



# Cambodian University of Specialties

## Faculty of Science and Technology



### Java Introduction



# Introduction to Java Programming

- Installing NetBeans
- Introduction to Java Programming
- Java Program Structure
- Java Data Types
- Java Operators



# Introduction to Java Programming

- Java Logical Arithmetic
- Java Methods
- Java Access Modifier



# Installing NetBeans

Installing Netbeans





# Installing NetBeans

Sri Lanka Institute of Information Technology  
University of Peradeniya

netbeans.apache.org/front/main/

Apache NetBeans

Search the docs

Community    Participate    Blog    Get Help    Plugins    Download

Latest release

Apache NetBeans 20

Download

Apache NetBeans

Fits the Pieces Together

Development Environment, Tooling Platform and Application Framework.

**Fast & Smart Editing**  
Apache NetBeans is much more than a text editor. It

**Java, JavaScript, PHP, HTML5, CSS, and More**  
Apache NetBeans provides

**Cross Platform**  
Apache NetBeans can be installed on all operating systems that support Java, i.e,



# Installing NetBeans

netbeans.apache.org/front/main/download/nb20/

Apache NetBeans Search the docs Community Participate Blog Get Help Plugins Download

## Downloading Apache NetBeans 20

Apache NetBeans 20 was released on December 1, 2023.

Apache NetBeans 20 is available for download from your closest Apache mirror.

**Binaries (Platform Independent):**

- [netbeans-20-bin.zip \(SHA-512, PGP ASC\)](#)

**Installers and Packages:**

- [Apache-NetBeans-20r1-bin-windows-x64.exe \(SHA-512, PGP ASC\)](#)
- [Apache-NetBeans-20.pkg \(SHA-512, PGP ASC\)](#)
- [apache-netbeans\\_20-1\\_all.deb \(SHA-512, PGP ASC\)](#)
- [apache-netbeans-20-0.noarch.rpm \(SHA-512, PGP ASC\)](#)
- [Linux snap package](#)

**Source:**

- [netbeans-20-source.zip \(SHA-512, PGP ASC\)](#)

Officially, it is important that you [verify the integrity](#) of the downloaded files using the PGP signatures (.asc file) or a hash (.sha512 file). The PGP keys used to sign this release are available [here](#).

Community Installers  
Deployment Platforms  
Known Issues  
Building from Source  
Community Approval  
Earlier Releases



# Installing NetBeans

The screenshot shows the Apache Software Foundation website for NetBeans. The URL in the address bar is apache.org/dyn/closer.lua/netbeans/netbeans-installers/20/Apache-NetBeans-20r1-bin-windows-x64.exe. The page features a dark blue header with social media icons (GitHub, LinkedIn, YouTube, Twitter) and navigation links for Community, Projects, Downloads, Learn, Resources & Tools, About, and a search icon. A "Sponsor the ASF" button is also present. The main content area displays the Apache Software Foundation logo and a red-bordered link to the download file.



We suggest the following location for your download:

<https://dlcdn.apache.org/netbeans/netbeans-installers/20/Apache-NetBeans-20r1-bin-windows-x64.exe>

Alternate download locations are suggested below.

It is essential that you verify the integrity of the downloaded file using the PGP signature ( [.asc](#) file) or a hash ( [.md5](#) or [.sha\\*](#) file).

## HTTP

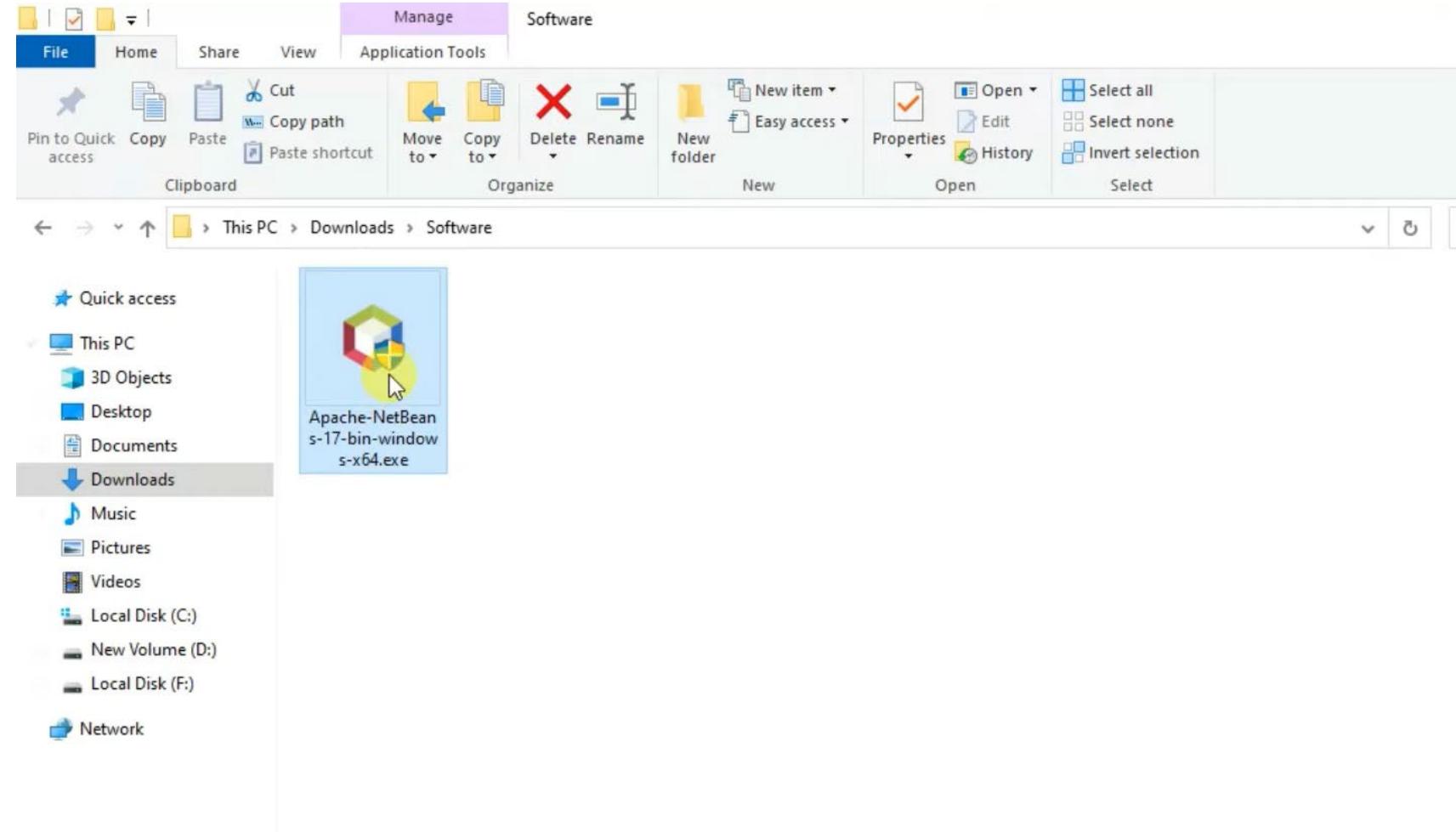
<https://dlcdn.apache.org/netbeans/netbeans-installers/20/Apache-NetBeans-20r1-bin-windows-x64.exe>

## BACKUP SITES

<https://dlcdn.apache.org/netbeans/netbeans-installers/20/Apache-NetBeans-20r1-bin-windows-x64.exe>

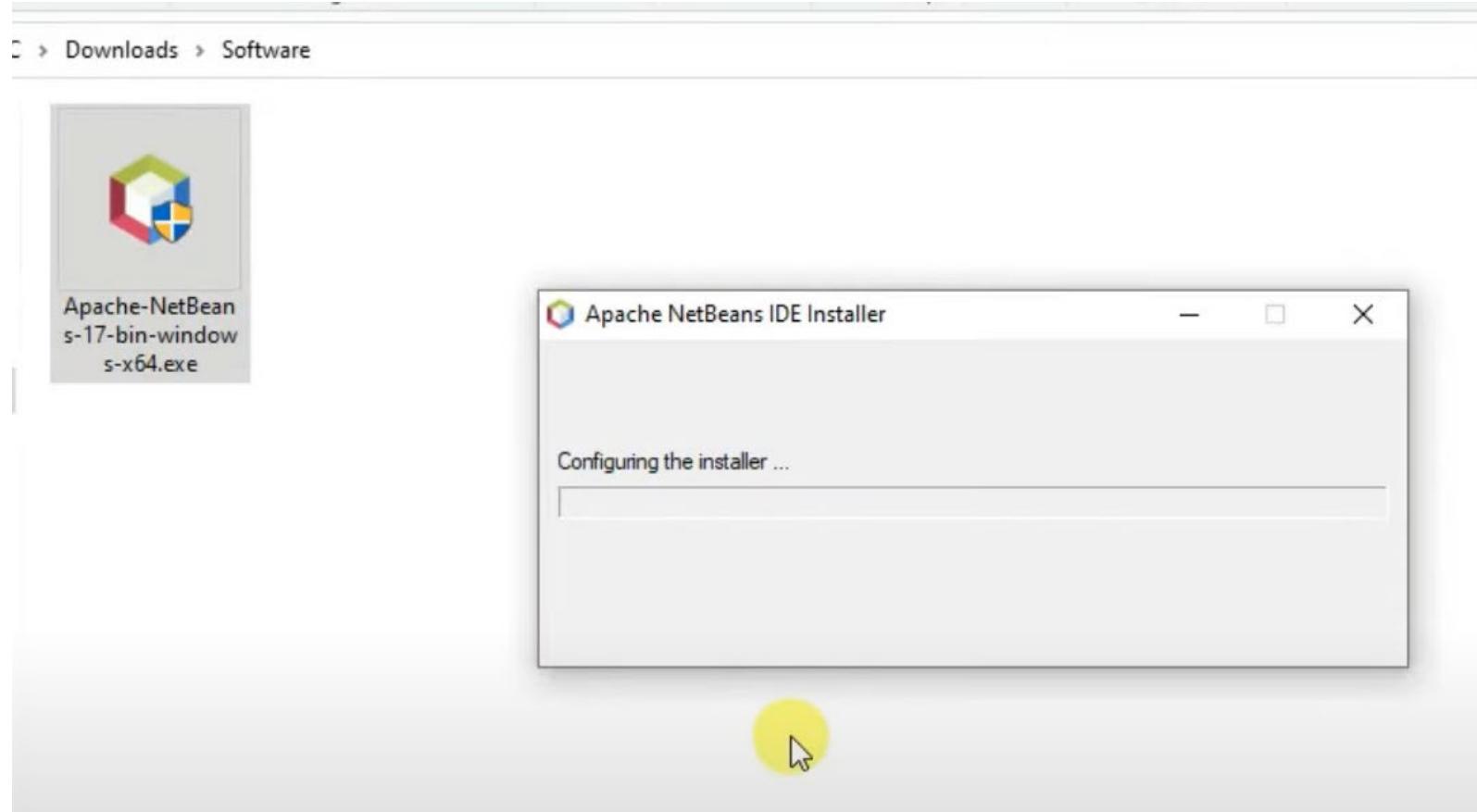


# Installing NetBeans



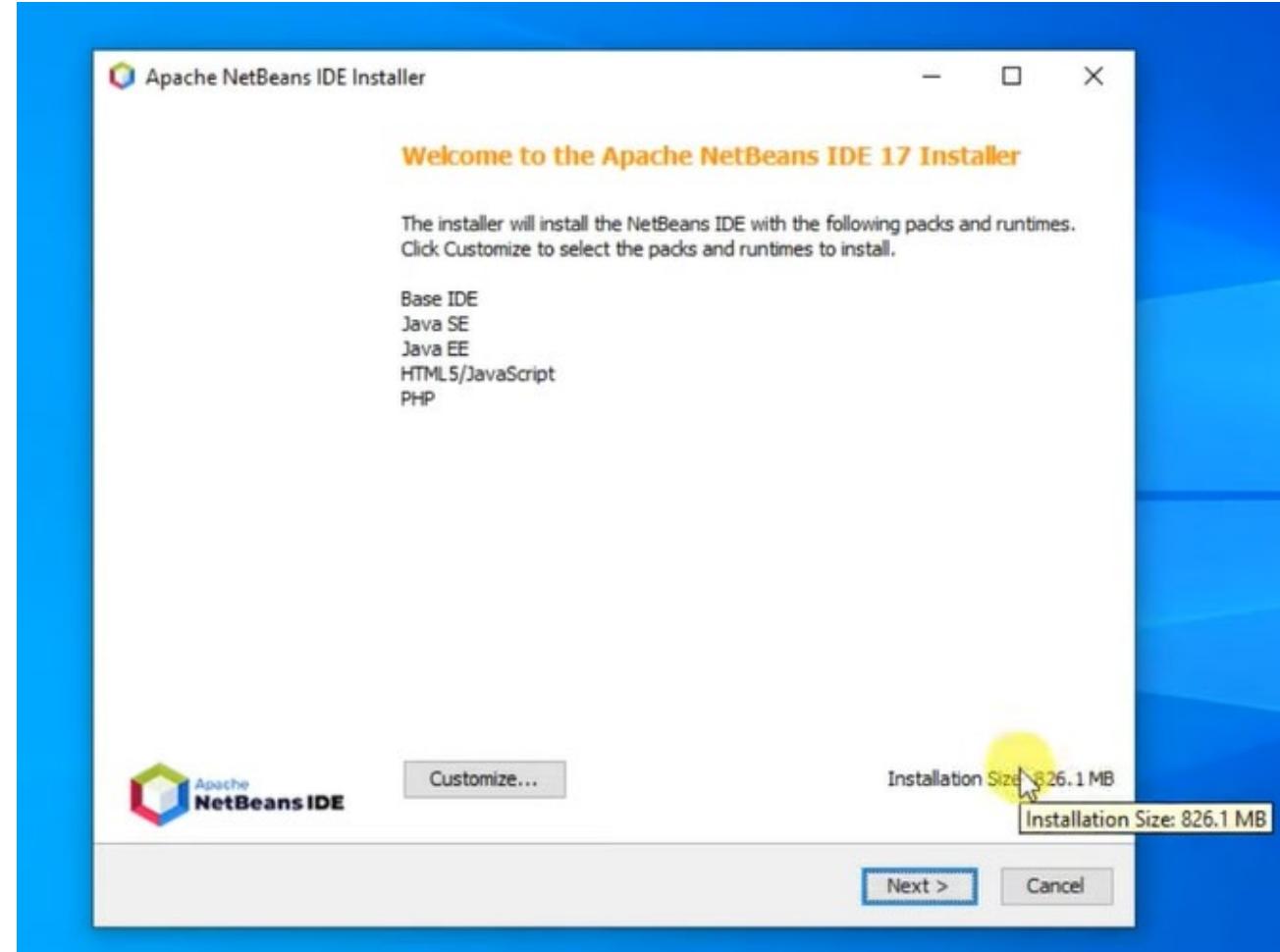


# Installing NetBeans





# Installing NetBeans





# Installing NetBeans

Apache NetBeans IDE Installer

**License Agreement**

Please read the following license agreement carefully.

APACHE NETBEANS IDE DEVELOPMENT VERSION ("Product")  
LICENSE AGREEMENT

PLEASE READ THE FOLLOWING LICENSE AGREEMENT TERMS AND CONDITIONS CAREFULLY, INCLUDING WITHOUT LIMITATION THOSE DISPLAYED ELSEWHERE (AS INDICATED BY LINKS LISTED BELOW), BEFORE USING THE SOFTWARE. THESE TERMS AND CONDITIONS CONSTITUTE A LEGAL AGREEMENT BETWEEN YOU, OR THE ENTITY FOR WHICH YOU ARE AN AUTHORIZED REPRESENTATIVE WITH FULL AUTHORITY TO ENTER INTO THIS AGREEMENT, AND APACHE. BY CLICKING "ACCEPT" OR THE EQUIVALENT YOU AGREE TO ALL OF THE TERMS AND CONDITIONS OF THIS LICENSE AGREEMENT. IF YOU DO NOT AGREE TO THIS LICENSE DO NOT CLICK "ACCEPT" OR THE EQUIVALENT AND DO NOT INSTALL OR USE THIS SOFTWARE.

Apache License  
Version 2.0, January 2004  
<http://www.apache.org/licenses/>

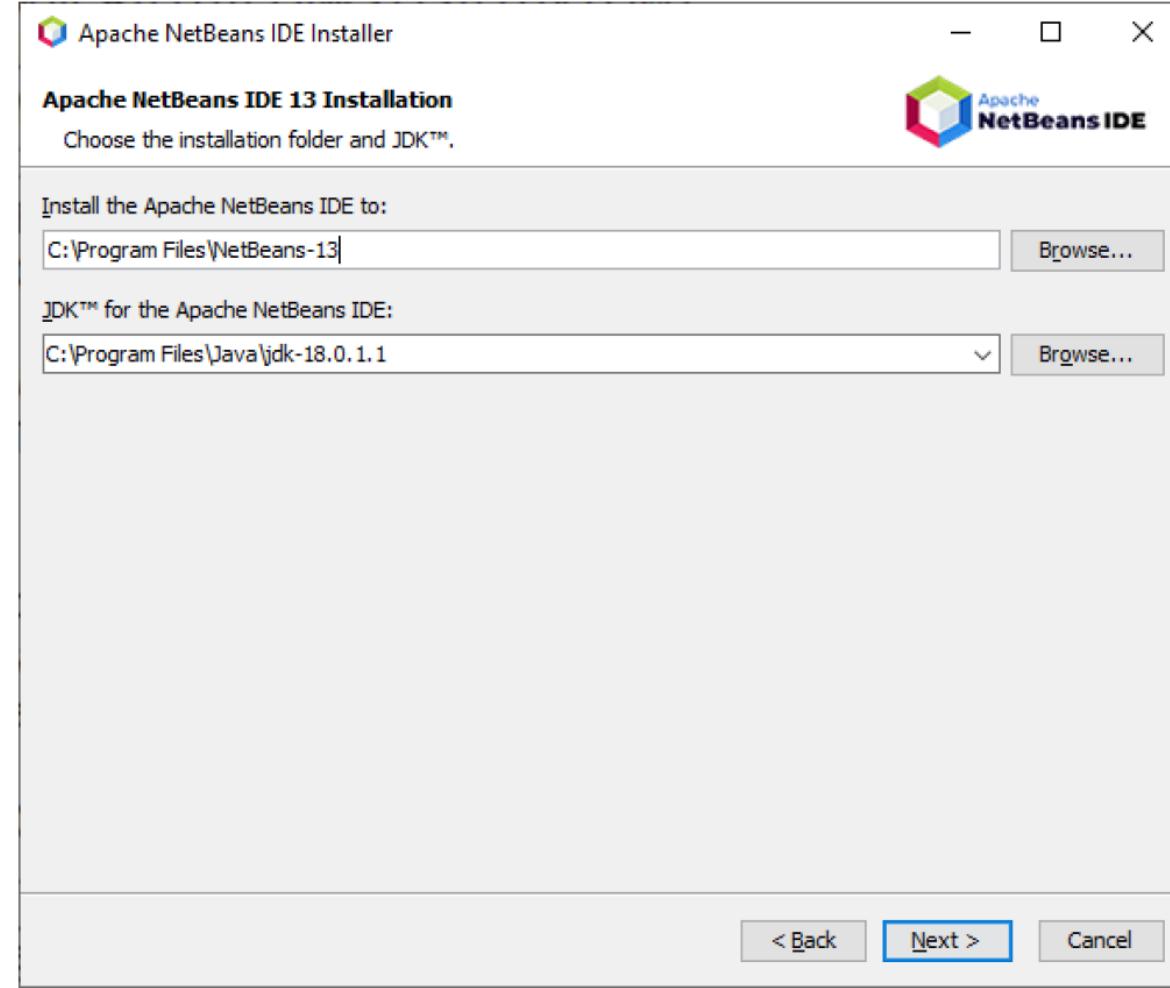
TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

I accept the terms in the license agreement

< Back    Next >    Cancel

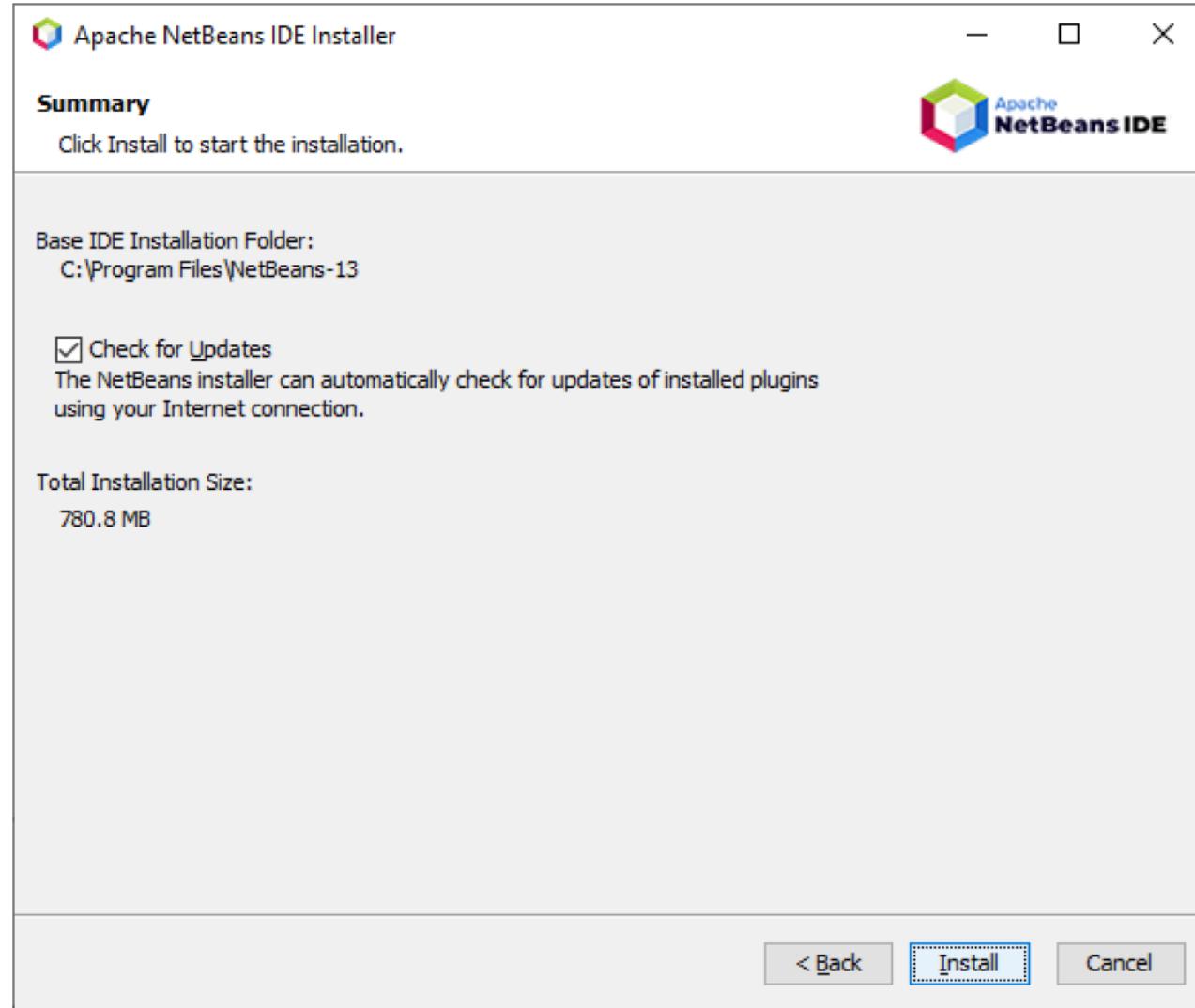


# Installing NetBeans



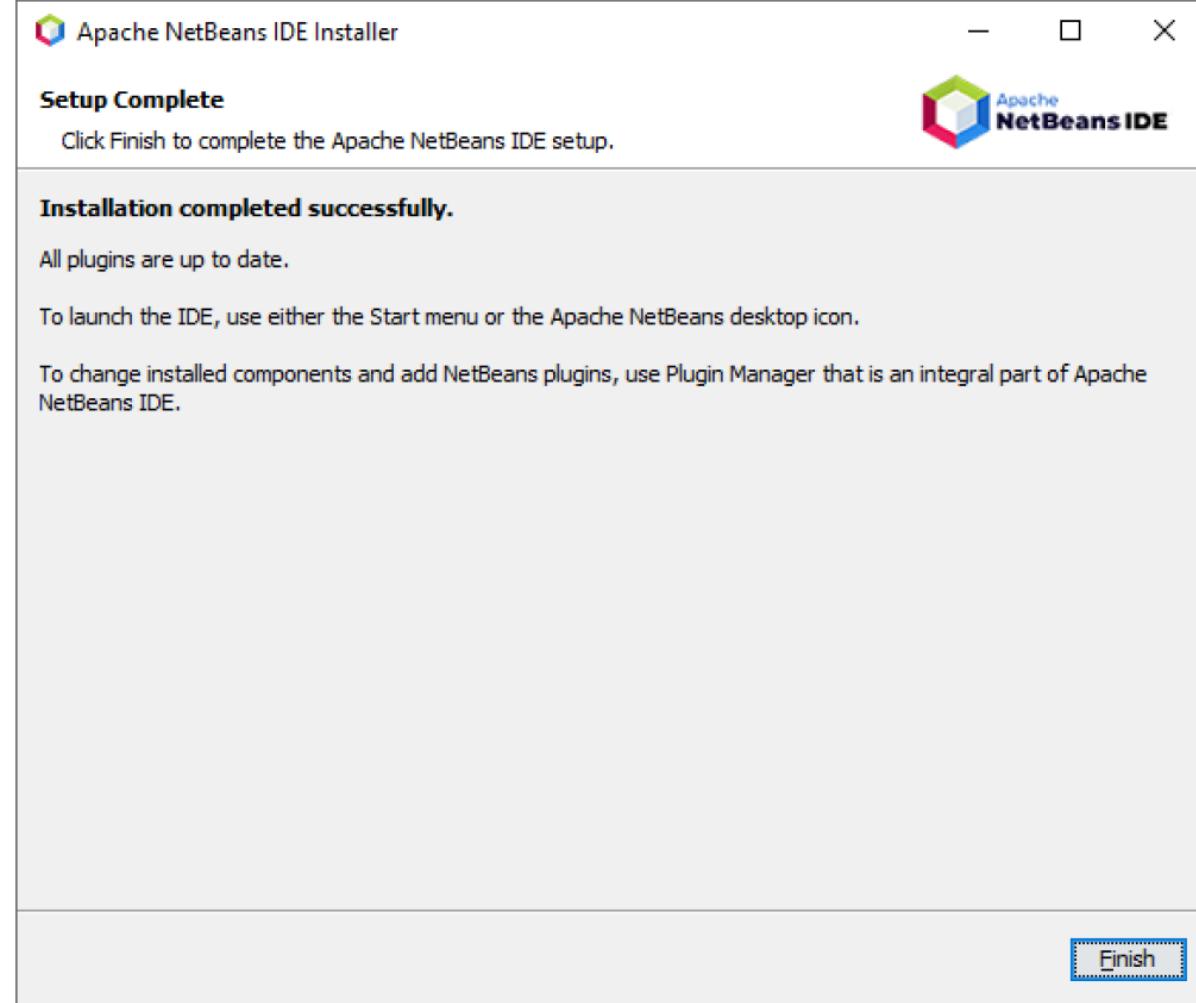


# Installing NetBeans



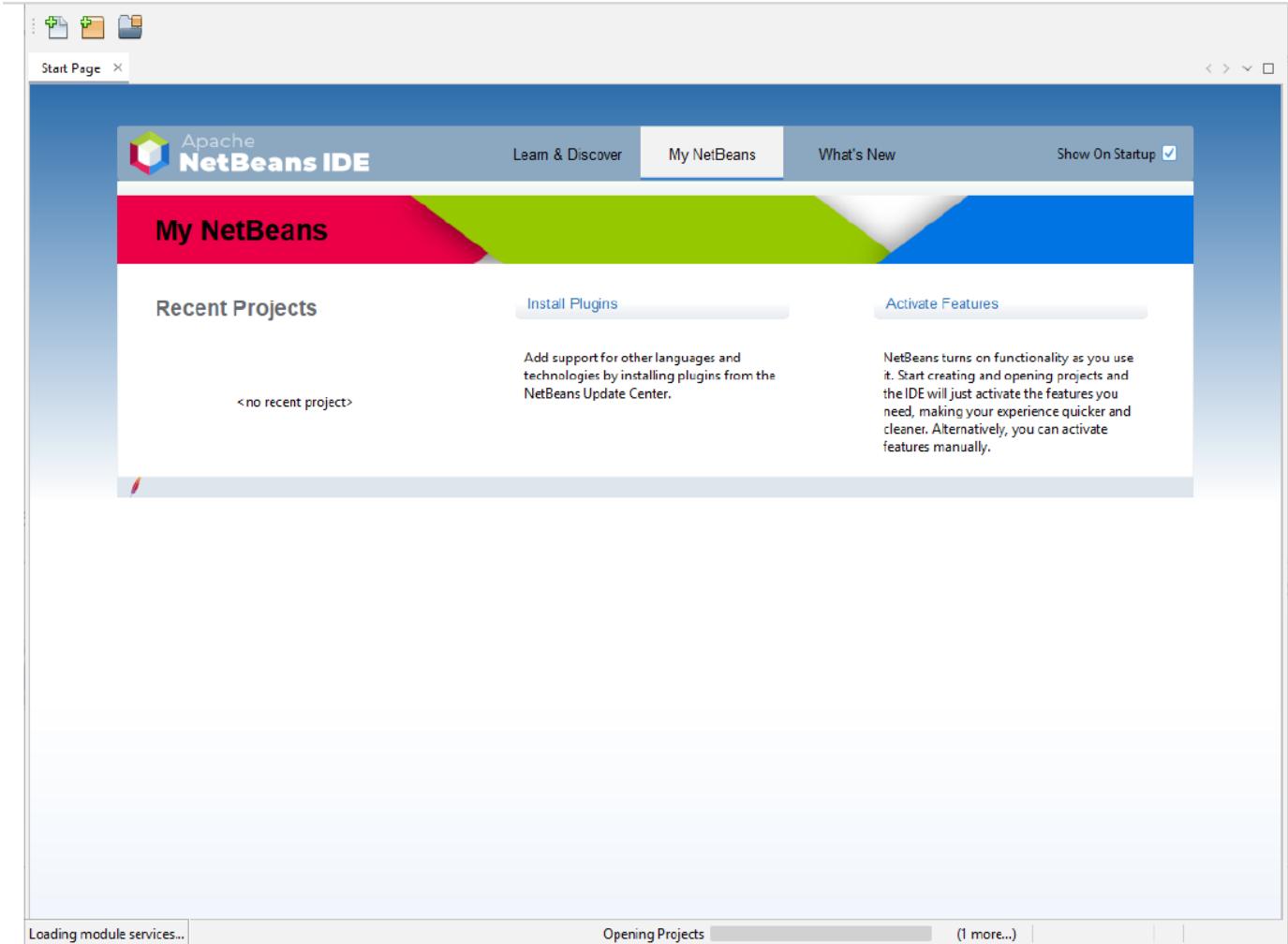


# Installing NetBeans



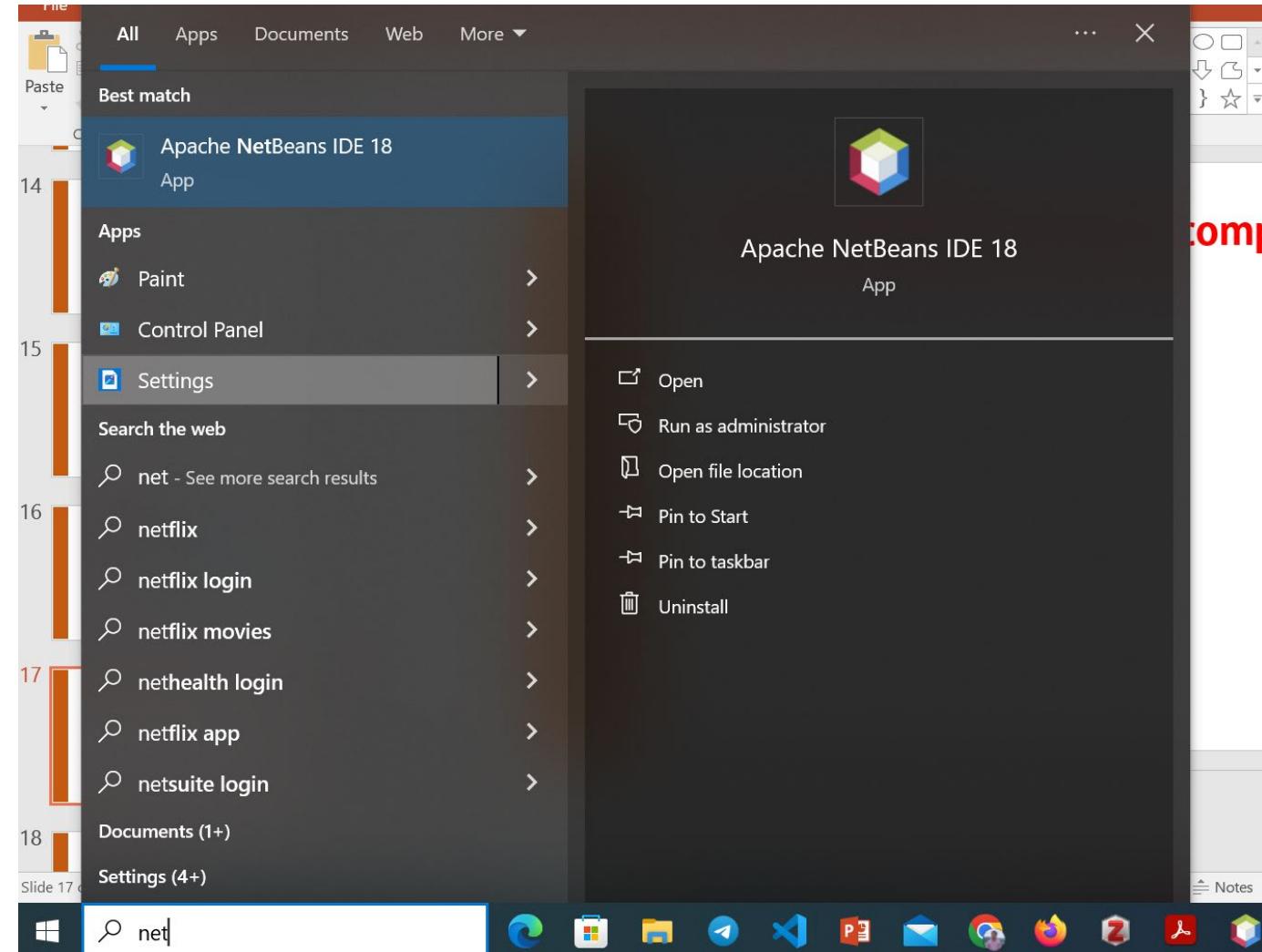


# Installing NetBeans





# Starting NetBeans on your computer





# Installing NetBeans

The screenshot shows the Apache NetBeans IDE 18 interface. On the left, the Project Explorer displays a Java application named "JavaApplication12" with source packages like "javaapplication12" containing files such as CallableJDBC.java, InsertDatabase.java, JavaApplication12.java, and NewClass.java. It also lists libraries including mysql-connector-j-8.0.3, commons-beanutils-1.7, commons-collections-2, commons-digester-1.7, commons-javaflow-200, commons-logging-1.0.2, commons-logging-api, groovy-all-1.5.5.jar, jasperreports-3.0.0.jar, itext-1.3.1.jar, and JDK 17 (Default). The main window shows the "Start Page" with the Apache NetBeans IDE logo at the top. Below it, a "What's New" section highlights recent releases:

- [ANNOUNCE] Apache NetBeans 20 Released (12/1/23): The Apache NetBeans team is pleased to announce that Apache NetBeans 20 was released on December 1, 2023. What's in the Apache NetBeans 20 release: <https://github.com/apache/netbeans/releases/tag/20> With thanks...
- [ANNOUNCE] Apache NetBeans 19 Released (9/1/23): The Apache NetBeans team is pleased to announce that Apache NetBeans 19 was released on September 1, 2023. What's in the Apache NetBeans 19 release: <https://github.com/apache/netbeans/releases/tag/19> With thank...
- [ANNOUNCE] Apache NetBeans 18 Released (6/1/23): The Apache NetBeans team is pleased to announce that Apache NetBeans 18 was released on May 30, 2023. What's in the Apache NetBeans 18 release: <https://github.com/apache/netbeans/releases/tag/18> With thanks to ...
- [ANNOUNCE] Apache NetBeans 17 Released (2/21/23): The Apache NetBeans team is pleased to announce that Apache NetBeans 17 was released on February 21, 2023. What's in the Apache NetBeans 17 release: <https://github.com/apache/netbeans/releases/tag/17> With thank...
- [ANNOUNCE] Apache NetBeans 16 Released (12/15/22): The Apache NetBeans team is pleased to announce that Apache NetBeans 16 was released on November 30, 2022 (though there's been a delay in announcing the release). What's in the Apache NetBeans 16 release: h...

At the bottom, the taskbar shows the Windows Start button, a search bar with "Type here to search", and various pinned icons including File Explorer, Mail, and Microsoft Office applications. The system tray shows the date and time as "8:09 AM 12/18/2023" and a notification for 6 new messages.



# New Project in NetBeans

The screenshot shows the Apache NetBeans IDE 18 interface. The 'File' menu is open, displaying options like 'New Project...', 'New File...', 'Open Project...', and 'Save'. The 'New Project...' option is highlighted. The main workspace shows a Java file named 'InsertDatabase.java' and a 'What's New' section. The taskbar at the bottom includes icons for various Windows applications and system status.

**File**

- New Project... Ctrl+Shift+N
- New File... Ctrl+N
- Open Project... Ctrl+Shift+O
- Open Recent Project
- Close Project
- Close Other Projects
- Close All Projects
- Open File...
- Open Recent File
- Project Groups...
- Project Properties
- Import Project >
- Export Project >
- Save Ctrl+S
- Save As...
- Save All Ctrl+Shift+S
- Page Setup...
- Print... Ctrl+Alt+Shift+P
- Print to HTML...
- Exit

Apache NetBeans IDE 18

Search (Ctrl+I)

New Project X InsertDatabase.java X

Apache NetBeans IDE Learn & Discover My NetBeans What's New Show On Startup

## What's New

### News

[ANNOUNCE] Apache NetBeans 20 Released 12/1/23  
The Apache NetBeans team is pleased to announce that Apache NetBeans 20 was released on December 1, 2023. What's in the Apache NetBeans 20 release: <https://github.com/apache/netbeans/releases/tag/20> With thanks...

[ANNOUNCE] Apache NetBeans 19 Released 9/1/23  
The Apache NetBeans team is pleased to announce that Apache NetBeans 19 was released on September 1, 2023. What's in the Apache NetBeans 19 release: <https://github.com/apache/netbeans/releases/tag/19> With thank...

[ANNOUNCE] Apache NetBeans 18 Released 6/1/23  
The Apache NetBeans team is pleased to announce that Apache NetBeans 18 was released on May 30, 2023. What's in the Apache NetBeans 18 release: <https://github.com/apache/netbeans/releases/tag/18> With thanks to ...

[ANNOUNCE] Apache NetBeans 17 Released 2/21/23  
The Apache NetBeans team is pleased to announce that Apache NetBeans 17 was released on February 21, 2023. What's in the Apache NetBeans 17 release: <https://github.com/apache/netbeans/releases/tag/17> With thank...

[ANNOUNCE] Apache NetBeans 16 Released 12/15/22  
The Apache NetBeans team is pleased to announce that Apache NetBeans 16 was released on November 30, 2022 (though there's been a delay in announcing the release). What's in the Apache NetBeans 16 release: h...

Type here to search

INS

8:11 AM 12/18/2023 6



# New Project in NetBeans

The screenshot shows the Apache NetBeans IDE 18 interface. The left sidebar displays a project tree for 'JavaApplication12' containing 'Source Packages' like 'javaapplication12' and 'CallableJDBC.java', and 'Libraries' such as 'mysql-connector-j-8.0.3'. The main area features a 'New Project' wizard titled 'Choose Project'. The 'Steps' section shows '1. Choose Project' and '2. ...'. The 'Categories' section includes 'Java with Maven', 'Java with Gradle', and 'Java with Ant' (which is selected). The 'Projects' section lists 'Java Application', 'Java Class Library', 'Java Project with Existing Sources', 'Java Modular Project', and 'Java Free-Form Project'. A detailed description box explains that it creates a new Java SE application using an IDE-generated Ant build script. At the bottom, there are buttons for 'Back', 'Next >', 'Finish', 'Cancel', and 'Help'. An announcement at the bottom left states: '[ANNOUNCE] Apache NetBeans 16 Released' with the date '12/15/22'.



# New Project in NetBeans

The screenshot shows the Apache NetBeans IDE 18 interface. The main window title is "Apache NetBeans IDE 18". The toolbar at the top includes icons for File, Edit, Format, Preview, View, Navigate, Source, Refactor, Run, Debug, Profile, Team, Tools, Window, Help, and a search bar. The status bar at the bottom shows memory usage (438.9/510.0MB) and system information (Windows 10, ENG, 8:12 AM, 12/18/2023).

The left sidebar displays the project structure under "Project Navigator":

- JavaApplication12
- Source Packages
  - javaapplication12
    - CallableJDBC.java
    - InsertDatabase.java
    - JavaApplication12.java
    - NewClass.java
  - Libraries
    - mysql-connector-j-8.0.3
    - commons-beanutils-1.7
    - commons-collections-2.1
    - commons-digester-1.7.j
    - commons-javaflow-200
    - commons-logging-1.0.2
    - commons-logging-api
    - groovy-all-1.5.5.jar
    - jasperreports-3.0.0.jar
    - itext-1.3.1.jar
    - JDK 17 (Default)

The central area shows the "Start Page" and the "New Java Application" dialog. The dialog has two tabs: "Steps" (selected) and "Name and Location". The "Name and Location" tab contains the following fields:

- Project Name:
- Project Location:
- Project Folder:
- Use Dedicated Folder for Storing Libraries  
Libraries Folder:  

Different users and projects can share the same compilation libraries (see Help for details).
- Create Main Class

At the bottom of the dialog, there are buttons for "Back", "Next >", "Finish" (highlighted in blue), "Cancel", and "Help". A small announcement at the bottom right reads: "[ANNOUNCE] Apache NetBeans 16 Released 12/15/22".



# New Project in NetBeans

The screenshot shows the NetBeans IDE interface with the following details:

- Project Explorer:** Shows the project structure under "JavaApplication12". It includes a "Source Packages" folder containing "javaapplication12" with files: CallableJDBC.java, InsertDatabase.java, JavaApplication12.java, and NewClass.java. It also lists "Libraries" with various JAR files like mysql-connector-j-8.0.3.jar, commons-beanutils-1.7.jar, etc.
- Code Editor:** The "JavaApplication13.java" file is open. The code is as follows:

```
10  /*
11   * To change this license header, choose License Headers in Project Properties.
12   * To change this template, choose Tools | Templates
13   * and open the template in the editor.
14   *
15   * @param args the command line arguments
16   */
17  public static void main(String[] args) {
18      System.out.println("Hello You");
19  }
20
21
22 }
```
- Status Bar:** Shows "JavaApplication13.java saved." at the bottom left and "17:41" at the bottom right.
- Taskbar:** At the very bottom, it shows the Windows taskbar with icons for File Explorer, Edge browser, File Manager, Task View, Mail, Google Chrome, Firefox, Adobe Acrobat Reader, and Microsoft Word. It also displays system status like "ENG", "8:13 AM", "12/18/2023", and a notification icon with the number "6".



# New Project in NetBeans

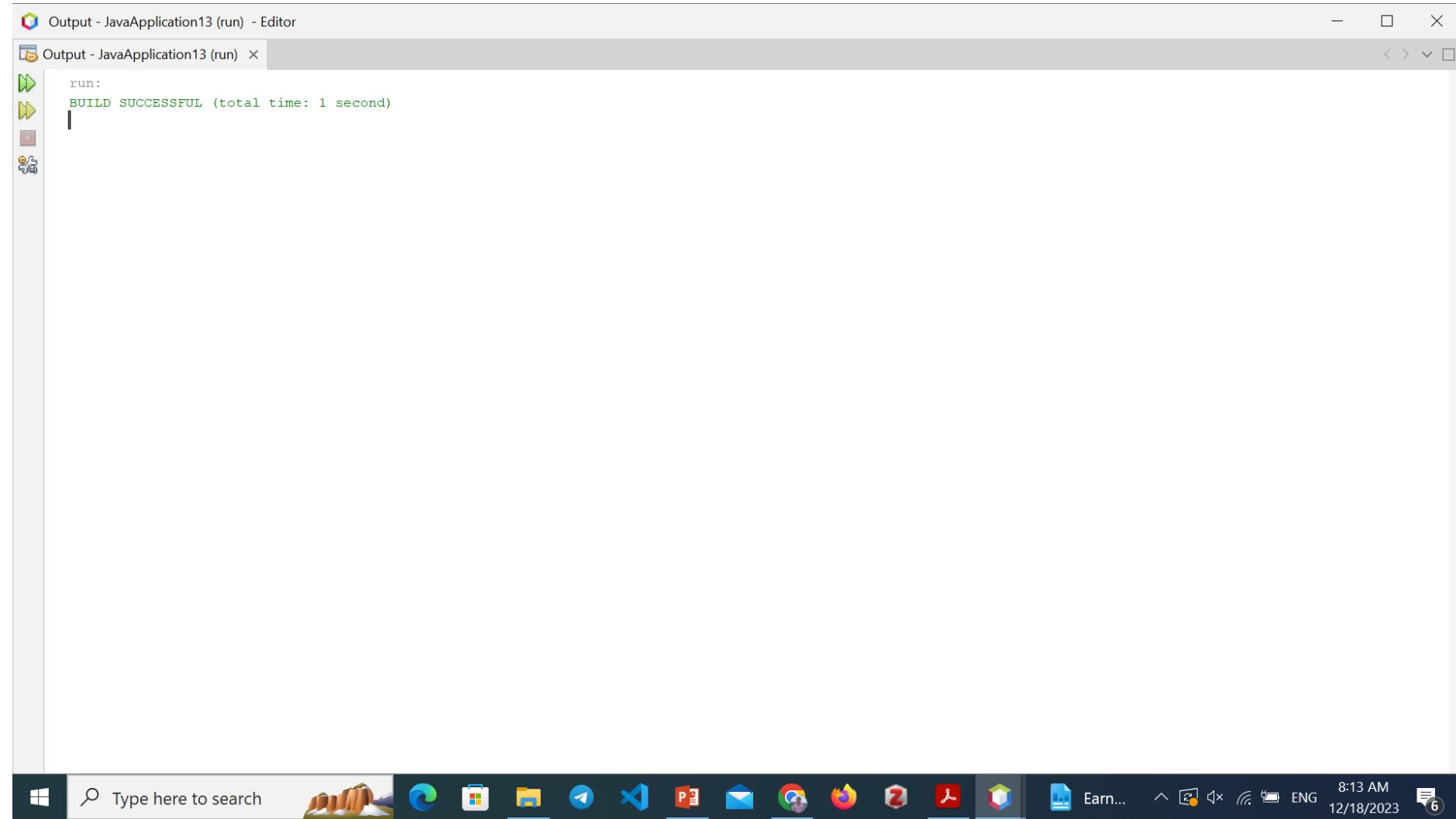
The screenshot shows the NetBeans IDE interface. The menu bar at the top includes File, Edit, Format, Preview, View, Navigate, Source, Refactor, Run, Debug, Profile, Team, Tools, Window, Help, and JavaApplication13 - Apac... The toolbar below the menu has icons for Run, Stop, Test, Build, Clean, and others. The left sidebar displays the Project Explorer with JavaApplication12 and JavaApplication13 projects, and the Libraries section listing various Java dependencies like mysql-connector-j, commons-beanutils, commons-collections, commons-digester, commons-javaflow, commons-logging, commons-logging-api, groovy-all, jasperreports, and iText. The main editor window shows a Java code snippet:

```
10  * /
11  public class
12  {
13      /* */
14      /**
15      * @
16      */
17      public
18  }
19
20
21
22 }
```

A context menu is open over the code at line 17, showing options like Run Project (JavaApplication13), Test Project (JavaApplication13), Build Project (JavaApplication13), Clean and Build Project (JavaApplication13), Set Project Configuration, Set Project Browser, Set Main Project, Open Java Shell for Project (JavaApplication13), Generate Javadoc (JavaApplication13), Run File, Test File, Compile File, Check File, Validate File, Repeat Build/Run, Stop Build/Run, and Report preview. The status bar at the bottom shows the time as 17:41 and the date as 12/18/2023.



# New Project in NetBeans





# Introduction to Java Programming

- Java is a **programming language** and a **platform**. Java is a high level, robust, object-oriented and secure programming language.
- Java was developed by *Sun Microsystems* (which is now the subsidiary of Oracle) in the year 1995. *James Gosling* is known as the father of Java. Before Java, its name was *Oak*. Since Oak was already a registered company, so James Gosling and his team changed the name from Oak to Java.



# Introduction to Java Programming

## Types of Java Applications

1. Standalone Application
2. Web Application
3. Enterprise Application
4. Mobile Application



# Introduction to Java Programming

## Standalone Application

Standalone applications are also known as desktop applications or window-based applications. These are traditional software that we need to install on every machine. Examples of standalone application are Media player, antivirus, etc. AWT and Swing are used in Java for creating standalone applications.



# Introduction to Java Programming

## Web Application

An application that runs on the server side and creates a dynamic page is called a web application.

Currently, Servlet, JSP, Struts, Spring, Hibernate, JSF, etc. technologies are used for creating web applications in Java.



# Introduction to Java Programming

## Enterprise Application

An application that is distributed in nature, such as banking applications, etc. is called an enterprise application. It has advantages like high-level security, load balancing, and clustering.

In Java, EJB is used for creating enterprise applications.



# Introduction to Java Programming

## Mobile Application

An application which is created for mobile devices is called a mobile application. Currently, Android and Java ME are used for creating mobile applications.



# Introduction to Java Programming

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



# Java Data Types

Data types are divided into two groups:

- Primitive data types -

includes `byte`, `short`, `int`, `long`, `float`, `double`, `boolean` and `char`

- Non-primitive data types - such

as [String](#), [Arrays](#) and [Classes](#) (you will learn more about these in a later chapter)



# Java Data Types

Data Type	Size	Description
byte	1 byte	Stores whole numbers from -128 to 127
short	2 bytes	Stores whole numbers from -32,768 to 32,767
int	4 bytes	Stores whole numbers from -2,147,483,648 to 2,147,483,647
long	8 bytes	Stores whole numbers from -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
float	4 bytes	Stores fractional numbers. Sufficient for storing 6 to 7 decimal digits
double	8 bytes	Stores fractional numbers. Sufficient for storing 15 decimal digits
boolean	1 bit	Stores true or false values
char	2 bytes	Stores a single character/letter or ASCII values



# Java Variables

Variables are containers for storing data values.

In Java, there are different **types** of variables, for example:

- **String** - stores text, such as "Hello". String values are surrounded by double quotes
- **int** - stores integers (whole numbers), without decimals, such as 123 or -123
- **float** - stores floating point numbers, with decimals, such as 19.99 or -19.99
- **char** - stores single characters, such as 'a' or 'B'. Char values are surrounded by single quotes
- **boolean** - stores values with two states: true or false



# Java Variables

- Declaring (Creating) Variables
- To create a variable, you must specify the type and assign it a value:

## Syntax

```
type variableName = value;
```



# Java Variables

```
public class Main
{
    public static void main(String[] args)
    {
        String name = "John";
        System.out.println(name);
    }
}
```

John



# Java Variables

```
int myNum = 5;
```

```
float myFloatNum = 5.99f;
```

```
char myLetter = 'D';
```

```
boolean myBool = true;
```

```
String myText = "Hello";
```



# Java Operators

Operators are used to perform operations on variables and values.

- Java divides the operators into the following groups:
- Arithmetic operators
- Assignment operators
- Comparison operators
- Logical operators
- Bitwise operators



# Java Operators

- **Arithmetic operators**

Arithmetic operators are used to perform common mathematical operations.

Operator	Name	Description	Example
+	Addition	Adds together two values	$x + y$
-	Subtraction	Subtracts one value from another	$x - y$
*	Multiplication	Multiplies two values	$x * y$



# Java Operators

## • Arithmetic operators

Arithmetic operators are used to perform common mathematical operations.

Operator	Name	Description	Example
/	Division	Divides one value by another	$x / y$
%	Modulus	Returns the division remainder	$x \% y$
++	Increment	Increases the value of a variable by 1	$++x$
--	Decrement	Decreases the value of a variable by 1	$--x$



# Java Operators

- Assignment operators

Assignment operators are used to assign values to variables.

Operator	Example	Same As
=	x = 5	x = 5
+=	x += 3	x = x + 3
-=	x -= 3	x = x - 3
*=	x *= 3	x = x * 3



# Java Operators

- Assignment operators

Assignment operators are used to assign values to variables.

Operator	Example	Same As
<code>/=</code>	<code>x /= 3</code>	<code>x = x / 3</code>
<code>%=</code>	<code>x %= 3</code>	<code>x = x % 3</code>
<code>&amp;=</code>	<code>x &amp;= 3</code>	<code>x = x &amp; 3</code>
<code> =</code>	<code>x  = 3</code>	<code>x = x   3</code>
<code>^=</code>	<code>x ^= 3</code>	<code>x = x ^ 3</code>
<code>&gt;&gt;=</code>	<code>x &gt;&gt;= 3</code>	<code>x = x &gt;&gt; 3</code>
<code>&lt;&lt;=</code>	<code>x &lt;&lt;= 3</code>	<code>x = x &lt;&lt; 3</code>



# Java Operators

- Assignment operators

```
public class Main {  
    public static void main(String[] args) {  
        int x = 5;  
        x |= 3;  
        System.out.println(x);  
    }  
}
```

7



# Java Operators

- Assignment operators

```
public class Main {  
    public static void main(String[] args) {  
        int x = 5;  
        System.out.println(x);  
    }  
}
```





# Java Operators

- Assignment operators

```
public class Main {  
    public static void main(String[] args) {  
        int x = 5;  
        x += 3;  
        System.out.println(x);  
    }  
}
```

8



# Java Operators

- Assignment operators

```
public class Main {  
    public static void main(String[] args) {  
        int x = 5;  
        x >>= 3;  
        System.out.println(x);  
    }  
}
```

0



# Java Operators

- Assignment operators

```
public class Main {  
    public static void main(String[] args) {  
        int x = 5;  
        x <= 3;  
        System.out.println(x);  
    }  
}
```

40



# Java Operators

- **Comparison operators**

Comparison operators are used to compare two values (or variables). This is important in programming, because it helps us to find answers and make decisions.

The return value of a comparison is either **true** or **false**. These values are known as *Boolean values*, and you will learn more about them in the [Booleans](#) and [If..Else](#) chapter.

In the following example, we use the **greater than** operator (**>**) to find out if 5 is greater than 3:



# Java Operators

- Comparison operators

Operator	Name	Example
<code>==</code>	Equal to	<code>x == y</code>
<code>!=</code>	Not equal	<code>x != y</code>
<code>&gt;</code>	Greater than	<code>x &gt; y</code>
<code>&lt;</code>	Less than	<code>x &lt; y</code>
<code>&gt;=</code>	Greater than or equal to	<code>x &gt;= y</code>
<code>&lt;=</code>	Less than or equal to	<code>x &lt;= y</code>



# Java Operators

- Comparison operators

```
public class Main
{
    public static void main(String[] args)
    {
        int x = 5;
        int y = 3;
        System.out.println(x > y); // returns true, because 5
is higher than 3
    }
}
```

true



# Java Operators

- Comparison operators

```
public class Main {  
    public static void main(String[] args) {  
        int x = 5;  
        int y = 3;  
        System.out.println(x == y); // returns false because 5  
is not equal to 3  
    }  
}
```

false



# Java Operators

- Comparison operators

```
public class Main {  
    public static void main(String[] args) {  
        int x = 5;  
        int y = 3;  
        System.out.println(x >= y); // returns true because 5  
        is greater, or equal, to 3  
    }  
}
```

true



# Java Operators

- Comparison operators

```
public class Main {  
    public static void main(String[] args) {  
        int x = 5;  
        int y = 3;  
        System.out.println(x != y); // returns true because 5  
is not equal to 3  
    }  
}
```

true



# Java Operators

- Logical operators

You can also test for `true` or `false` values with logical operators.

Logical operators are used to determine the logic between variables or values:



# Java Operators

- Logical operators

Operator	Name	Description	Example
<code>&amp;&amp;</code>	Logical and	Returns true if both statements are true	<code>x &lt; 5 &amp;&amp; x &lt; 10</code>
<code>  </code>	Logical or	Returns true if one of the statements is true	<code>x &lt; 5    x &lt; 4</code>
<code>!</code>	Logical not	Reverse the result, returns false if the result is true	<code>!(x &lt; 5 &amp;&amp; x &lt; 10)</code>



# Java Operators

- Bitwise operators

```
public class Main {  
    public static void main(String[] args) {  
        int x = 5;  
        System.out.println(x > 3 && x < 10); // returns true  
because 5 is greater than 3 AND 5 is less than 10  
    }  
}
```

true



# Java Operators

- Bitwise operators

```
public class Main {  
    public static void main(String[] args) {  
        int x = 5;  
        System.out.println(!(x > 3 && x < 10)); // returns  
false because ! (not) is used to reverse the result  
    }  
}
```

false



# Java Operators

- Bitwise operators

```
public class Main {  
    public static void main(String[] args) {  
        int x = 5;  
        System.out.println(!(x > 3 && x < 10)); // returns  
false because ! (not) is used to reverse the result  
    }  
}
```

false



# Java Comments

Comments can be used to explain Java code, and to make it more readable. It can also be used to prevent execution when testing alternative code.

Single-line comments start with two forward slashes (//).

```
public class Main {  
    public static void main(String[] args) {  
        // This is a comment  
        System.out.println("Hello World");  
    }  
}
```

Hello World



# Java Comments

Comments can be used to explain Java code, and to make it more readable. It can also be used to prevent execution when testing alternative code.

**Multi-line comments start with `/*` and ends with `*/`.**

```
public class Main {  
    public static void main(String[] args) {  
        /* The code below will print the words  
Hello World  
        to the screen, and it is amazing */  
        System.out.println("Hello World");  
    }  
}
```

Hello World



# Java Control Statements

Java compiler executes the code from top to bottom. The statements in the code are executed according to the order in which they appear. However, [Java](#) provides statements that can be used to control the flow of Java code. Such statements are called control flow statements. It is one of the fundamental features of Java, which provides a smooth flow of program.



# Java if...else Statement

## Condition is true

```
int number = 10;  
  
if (number > 0) {  
    // code  
}  
  
// code after if
```

## Condition is false

```
int number = 10;  
  
if (number < 0) {  
    // code  
}  
  
// code after if
```



# Java if...else Statement

## Condition is true

```
int number = 5;  
  
if (number > 0) {  
    // code  
}  
else {  
    // code  
}  
  
// code after if...else
```

## Condition is false

```
int number = 5;  
  
if (number < 0) {  
    // code  
}  
else {  
    // code  
}  
  
// code after if...else
```



# Java if...else Statement

## 1st Condition is true

```
int number = 2;  
if (number > 0) {  
    // code  
}  
else if (number == 0){  
    // code  
}  
else {  
    //code  
}  
  
//code after if
```

## 2nd Condition is true

```
int number = 0;  
if (number > 0) {  
    // code  
}  
else if (number == 0){  
    // code  
}  
else {  
    //code  
}  
  
//code after if
```

## All Conditions are false

```
int number = -2;  
if (number > 0) {  
    // code  
}  
else if (number == 0){  
    // code  
}  
else {  
    //code  
}  
  
//code after if
```



# Java switch Statement

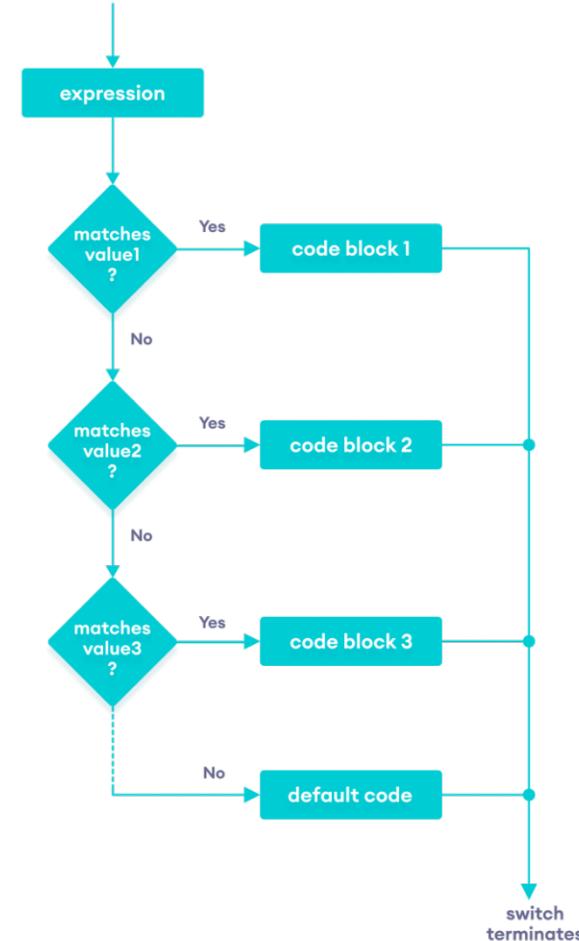
\*Example.java ×

```
1 package Examples;
2 public class Example{
3     public static void main(String[] args)
4     {
5         int num=23;
6         switch(num){
7             case 1: System.out.println("15");
8                 break;
9             case 2: System.out.println("25");
10                break;
11            case 3: System.out.println("35");
12                break;
13            default:System.out.println("Not Exist");
14
15        }
16    }
17 }
```

Console ×

```
<terminated> Example [Java Application] C:\Users\user\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.
Not Exist
```

Code → Output





# Java switch Statement

```
public class Main {  
    public static void main(String[] args) {  
        if (20 > 18) {  
            System.out.println("20 is greater than  
18"); // obviously  
        }  
    }  
}
```

20 is greater than 18



# Java switch Statement

```
public class Main {  
    public static void main(String[] args) {  
        int day = 4;  
        switch (day) {  
            case 1:  
                System.out.println("Monday");  
                break;  
            case 2:  
            case 3:  
            case 4:  
                System.out.println("Thursday");  
            case 7:  
                System.out.println("Sunday");  
                break;  
        }  
    }  
}
```

Thursday  
Sunday



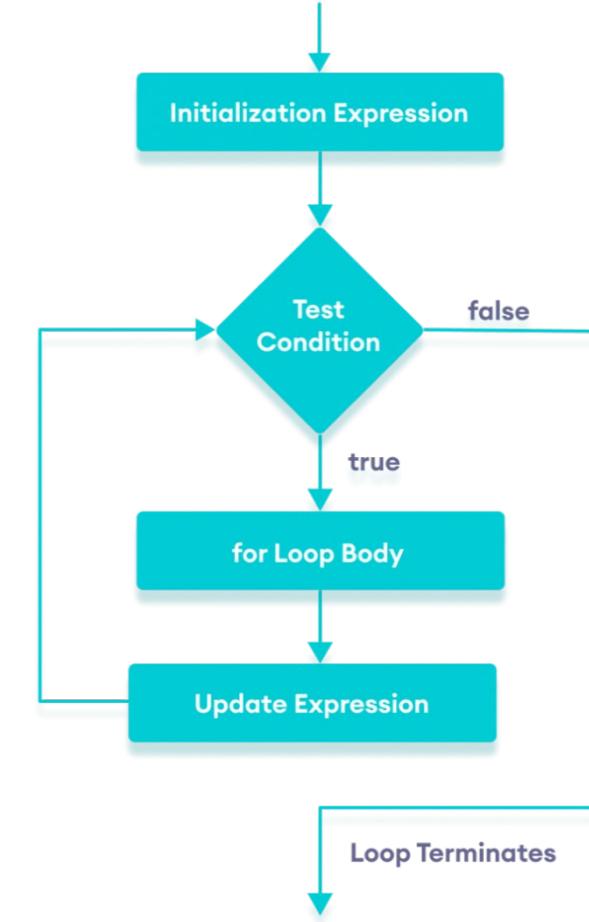
# Java for Loop

```
for (int i=0; i<10; i++)
```

Initialization

Condition

Iteration





# Java for Loop

```
public class Main {  
    public static void main(String[] args) {  
        for (int i = 0; i < 5; i++) {  
            System.out.println(i);  
        }  
    }  
}
```

0  
1  
2  
3  
4

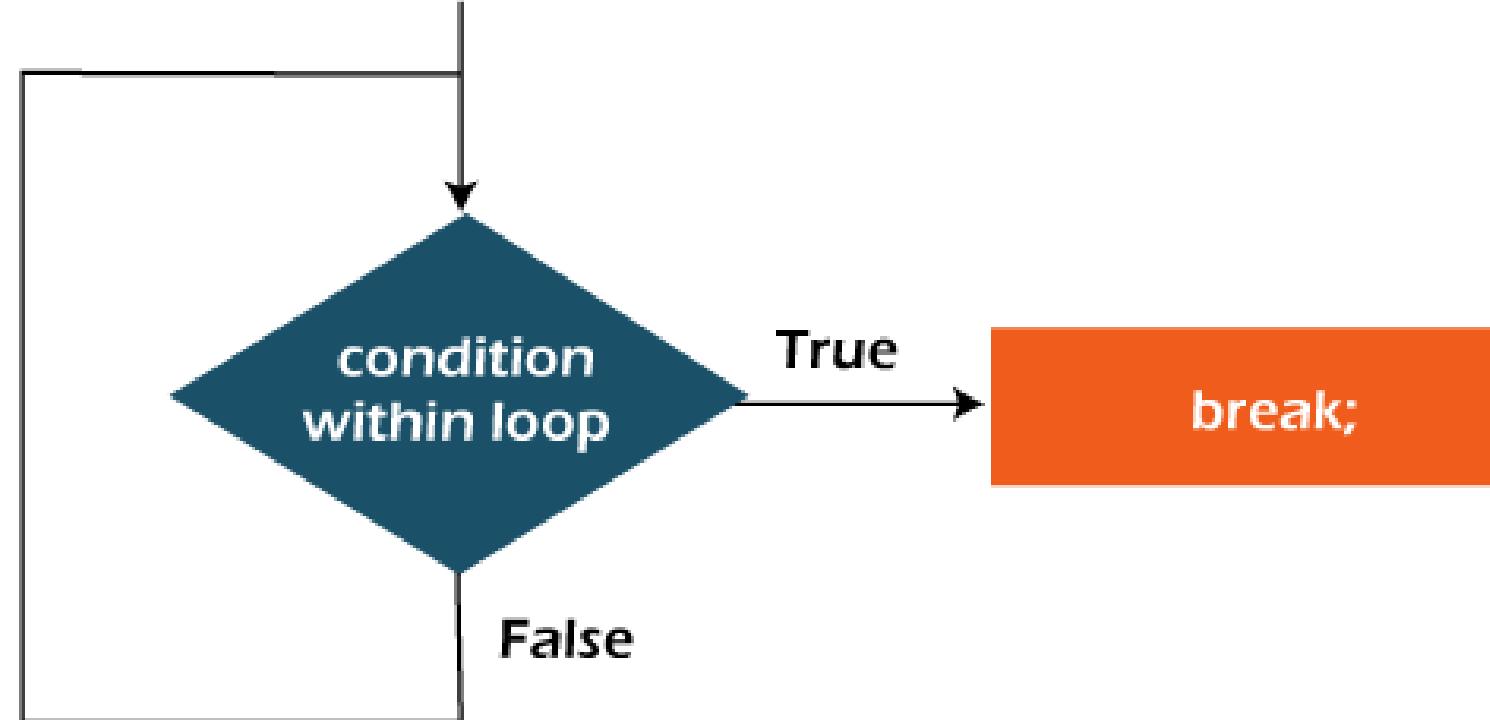


# Java Break Statement

- When a **break** statement is encountered inside a loop, the loop is immediately terminated and the program control resumes at the next statement following the loop.
- The Java *break* statement is used to break loop or [switch](#) statement. It breaks the current flow of the program at specified condition. In case of inner loop, it breaks only inner loop.
- We can use Java break statement in all types of loops such as [for loop](#), [while loop](#) and [do-while loop](#).



# Java Break Statement



Flowchart of break statement



# Java Break Statement

```
public class Main {  
    public static void main(String[] args) {  
        for (int i = 0; i < 10; i++) {  
            if (i == 4) {  
                break;  
            }  
            System.out.println(i);  
        }  
    }  
}
```

0  
1  
2  
3



# Java Access Modifier

Modifier	Description
public	The code is accessible for all classes
private	The code is only accessible within the declared class
<i>default</i>	The code is only accessible in the same package. This is used when you don't specify a modifier. You will learn more about packages in the <a href="#">Packages chapter</a>
protected	The code is accessible in the same package and <b>subclasses</b> . You will learn more about subclasses and superclasses in the <a href="#">Inheritance chapter</a>



# Java Access Modifier

Main.java

Second.java

```
public class Main {  
    public String fname = "John";  
    public String lname = "Doe";  
    public String email = "john@doe.com";  
    public int age = 24;  
}
```



# Java Access Modifier

Main.java

Second.java

```
class Second {  
    public static void main(String[] args) {  
        Main myObj = new Main();  
        System.out.println("Name: " +  
myObj.fname + " " + myObj.lname);  
        System.out.println("Email: " +  
myObj.email);  
        System.out.println("Age: " + myObj.age);  
    }  
}
```

Name: John Doe

Email: john@doe.com

Age: 24



# Java Access Modifier

```
public class Main {  
    private String fname = "John";  
    private String lname = "Doe";  
    private String email = "john@doe.com";  
    private int age = 24;  
  
    public static void main(String[] args) {  
        Main myObj = new Main();  
        System.out.println("Name: " + myObj.fname + " " +  
myObj.lname);  
        System.out.println("Email: " + myObj.email);  
        System.out.println("Age: " + myObj.age);  
    }  
}
```

Name: John Doe  
Email: john@doe.com  
Age: 24



# Java Access Modifier

```
class Person {  
    protected String fname = "John";  
    protected String lname = "Doe";  
    protected String email = "john@doe.com";  
    protected int age = 24;  
}  
  
class Student extends Person {  
    private int graduationYear = 2018;  
    public static void main(String[] args) {  
        Student myObj = new Student();  
        System.out.println("Name: " + myObj.fname + " " +  
myObj.lname);  
        System.out.println("Email: " + myObj.email);  
        System.out.println("Age: " + myObj.age);  
        System.out.println("Graduation Year: " +  
myObj.graduationYear);  
    }  
}
```

Name: John Doe  
Email: john@doe.com  
Age: 24  
Graduation Year: 2018



# Java Methods

- A **method** is a block of code which only runs when it is called.
- You can pass data, known as parameters, into a method.
- Methods are used to perform certain actions, and they are also known as **functions**.
- Why use methods? To reuse code: define the code once, and use it many times.



# Java Methods

## Create a Method

A method must be declared within a class. It is defined with the name of the method, followed by parentheses (). Java provides some pre-defined methods, such as `System.out.println()`, but you can also create your own methods to perform certain actions:



# Java Methods

Create a method inside Main:

```
public class Main {  
    static void myMethod() {  
        // code to be executed  
    }  
}
```



# Java Methods

## Example Explained

- `myMethod()` is the name of the method
- `static` means that the method belongs to the Main class and not an object of the Main class. You will learn more about objects and how to access methods through objects later in this tutorial.
- `void` means that this method does not have a return value. You will learn more about return values later in this chapter



# Java Methods

```
public class Main {  
    static void myMethod() {  
        System.out.println("I just got executed!");  
    }  
  
    public static void main(String[] args) {  
        myMethod();  
    }  
}
```

I just got executed!



# Java Methods

```
public class Main {  
    static void myMethod() {  
        System.out.println("I just got executed!");  
    }  
  
    public static void main(String[] args) {  
        myMethod();  
        myMethod();  
        myMethod();  
    }  
}
```

I just got executed!  
I just got executed!  
I just got executed!



# Java Method Parameters

- Information can be passed to methods as parameter. Parameters act as variables inside the method.
- Parameters are specified after the method name, inside the parentheses. You can add as many parameters as you want, just separate them with a comma.

The following example has a method that takes a **String** called **fname** as parameter. When the method is called, we pass along a first name, which is used inside the method to print the full name:



# Java Method Parameters

```
public class Main {  
    static void myMethod(String fname) {  
        System.out.println(fname + " Refsnes");  
    }  
  
    public static void main(String[] args) {  
        myMethod("Liam");  
        myMethod("Jenny");  
        myMethod("Anja");  
    }  
}
```

Liam Refsnes  
Jenny Refsnes  
Anja Refsnes



# Java Method Parameters

When a **parameter** is passed to the method, it is called an **argument**. So, from the example above: `fname` is a **parameter**, while `Liam`, `Jenny` and `Anja` are **arguments**.

- Multiple Parameters
- Note that when you are working with multiple parameters, the method call must have the same number of arguments as there are parameters, and the arguments must be passed in the same order.



# Java Method Parameters

## Return Values

The `void` keyword, used in the examples above, indicates that the method should not return a value. If you want the method to return a value, you can use a primitive data type (such as `int`, `char`, etc.) instead of `void`, and use the `return` keyword inside the method:



# Java Method Parameters

8

```
public class Main {  
    static int myMethod(int x) {  
        return 5 + x;  
    }  
  
    public static void main(String[] args) {  
        System.out.println(myMethod(3));  
    }  
}
```



# Java Scope

In Java, variables are only accessible inside the region they are created. This is called **scope**.

- **Method Scope**
- Variables declared directly inside a method are available anywhere in the method following the line of code in which they were declared:



# Java Scope

```
public class Main {  
    public static void main(String[] args) {  
  
        // Code here cannot use x  
  
        int x = 100;  
  
        // Code here can use x  
        System.out.println(x);  
    }  
}
```

100



# Java Scope

In Java, variables are only accessible inside the region they are created. This is called **scope**.

## Block Scope

A block of code refers to all of the code between curly braces {}.

Variables declared inside blocks of code are only accessible by the code between the curly braces, which follows the line in which the variable was declared:



# Java Scope

```
public class Main {  
    public static void main(String[] args)  
    {  
        {  
            int x = 100;  
            System.out.println(x);  
        }  
    }  
}
```

100



# Java Scope

A block of code may exist on its own or it can belong to an `if`, `while` or `for` statement. In the case of `for` statements, variables declared in the statement itself are also available inside the block's scope.