Class 01

What is Java?

Java is a popular object-oriented programming language, created in 1995.

Now, it is owned by Oracle, originally developed by Sun Microsystems. More than 3 billion devices run Java.

It is used for:

- **Mobile applications (specially Android apps)**
- **Desktop applications**
- **Web applications**
- **Web servers and application servers**
- **Games**
- **Database connection**
- **And much, much more!**

Why Use Java?

- **Java works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc.)**
- **It is one of the most popular programming language in the world**
- **It has a large demand in the current job market**
- **It is easy to learn and simple to use**
- **It is open-source and free**
- **It is secure, fast and powerful**
- **It has a huge community support (tens of millions of developers)**
- **Java is an object oriented language which gives a clear structure to programs and allows code to be reused, lowering development costs**
- **As Java is close to C++ and C#, it makes it easy for programmers to switch to Java or vice versa**

Example: Basic Java Program

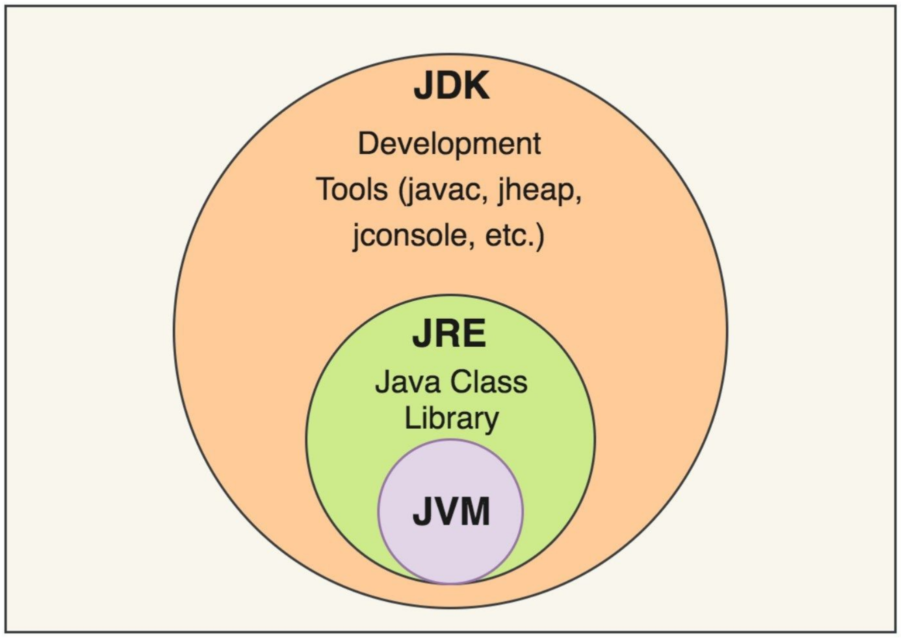
Main.java

```
public class Main {

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

-----------output: naztech

Java Runtime Environment consist with the following components:



JVM (Java Virtual Machine) is the runtime engine that executes Java bytecode, ensuring platform independence. It interprets or compiles bytecode into native machine code for execution, handling memory and threads.

JRE (Java Runtime Environment) includes the JVM, libraries, and other runtime components required to run Java applications. It's sufficient for running Java programs but doesn't include development tools.

JDK (Java Development Kit) is a comprehensive package containing the JRE, development tools, and libraries necessary for creating Java applications. It includes a compiler, debugger, and other utilities for coding, testing, and deploying Java programs. JDK is essential for both development and running Java applications.

Java Identifiers

Identifiers

All Java variables must be identified with unique names.

These unique names are called identifiers.

Identifiers can be short names (like x and y) or more descriptive names (age, sum, totalVolume).

Note: It is recommended to use descriptive names in order to create understandable and maintainable code:

Example

```
int minutesPerHour = 60;
```

What are Java Reserved Keywords?

Java has 51 reserved keywords. These keywords have specific meanings in the language and cannot be used for other purposes like naming variables or methods. Here's the list of Java keywords:

Keyword	**Description**
Abstract:	A non-access modifier. Used for classes and methods: An abstract class cannot be used to create objects (to access it, it must be inherited from another class). An abstract method can only be used in an abstract class, and it does not have a body. The body is provided by the subclass (inherited from)
Assert :	For debugging
Boolean :	A data type that can only store true and false values
Break:	Breaks out of a loop or a switch block
Byte:	A data type that can store whole numbers from -128 and 127
Case:	Marks a block of code in switch statements
Catch:	Catches exceptions generated by try statements
char	A data type that is used to store a single character
class	Defines a class
continue:	Continues to the next iteration of a loop
const:	Defines a constant. Not in use - use final instead
default:	Specifies the default block of code in a switch statement
do:	Used together with while to create a do-while loop
double :	A data type that can store whole numbers from 1.7e−308 to 1.7e+308
else:	Used in conditional statements
enum:	Declares an enumerated (unchangeable) type
exports :	Exports a package with a module. New in Java 9
extends :	Extends a class (indicates that a class is inherited from another class)

**final : ** A non-access modifier used for classes, attributes and methods, which makes them non-changeable (impossible to inherit or override)

finally : Used with exceptions, a block of code that will be executed no matter if there is an exception or not

float : A data type that can store whole numbers from 3.4e−038 to 3.4e+038

for : Create a for loop

goto : Not in use, and has no function

if : Makes a conditional statement

implements : Implements an interface

import : Used to import a package, class or interface

instanceof : Checks whether an object is an instance of a specific class or an interface

int: A data type that can store whole numbers from -2147483648 to 2147483647

interface : Used to declare a special type of class that only contains abstract methods

long : A data type that can store whole numbers from -9223372036854775808 to 9223372036854775808

module : Declares a module. New in Java 9

native : Specifies that a method is not implemented in the same Java source file (but in another language)

new : Creates new objects

package : Declares a package

private An access modifier used for attributes, methods and constructors, making them only accessible within the declared class

protected: An access modifier used for attributes, methods and constructors, making them accessible in the same package and subclasses

public : An access modifier used for classes, attributes, methods and constructors, making them accessible by any other class

requires : Specifies required libraries inside a module. New in Java 9

return : Finished the execution of a method, and can be used to return a value from a method

short: A data type that can store whole numbers from -32768 to 32767

static : A non-access modifier used for methods and attributes. Static methods/attributes can be accessed without creating an object of a class

strictfp **:** Restrict the precision and rounding of floating point calculations

super : Refers to superclass (parent) objects

switch **:** Selects one of many code blocks to be executed

synchronized : A non-access modifier, which specifies that methods can only be accessed by one thread at a time

this: Refers to the current object in a method or constructor

throw : Creates a custom error

throws : Indicates what exceptions may be thrown by a method

transient : A non-accesss modifier, which specifies that an attribute is not part of an object's persistent state

try : Creates a try...catch statement

var : Declares a variable. New in Java 10

void : Specifies that a method should not have a return value

volatile : Indicates that an attribute is not cached thread-locally, and is always read from the "main memory"

while : Creates a while loop

Note: true, false, and null are not keywords, but they are literals and reserved words that cannot be used as identifiers.

What are the key **features of java?**

Key features of the Java programming language:

1. **Platform Independence:** Java is designed to be platform-independent, meaning that a Java program can run on any operating system with a compatible Java Virtual Machine (JVM).
2. **Object-Oriented:** Java follows the object-oriented programming (OOP) paradigm, allowing developers to model real-world entities as objects and create reusable and modular code.
3. **Simple and Easy to Learn:** Java has a clean and straightforward syntax, making it relatively easy for beginners to grasp the language's concepts.
4. **Automatic Memory Management:** Java manages memory automatically through its built-in garbage collection system, reducing the chances of memory leaks and providing memory efficiency.
5. **Strongly Typed:** Java enforces strong type-checking during both compile-time and runtime, enhancing program reliability and reducing errors.

6. **Platform-Independent Libraries:** Java comes with a vast standard library that provides pre-built classes and methods for various tasks, saving developers time and effort.
7. **Multithreading:** Java supports multithreading, enabling the creation of programs that can execute multiple tasks concurrently, improving efficiency and responsiveness.
8. **Exception Handling:** Java has a robust exception handling mechanism that helps manage and recover from runtime errors, ensuring stable and predictable program behavior.
9. **Security:** Java has built-in security features, including a security manager and sandbox environment, making it suitable for developing secure applications, especially in web and network environments.
10. **Rich API:** Java offers a wide range of APIs (Application Programming Interfaces) for tasks such as I/O, networking, databases, GUI development, and more.
11. **Distributed Computing:** Java supports network programming and communication through its rich libraries, making it ideal for building distributed and networked applications.
12. **Dynamic Loading:** Java allows classes to be loaded dynamically, which facilitates more flexible and adaptable program structures.
13. **Open Source Community:** Java has a vibrant open-source community that contributes to the development of various libraries, frameworks, and tools.
14. **Portability:** Java's "Write Once, Run Anywhere" philosophy ensures that code written in Java can be compiled once and executed on multiple platforms without modification.
15. **Continuous Evolution:** Java continues to evolve with regular updates and new features, ensuring that it remains relevant and competitive in the modern programming landscape.

These features collectively contribute to Java's popularity and worldwide acceptance as an object oriented programming language.