Here's a **clear step-by-step guide to run a Java file on a Windows PC**

□ 1. Install Java (JDK)

You need the **Java Development Kit (JDK)**.

Steps:

1. Go to the official site of Oracle Corporation
2. Download **JDK (latest version)** for Windows
3. Install it (just click *Next* → *Next* → *Finish*)

□ 2. Set Environment Variables (Important)

Steps:

1. Search **"Environment Variables"** in Windows
2. Click **Edit the system environment variables**
3. Click **Environment Variables**

Add this:

- Under **System Variables → Path → Edit**
- Add:

```
C:\Program Files\Java\jdk-XX\bin
```

□ Replace `XX` with your installed version

□ 3. Verify Installation

Open **Command Prompt (cmd)** and type:

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

□ If installed correctly, it will show version numbers.

```
:\Users\Big Data>java --version
penjdk 25.0.2 2026-01-20 LTS
penJDK Runtime Environment Temurin-25.0.2+10 (build 25.0.2+10-LTS)
penJDK 64-Bit Server VM Temurin-25.0.2+10 (build 25.0.2+10-LTS, mixed mode, sharing)

:\Users\Big Data>where java
:\Program Files\Eclipse Adoptium\jdk-25.0.2.10-hotspot\bin\java.exe

:\Users\Big Data>echo %JAVA_HOME%
:\Program Files\Eclipse Adoptium\jdk-25.0.2.10-hotspot

:\Users\Big Data>d:
```

□ 4. Create a Java File

1. Open Notepad
2. Write this code:

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

3. Save file as:

```
Main.java (File name must match class name )
```

□ Make sure:

- File name = class name
- Extension = `.java`

▶□ 5. Compile the Java File

1. Open Command Prompt
2. Go to the folder where file is saved:

```
cd Desktop
```

3. Compile:

```
javac Main.java
```

□ This creates:

Main.class

▶ 6. Run the Java Program

```
java Main
```

 Output:

```
Hello Java
```

Here is your **complete beginner-friendly Java guide with explanation**

□ □ 1. What is Java?

Java is a **high-level, object-oriented programming language** used to build:

- Apps □
- Websites □
- Software □

□ Key Features:

- **Platform Independent** → Runs anywhere (Windows, Mac, Linux)
- **Object-Oriented** → Uses classes and objects
- **Secure & Robust** → Safe and reliable

□ Think of Java like a **universal language for computers**

□ 2. Basic Java Program Structure

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

Explanation:

- `class Main` → Blueprint of program
- `main()` → Entry point (program starts here)
- `System.out.println()` → Prints output on screen

Output:

```
Hello, World!
```

□ 3. Data Types in Java

Java stores different types of data using data types.

Primitive Data Types

Type	Example	Description
int	10	Whole numbers
float	10.5f	Decimal numbers
double	10.55	More precise decimals
char	'A'	Single character
boolean	true/false	Yes/No values

Code:

```
public class Main {
    public static void main(String[] args) {

        int age = 20;
        float price = 99.99f;
        double pi = 3.14159;
        char grade = 'A';
        boolean isJavaFun = true;

        System.out.println(age);
        System.out.println(price);
        System.out.println(pi);
        System.out.println(grade);
        System.out.println(isJavaFun);
    }
}
```

Output:

```
20
99.99
3.14159
```

```
A
true
```

□ 4. Operators in Java

Operators perform calculations and comparisons.

Arithmetic Operators

```
public class Main {
    public static void main(String[] args) {

        int a = 10, b = 5;

        System.out.println(a + b); // 15
        System.out.println(a - b); // 5
        System.out.println(a * b); // 50
        System.out.println(a / b); // 2
    }
}
```

□ Output:

```
15
5
50
2
```

Comparison Operators

```
System.out.println(a > b);   // true
System.out.println(a == b);  // false
```

□ Output:

```
true
false
```

Logical Operators

```
System.out.println(a > 5 && b < 10);
```

□ Explanation:

- `&&` means BOTH conditions must be true

□ Output:

```
true
```

□ 5. if, else if, else

Used for decision making.

```
public class Main {
    public static void main(String[] args) {

        int number = 10;

        if (number > 0) {
            System.out.println("Positive");
        } else if (number == 0) {
            System.out.println("Zero");
        } else {
            System.out.println("Negative");
        }
    }
}
```

Output:

```
Positive
```

□ 6. for Loop

Used when you know how many times to repeat.

```
public class Main {
    public static void main(String[] args) {

        for (int i = 1; i <= 5; i++) {
            System.out.println(i);
        }
    }
}
```

Output:

```
1
2
3
4
5
```

□ 7. while Loop

Runs until condition becomes false.

```
public class Main {
    public static void main(String[] args) {

        int i = 1;

        while (i <= 5) {
            System.out.println(i);
            i++;
        }
    }
}
```

Output:

```
1
2
3
4
5
```

□ 8. Functions (Methods)

Functions are **reusable blocks of code**.

Example:

```
public class Main {

    public static void greet() {
        System.out.println("Hello!");
    }

    public static void main(String[] args) {
        greet();
    }
}
```

Explanation:

- `greet()` → function definition
- `greet();` → function call

Output:

```
Hello!
```

Function with Parameters

```
public class Main {

    public static void add(int a, int b) {
        System.out.println(a + b);
    }

    public static void main(String[] args) {
        add(5, 10);
    }
}
```

Output:

```
15
```

□ 9. Class and Object

Class = Blueprint

Object = Real-world instance

Class Example:

```
class Car {
    String color;
    int speed;

    void drive() {
        System.out.println("Car is driving");
    }
}
```

Creating Object:

```
public class Main {
    public static void main(String[] args) {

        Car myCar = new Car();

        myCar.color = "Red";
        myCar.speed = 100;

        System.out.println(myCar.color);
        myCar.drive();
    }
}
```

Explanation:

- `Car myCar = new Car();` → object creation
- `myCar.color` → accessing property
- `myCar.drive()` → calling method

Output:

```
Red
Car is driving
```

□ FINAL SUMMARY

Concept	Purpose
Data Types	Store data
Operators	Perform operations
if/else	Decision making
Loops	Repetition
Functions	Reusable code
Classes & Objects	Real-world modeling