# JAVA INTRODUCTION

Instructor: DieuNT1

## Table of contents

Section 1

## Introduction to Java

## Introduction to Java

- **History:**
  - ✓ In 1991: OAK

  - ✓ A programming language that was introduced by Sun Microsystems in 1995, later acquired by **Oracle Corporation**.
    - Originally for intelligent consumer-electronic devices
    - Then used for creating Web pages with dynamic content

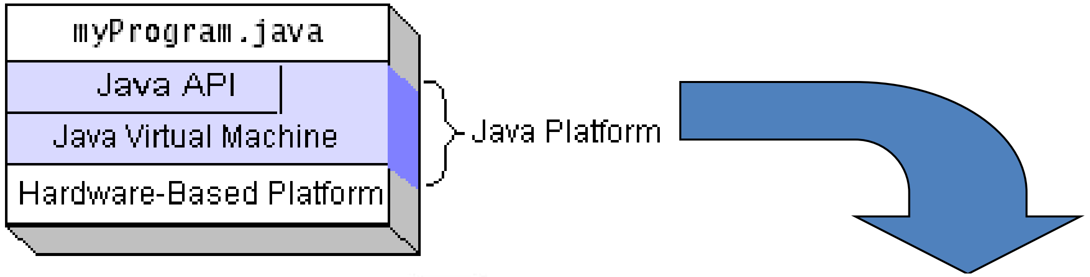
## Introduction to Java (2)

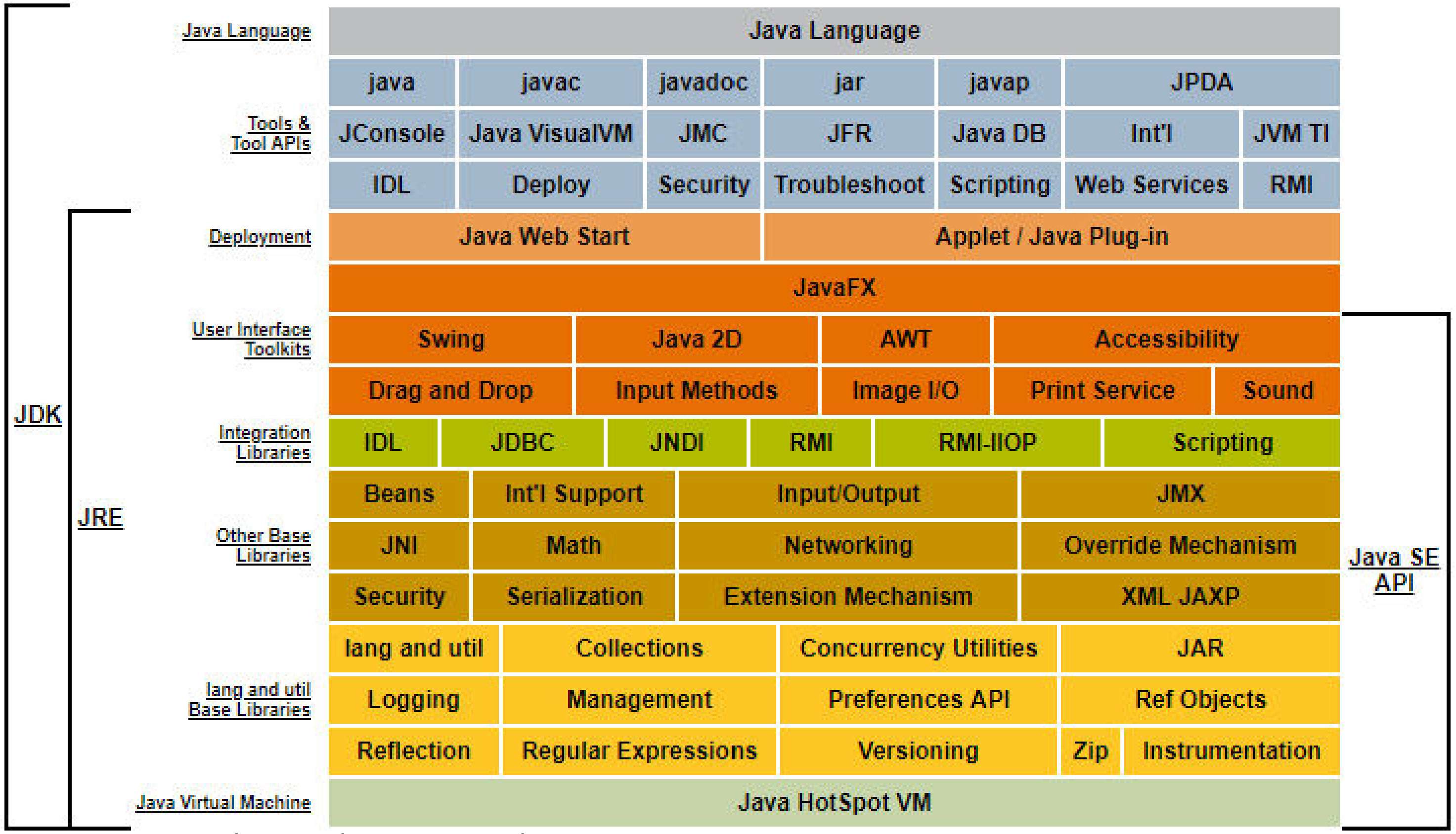
- **Now also used for:**
  - ✓ Develop large-scale enterprise applications
  - ✓ Enhance WWW server functionality
  - ✓ Provide applications for consumer[tiêu dùng] devices (cell phones, cloud, etc.)

## Main Features of JAVA

- The Java programming language is a high-level language that can be characterized by all of the following buzzwords:
  - ✓ **Simple**
  - ✓ **Object oriented**
  - ✓ **Distributed**
  - ✓ **Multithreaded**
  - ✓ **Dynamic**
  - ✓ **Architecture neutral**
  - ✓ **Portable**
  - ✓ **High performance**
  - ✓ **Robust**
  - ✓ **Secure**

# Java Platform





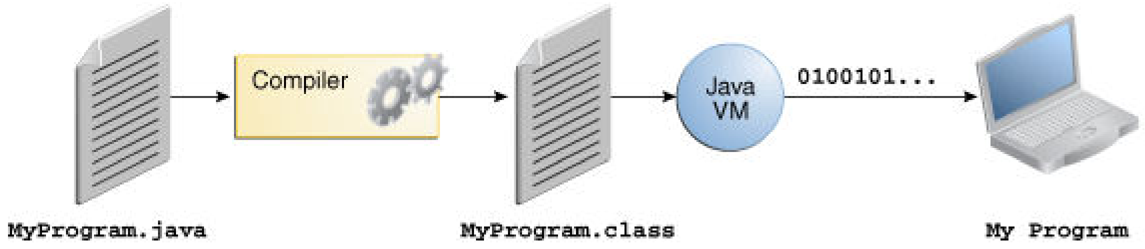
| | | | | | | | |
|---|---|---|---|---|---|---|---|
| Java Language | Java Language | | | | | | |
| Tools & Tool APIs | java | javac | javadoc | jar | javap | JPDA | |
| | JConsole | Java VisualVM | JMC | JFR | Java DB | Int'l | JVM TI |
| | IDL | Deploy | Security | Troubleshoot | Scripting | Web Services | RMI |
| Deployment | Java Web Start | Applet / Java Plug-in | | | | | |
| User Interface Toolkits | JavaFX | | | | | | |
| | Swing | Java 2D | AWT | Accessibility | | | |
| | Drag and Drop | Input Methods | Image I/O | Print Service | Sound | | |
| Integration Libraries | IDL | JDBC | JNDI | RMI | RMI-IIOP | Scripting | |
| Other Base Libraries | Beans | Int'l Support | Input/Output | JMX | | | |
| | JNI | Math | Networking | Override Mechanism | | | |
| | Security | Serialization | Extension Mechanism | XML JAXP | | | |
| lang and util Base Libraries | lang and util | Collections | Concurrency Utilities | JAR | | | |
| | Logging | Management | Preferences API | Ref Objects | | | |
| | Reflection | Regular Expressions | Versioning | Zip | Instrumentation | | |
| Java Virtual Machine | Java HotSpot VM | | | | | | |

## Java terminology

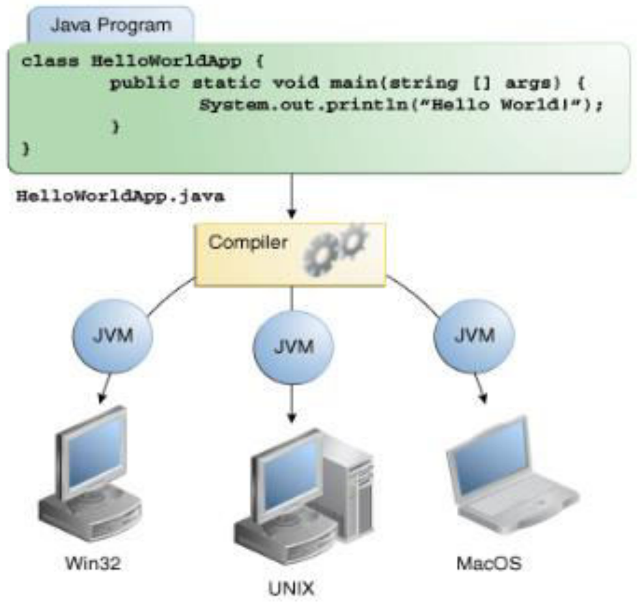
- **Java Development Kit(JDK)**
  - ✓ A complete java development kit that includes JRE (Java Runtime Environment), compilers and various tools like JavaDoc, Java debugger etc.
  - ✓ In order to create, compile and run Java program you would need JDK installed on your computer.
- **Java Runtime Environment(JRE)**
  - ✓ JRE is a part of JDK
  - ✓ When you have JRE installed on your system, you can run a java program however you won't be able to compile it.
  - ✓ JRE includes JVM, browser plugins and applets support. When you only need to run a java program on your computer, you would only need JRE.

## Java terminology

- **Java Virtual Machine (JVM)**



An overview of the software development process.



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

Through the Java VM, the same application is capable of running on multiple platforms.

Section 2

## First Java Program

## First Sample: Printing a Line of Text

```
//This is a simple program called First.java
public class First {
    public static void main(String[] args) {
    System.out.println("My first program in Java ");
    }
}
```

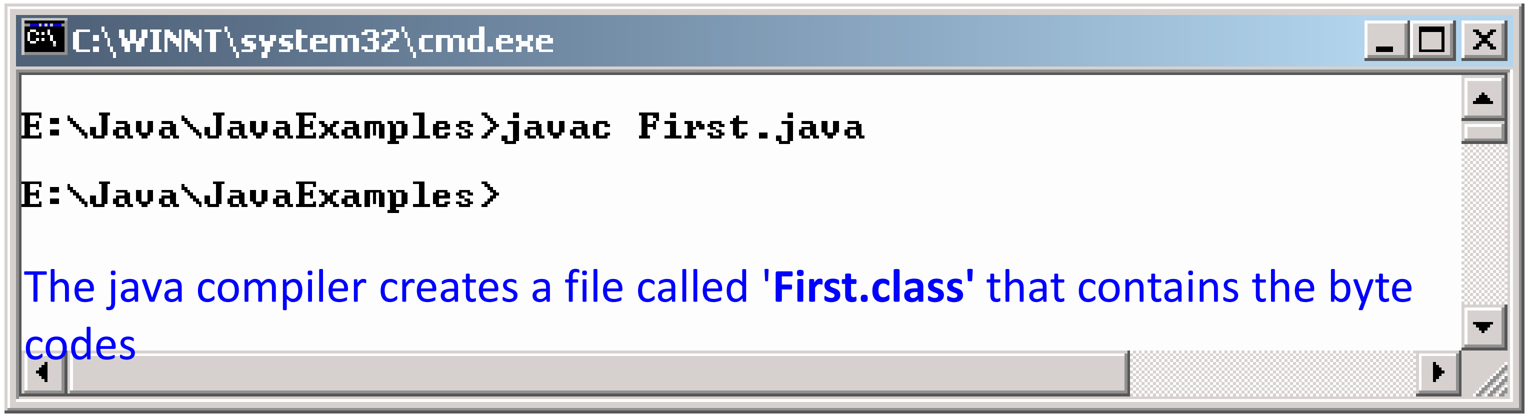
## First Sample: Analyzing the Java Program

- **In which:**
  - ✓ The symbol `//` stands for commented line.
  - ✓ The line `class First` declares a new class called `First`.
  - ✓ `public static void main(String[] args)`

    This is the main method from where the program begins its execution.
  - ✓ `System.out.println("My first program in Java ");`

    This line displays the string **My first program in java** on the screen.

# Compiling and executing

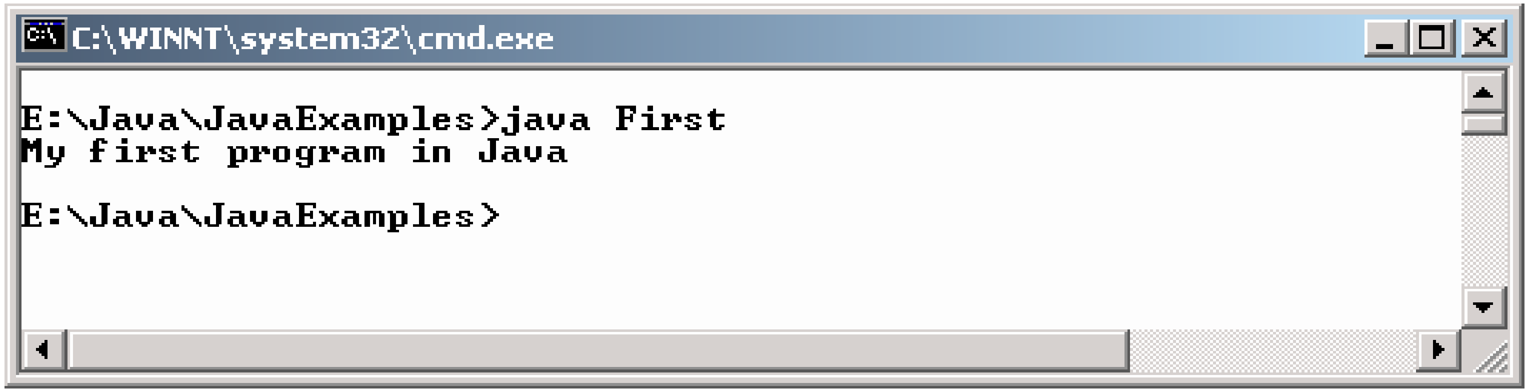


```
E:\Java\JavaExamples>javac First.java

E:\Java\JavaExamples>
```

The java compiler creates a file called '**First.class**' that contains the byte codes

To actually run the program, a java interpreter called `java` is required to execute the code.



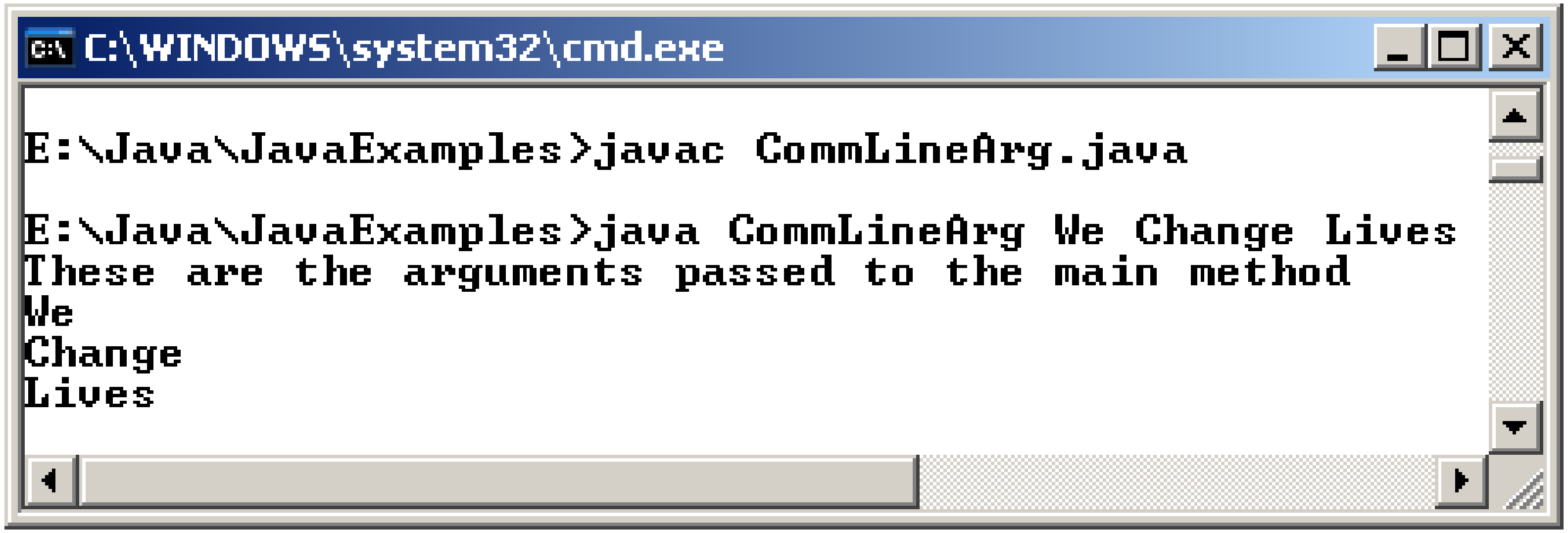
```
E:\Java\JavaExamples>java First
My first program in Java

E:\Java\JavaExamples>
```

## Passing Command Line Arguments

```
public class CommLineArg {
    public static void main(String[] pargs) {
        System.out.
        println("These are the arguments passed to the
          main method.");
        System.out.println(pargs[0]);
        System.out.println(pargs[1]);
        System.out.println(pargs[2]);
    }
}
```

# Passing Command Line Arguments



```
E:\Java\JavaExamples>javac CommLineArg.java

E:\Java\JavaExamples>java CommLineArg We Change Lives
These are the arguments passed to the main method
We
Change
Lives
```

Section 3

## Basic Java Syntax

## Code Comment

```
/*
 * Multi line
 */

// Single line

/**
 * Special comment for Javadocs
 */
```

# Name Styles

- In Java, names are case-insensitive, may contains **letter**, **number**, the dollar sign "**$**", or the underscore character "**_**".
- Some convention name styles:
  - ✓ Class names: `CustomerInfo`
  - ✓ Variable, function names: `basicAnnualSalary`
  - ✓ Constants name: `MAXIMUM_NUM_OF_PARTICIPANTS`

## Basic Data Types

- **byte**: The byte data type is an **8-bit** signed two's complement integer. It has a minimum value of -128 and a maximum value of 127 (inclusive).
- **short**: The short data type is a 16-bit signed two's complement integer. It has a minimum value of -32,768 and a maximum value of 32,767 (inclusive)
- **int**: The int data type is a 32-bit signed two's complement integer. It has a minimum value of -2,147,483,648 and a maximum value of 2,147,483,647 (inclusive).
- **long**: The long data type is a 64-bit signed two's complement integer. It has a minimum value of -9,223,372,036,854,775,808 and a maximum value of 9,223,372,036,854,775,807 (inclusive)

## Basic Data Types (2)

- **float**: The float data type is a single-precision 32-bit IEEE 754 floating point. Its range of values is from $3.4E^{-45}$ to $3.4E^{38}$
- **double**: The double data type is a double-precision 64-bit IEEE 754 floating point. Its range of values is from $1.7E^{-324}$ to $1.7976931348623157E^{308}$
- **boolean**: The boolean data type has only two possible values: true and false. Use this data type for simple flags that track true/false conditions. This data type represents one bit of information, but its "size" isn't something that's precisely defined.
- **char**: The char data type is a single 16-bit Unicode character. It has a minimum value of '\u0000' (or 0) and a maximum value of '\uffff' (or 65,535 inclusive).

# Basic Data Types (3)

- **Default Values**
  - ✓ It's not always necessary to assign a value when a field is declared
  - ✓ Fields that are declared but not initialized will be set to a reasonable default by the compiler
  - ✓ Generally speaking, this default will be **zero** or **null**, depending on the data type. However, **is generally considered bad programming style**.

| Data Type | Default Value (for fields) |
|---|---|
| byte | 0 |
| short | 0 |
| int | 0 |
| long | 0L |
| float | 0.0f |
| double | 0.0d |
| char | '\u0000' |
| String (or any object) | null |
| boolean | false |

Section 4

## Operators

# Operators

- **Simple Assignment Operator**

| | |
|---|---|
| = | Simple assignment operator |

- **Arithmetic Operators**

| | |
|---|---|
| + | Additive operator |
| - | Subtraction operator |
| * | Multiplication operator |
| / | Division operator |
| % | Remainder operator |

- **Unary Operators**

| | |
|---|---|
| + | Unary plus operator; indicates positive value |
| - | Unary minus operator; negates an expression |
| ++ | Increment operator; increments a value by 1 |
| -- | Decrement operator; decrements a value by 1 |
| ! | Logical compliment operator; inverts the value of a boolean |

# Operators

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

    double number1 = 12.5, number2 = 3.5, result;

    // Using addition operator
    result = number1 + number2;
    System.out.println("number1 + number2 = " + result);

    // Using subtraction operator
    result = number1 - number2;
    System.out.println("number1 - number2 = " + result);

    // Using multiplication operator
    result = number1 * number2;
    System.out.println("number1 * number2 = " + result);

    // Using division operator
    result = number1 / number2;
    System.out.println("number1 / number2 = " + result);

    // Using remainder operator
    result = number1 % number2;
    System.out.println("number1 % number2 = " + result);
  }
}
```

**Output:**

```
number1 + number2 = 16.0

number1 - number2 = 9.0

number1 * number2 = 43.75

number1 / number2 = 3.5714285714285716

number1 % number2 = 2.0
```

# Operators

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

    double number = 5.2;
    boolean flag = false;

    System.out.println("+number = " + +number);
    // number is equal to 5.2 here.

    System.out.println("-number = " + -number);
    // number is equal to 5.2 here.

    // ++number is equivalent to number = number + 1
    System.out.println("number = " + ++number);
    // number is equal to 6.2 here.

    // -- number is equivalent to number = number - 1
    System.out.println("number = " + --number);
    // number is equal to 5.2 here.

    System.out.println("!flag = " + !flag);
    // flag is still false.
  }
}
```

**Output:**

```
+number = 5.2
-number = -5.2
number = 6.2
number = 5.2
!flag = true
```

## Operators

- **Equality and Relational Operators**

| | |
|---|---|
| == | Equal to |
| != | Not equal to |
| > | Greater than |
| >= | Greater than or equal to |
| < | Less than |
| <= | Less than or equal to |

- **Conditional Operators**

| | |
|---|---|
| && | Conditional-AND |
| \|\| | Conditional-OR |
| ?: | Ternary (shorthand for if-then-else statement) |

- **Type Comparison Operator**

| | |
|---|---|
| instanceof | Compares an object to a specified type |

# Type Casting

- In type casting, a data type is converted into another data type.
- **Automatic Type Promotion in Expressions**
- **Example:**

```
public class AutomaticTypePromotion {
    public static void main(String[] argv) {
        byte a = 40;
        byte b = 50;
        byte c = 100;
        int d = a * b / c;
        b = b * 2; // Error! Cannot assign an int to a byte!
        System.out.println("Value d: " + d);
    }
}
```

Type Promotion in Arithmetic Operations: Java automatically promotes the smaller integral types (byte, short, and char) to int when used in arithmetic expressions

b * 2 trả về 1 int

## Type Casting

- **Type casting in Expressions**

  Casting is used for explicit type conversion. It loses information above the magnitude of the value being converted

- **Example:**

```
float f = 34.89675f;
d = (int) (f + 10);
```

## Type Casting

- **Widening**[an toàn/mở rộng]**conversions:**

  char->int

  byte->short->int->long->float->double

- **Here are the Type Promotion Rules**

  - ✓ All `byte` and `short` values are promoted to `int` type.
  - ✓ If one operand is `long`, the whole expression is promoted to `long`.
  - ✓ If one operand is `float` then the whole expression is promoted to `float`.
  - ✓ If one operand is `double` then the whole expression is promoted to `double`.

Section 5

## Variable and Constant

## Variables and constants

- **Variable:**
- Three components of a variable declaration are:
  - ✓ Data type
  - ✓ Name
  - ✓ Initial value to be assigned (optional)
- **Syntax**

```
datatype identifier [=value][, identifier[=value]...];
```

- **Example:**

```
int foo = 42;
double d1 = 3.14, d2 = 2 * 3.14;
boolean isFun = true;
```

## Variables and constants

- **Constants:**
  - ✓ It makes code more readable
  - ✓ It saves work when you make a change
  - ✓ You avoid risky[rủi ro] errors
  - ✓ In the case of string text
- **Syntax**

```
static final datatype CONSTNAME = value;
```

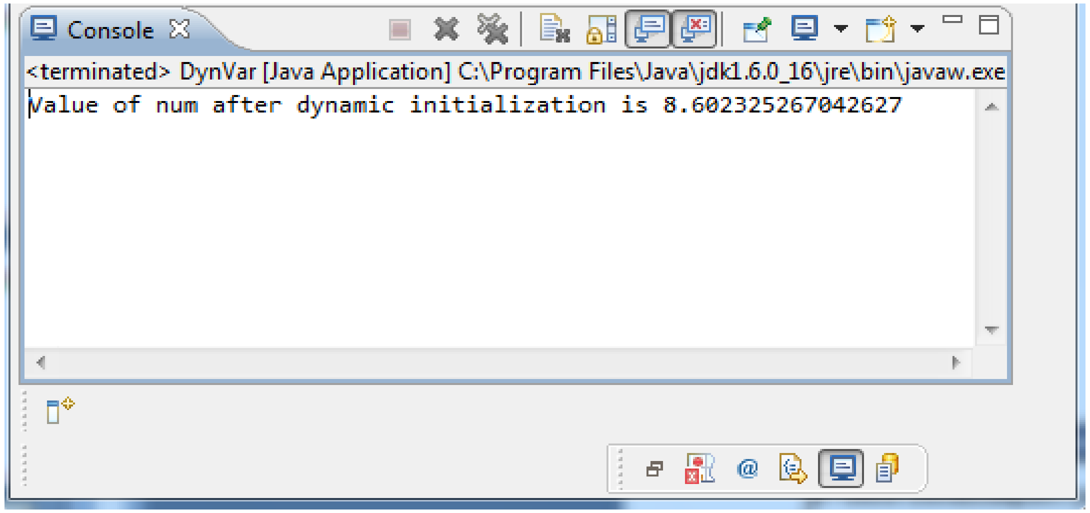
- **Example:**

```
static final int MAX_SECONDS = 25;
static final float PI = 3.14f;
```

## Variables and constants

- **Example:**

```
public class DynVar {
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        double len = 5.0, wide = 7.0;
        double num = Math.sqrt(len * len + wide * wide);
        System.out.println("Value of num after dynamic
            initialization is " + num);
    }
}
```



```
<terminated> DynVar [Java Application] C:\Program Files\Java\jdk1.6.0_16\jre\bin\javaw.exe
Value of num after dynamic initialization is 8.602325267042627
```

# Thank you