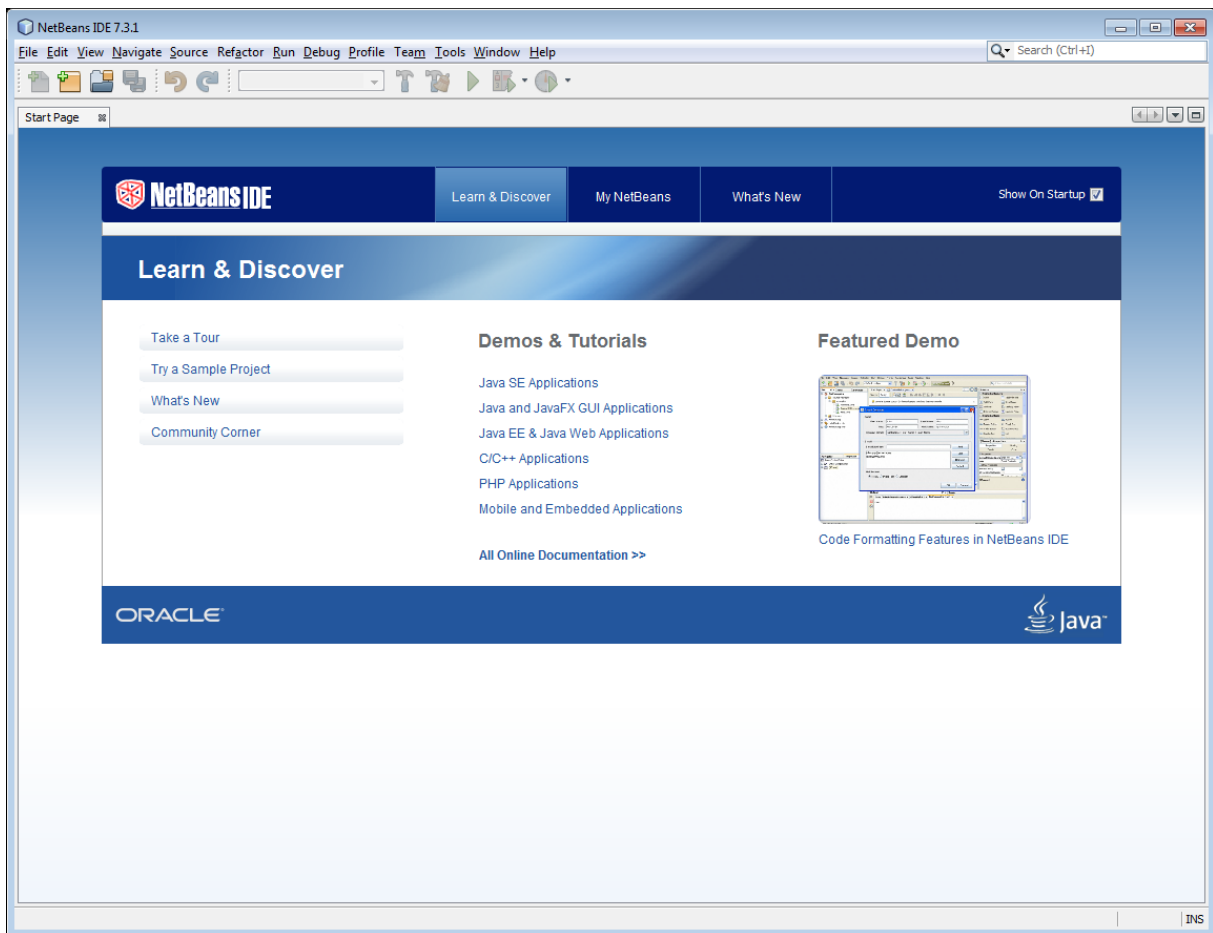
- Assuming you have installed Java in c:\Program Files\java\jdk directory, right-click on 'My Computer' and select 'Properties'.
- Click on the 'Environment variables' button under the 'Advanced' tab.
- Now, alter the 'Path' variable so that it also contains the path to the Java executable. Example, if the path is currently set to 'C:\WINDOWS\SYSTEM32', then change your path to read 'C:\WINDOWS\SYSTEM32;c:\Program Files\java\jdk\bin'.

Then load and run NetBeans installation program from [this link](#). Use „Java SE“ or „All“ version. Run installation .EXE file and follow instructions.

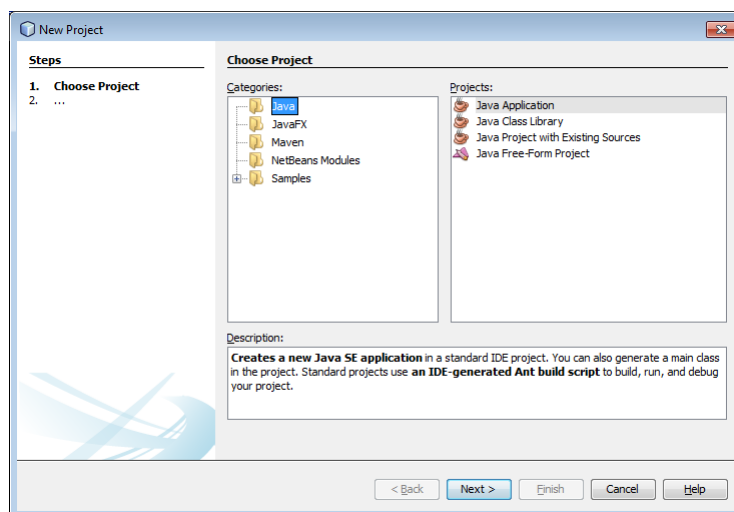
FIRST START OF NETBEANS

FIRST STEPS

When you first run NetBeans, you will see NetBeans introductory page like this:



Click File → New Project and select Java, Java Application:



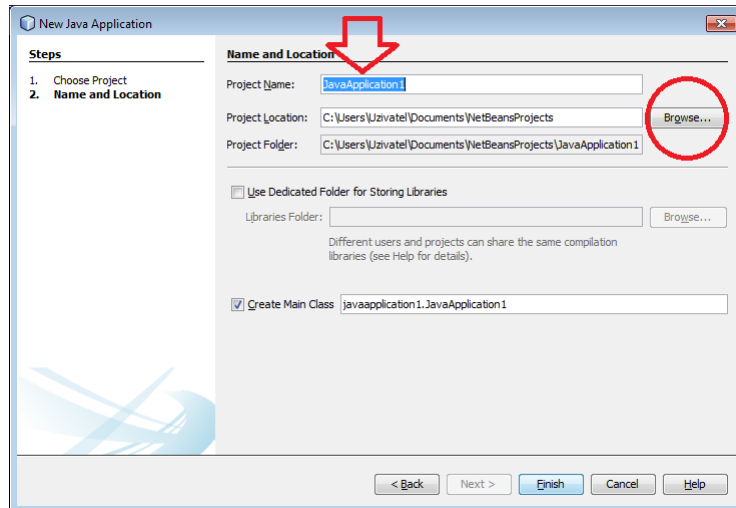
REMARK

Do not try to select other options. These options are intended for advanced users only.

NEW JAVA APPLICATION

After clicking „Next“, new opens a window.

It is important to fill an unique and self-explanatory name for your new application.



Also, if necessary, you can change folders where uour project will be saved. Normally, it is in C:\Users\username\Documents\NetBeansProjects.

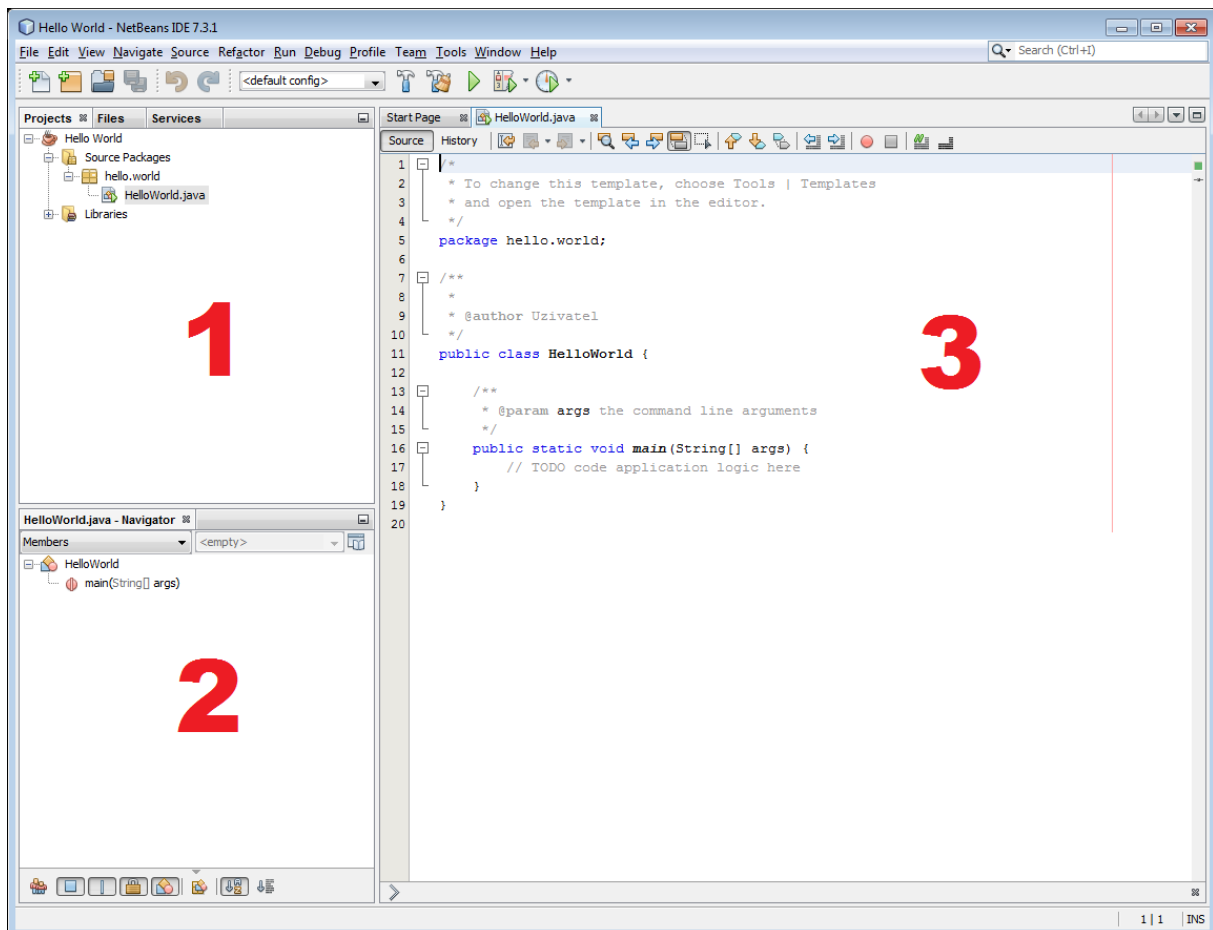
HELLO WORLD

PROGRAMMING ENVIRONMENT

Usually, in any programming langue, teachers show the first program on an example called „Hello World“. This program does nothing more than it writes greeting on screen.

So, please, create an aplicacion „Hello World“. After you create it, you can see your NetBeans screen divided into three main parts:

1. In the first part, we can select our projects, files and libraries. For beginners, it is better not to change anything in this part.
2. Second part is of much better use. Here we can check variables, their values, attributes of classes etc. Now, you cannot understand it, but it will change soon.
3. The third part is of most importance for us. Into this part we will write our programs. Please make a note that there exist some pre-defined texts. These texts form a standard skeleton of any java program.



WRITE A PROGRAM

Please copy the following text

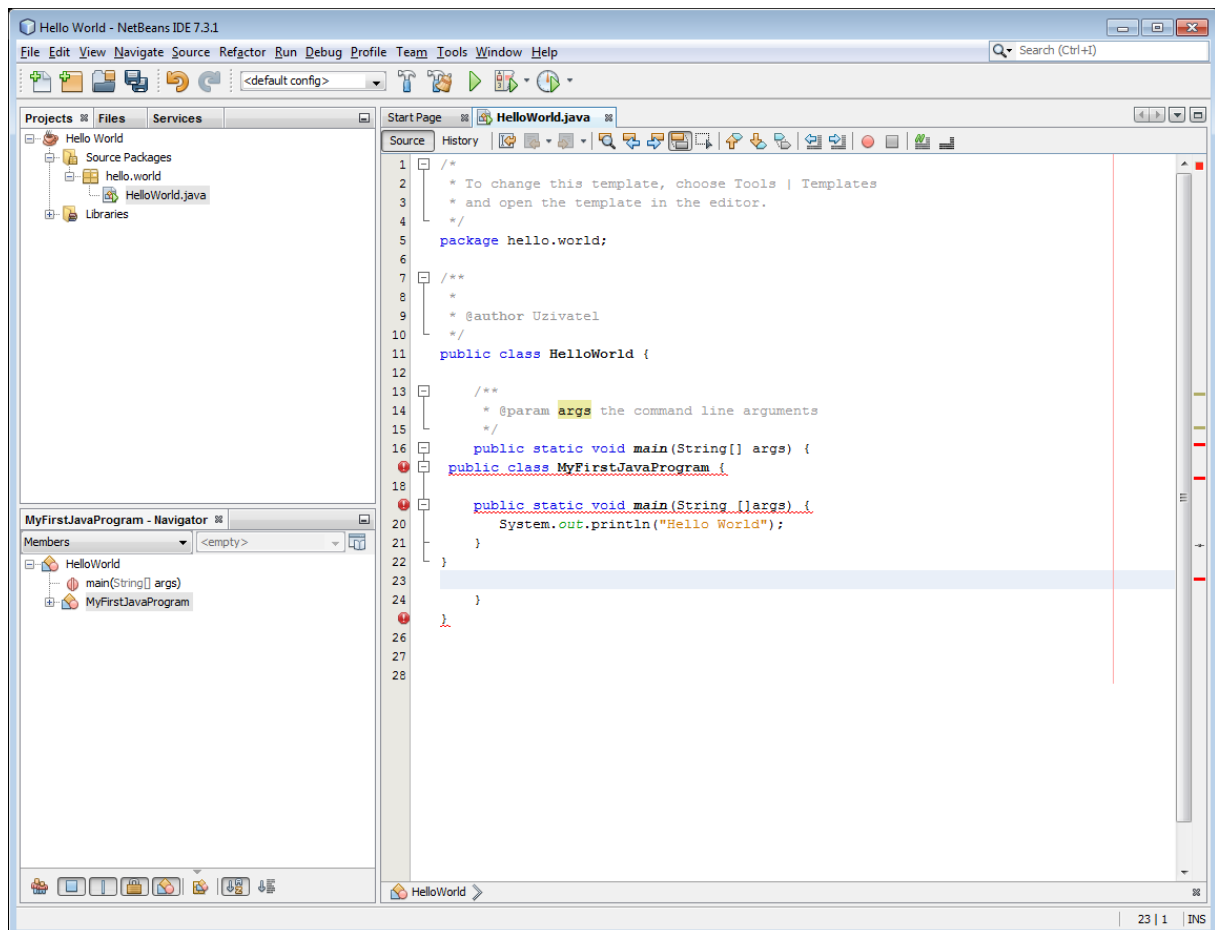
```
public class MyFirstJavaProgram {  
  
    public static void main(String []args) {  
        System.out.println("Hello World");  
    }  
}
```

to the 3rd part of your NetBeans Screen, instead of line

```
// TODO code application logic here
```

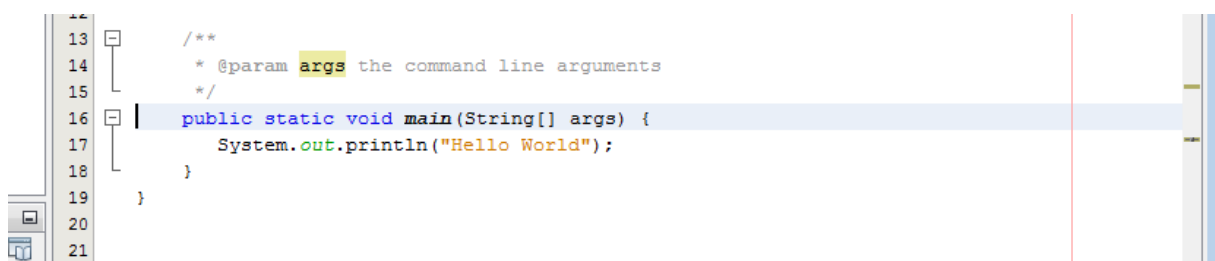
You can use standard Ctrl+C and Ctrl+V shortcuts. Naturally, you cannot understand, what this text means. However, you can write it properly „as is“ to test our java compiler and programming environment.

Your screen changes as follows:



You can see that in the left, red points with exclamation marks appeared. These red points mean that there is an error in this line. (It is better to say that *there is an error somewhere around this line*, or exactly, that this is the place where an error was detected).

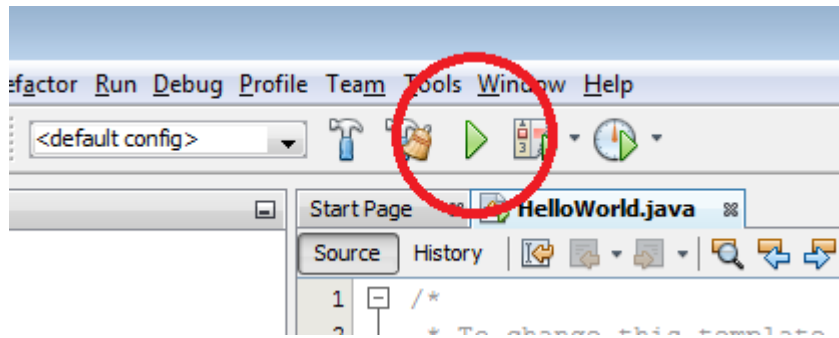
In our code, we intentionally made some errors to show, what happened. After you delete all erroneous lines, our code becomes clean:



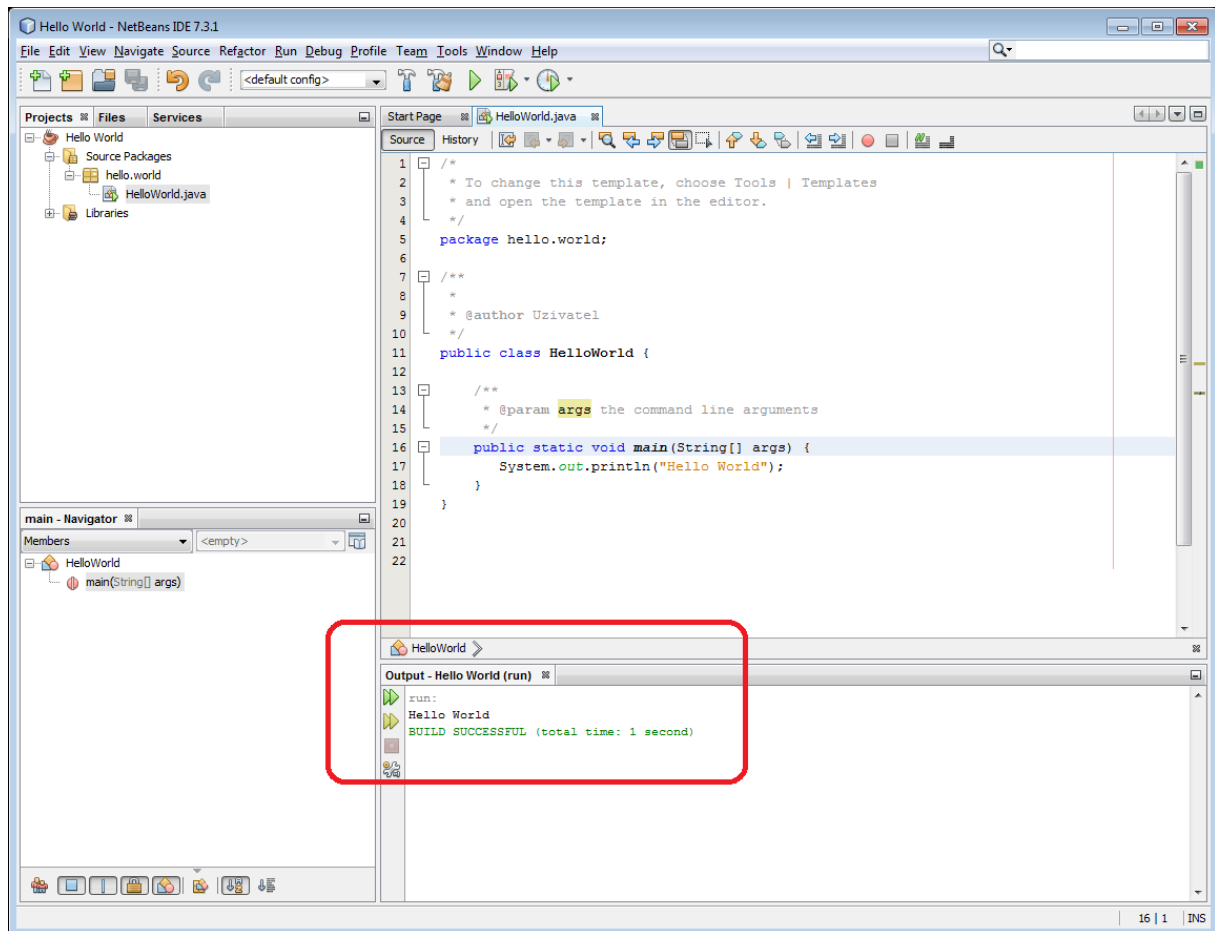
RUN A PROGRAM

You can find a valid source code [here](#).

In the menu, you can either select green arrow, or press Run → Run Project, or press F6. In either case, begins a process of compiling and running of our program.



In the lower part of screen, new window appears. Green colour and „Build Successful“ show that compilation was successful and our first program runs properly.



Just after line „run:“, i.e. before green „BUILD SUCCESSFUL“ you can see results of our program. As it was expected, program wrote „Hello World“ to the output.

STEPPING A PROGRAM

NetBeans compiler helps us to find and remove all syntax errors effectively. But sometimes it happens that we have in our program a semantic error, which is difficult to find. In such case, it is useful to use **breakpoints**, **stepping** and **variable watching**.

Let's write a program similar to Hello World, but a bit more complicated. The inner part of [code](#) will be as follows:

```

public static void main(String[] args) {
    System.out.println("Program begins");
    int x = 123;
    System.out.println("Value x=");
    System.out.println(x);
    x++;
    System.out.println("Now, value x=");
    System.out.println(x);
    System.out.println("Program terminates");
}

```

As you can see, we have several lines like `System.out.println() ;` that, as we saw earlier, prints something to the output. Also, we have a line `int x = 123;` This creates a variable whose name is `x` and whose value is 123.

Now, let's find what happens in line 21, where is command `x++;`

First of all, we will **set a breakpoint** to line 21. This means, that the program will run until it reaches line 21. Just **before** executing line 21, it stops. We can set breakpoint by clicking on number 21 left to the program line, or by right mouse → Toggle Breakpoint, or by `Ctrl+F8`. In either case, a red square appears instead of line number:

```

16 public static void main(String[] args) {
17     System.out.println("Program begins");
18     int x = 123;
19     System.out.println("Value x=");
20     System.out.println(x);
21     x++;
22     System.out.println("Now, value x=");
23     System.out.println(x);
24     System.out.println("Program terminates");
25 }

```

Now, if we run the program (`F6`), it normally runs and terminates.

But if we instead use `Debug` → `Debug Project` (`Ctrl+F5`), it stops just on the beginning of line 21 (red oval). We also can see that variable `x` has value 123 at this moment.

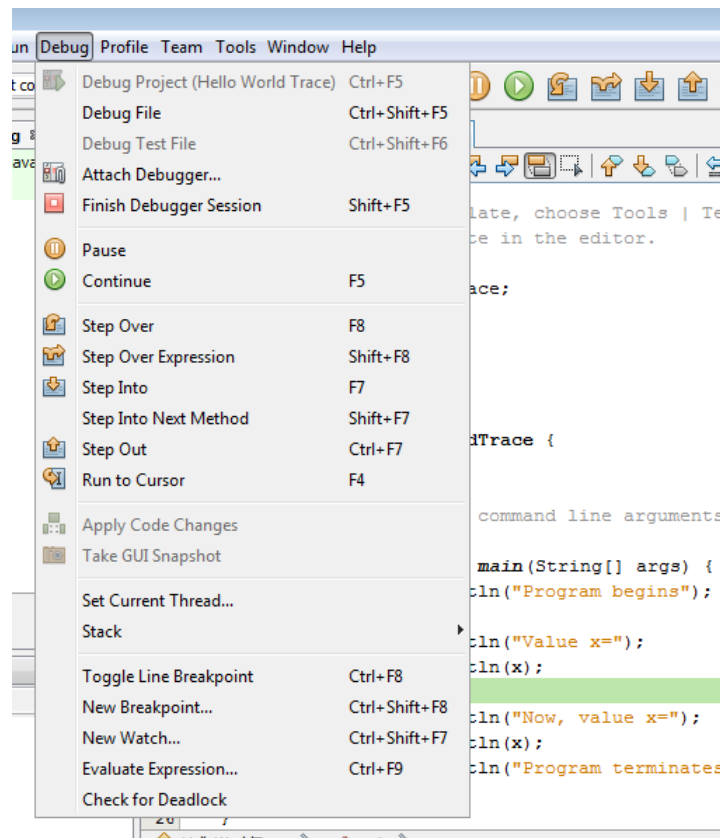
```

14     * @param args the command line arguments
15     */
16     public static void main(String[] args) {
17         System.out.println("Program begins");
18         int x = 123;
19         System.out.println("Value x=");
20         System.out.println(x);
21         x++;
22         System.out.println("Now, value x=");
23         System.out.println(x);
24         System.out.println("Program terminates");
25     }
26 }

```

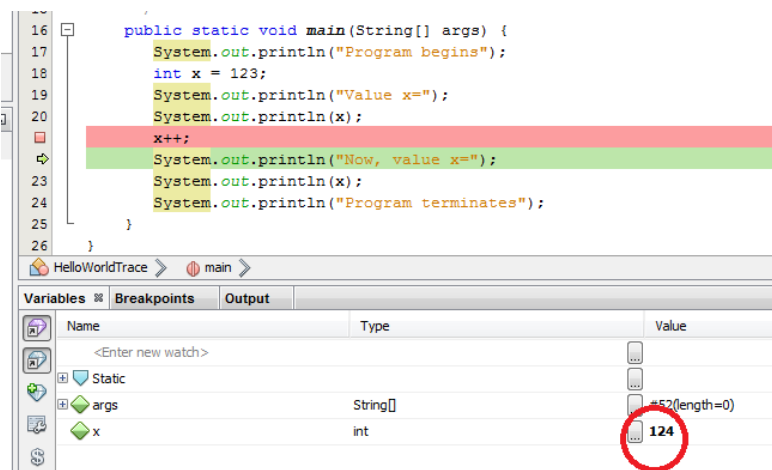
Name	Type	Value
<Enter new watch>		
Static		
args	String[]	#52(length=0)
x	int	123

Now, we can continue running of program (F5), but this is of no use for debugging and troubleshooting. Much better is to step the program, i.e. to go line-by-line. In debug menu we have several options:

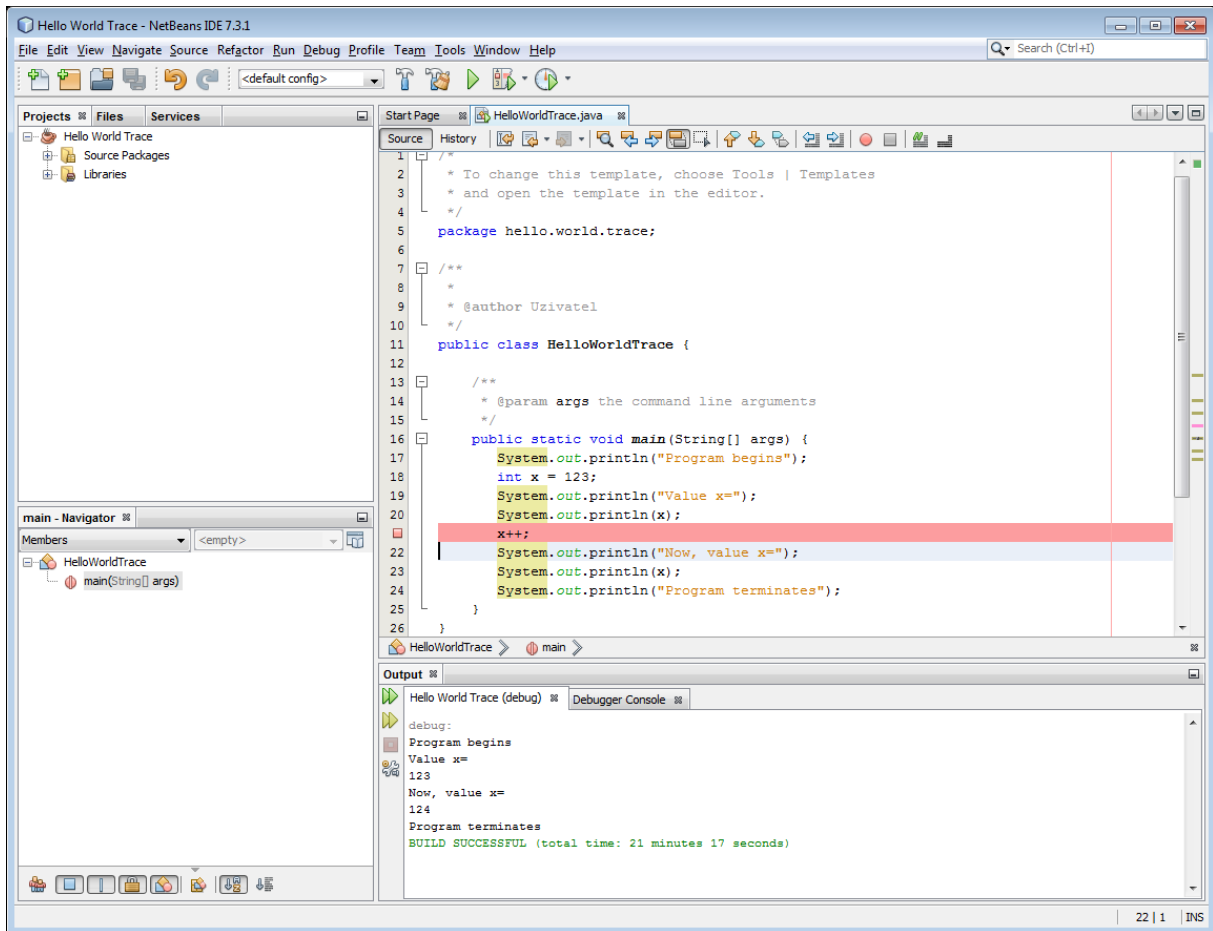


- Step over – processes one line
- Step into – processes one line, but if this line is a procedure call, it steps on first line of the procedure
- Step out – runs until it leaves procedure
- Run to cursor – runs to cursor

In this situation, we use F8 = Step over. Program stops just before line 22. As you can see on debug window, value of `x` changed to 124. In the other words, command `x++`; on line 21 increased value in `x` by 1.



From this point we can step again and again, or continue (F5) to finish the program. Resulting output follows:



REMARK

Though it is possible to use spaces in application names, it is better use names without spaces. As you can see in „src“ directory of your NetBeans project, each space results in one deeper level of directory structure.