Website

https://www.curiousinfotechsolutions.com

Day-1

About Basics of Java
What is Java, Why Use Java

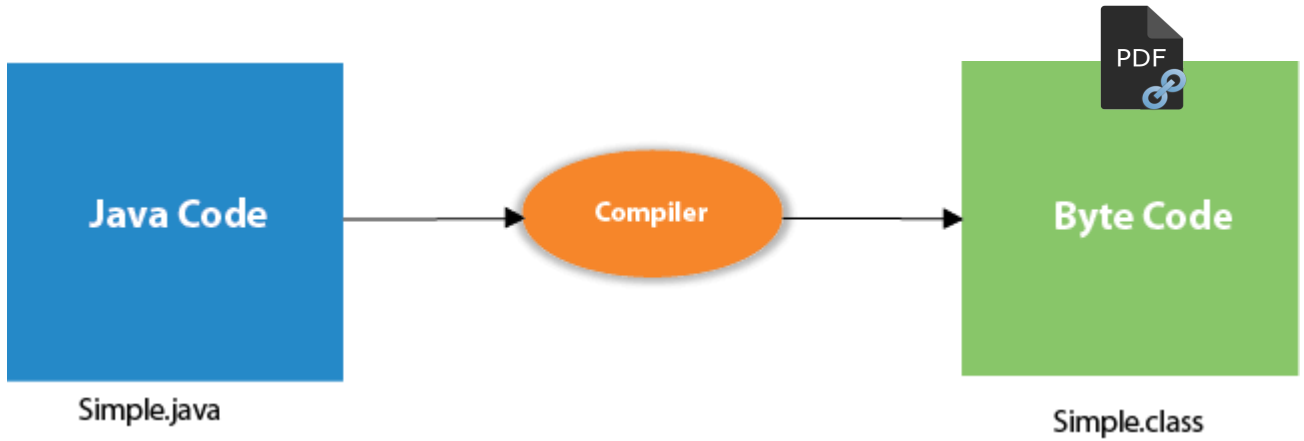
Java Programming

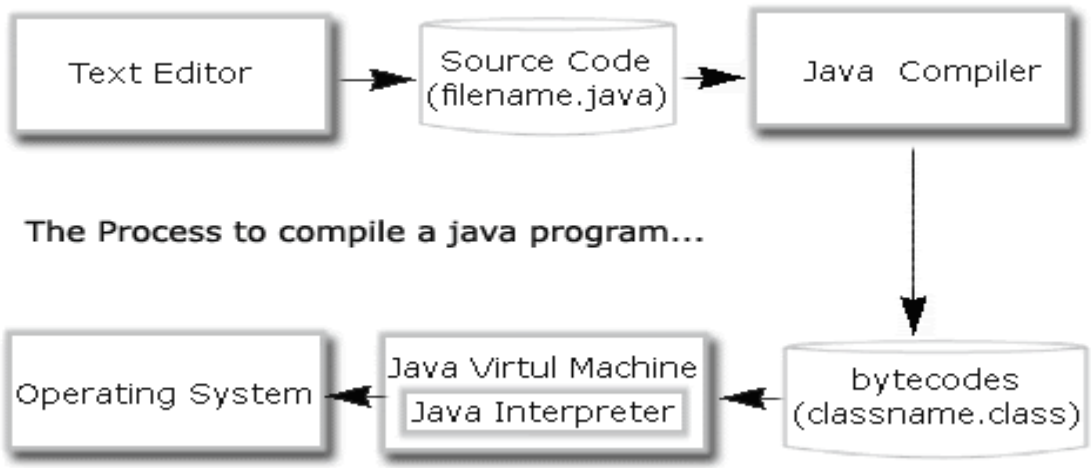
Day-2

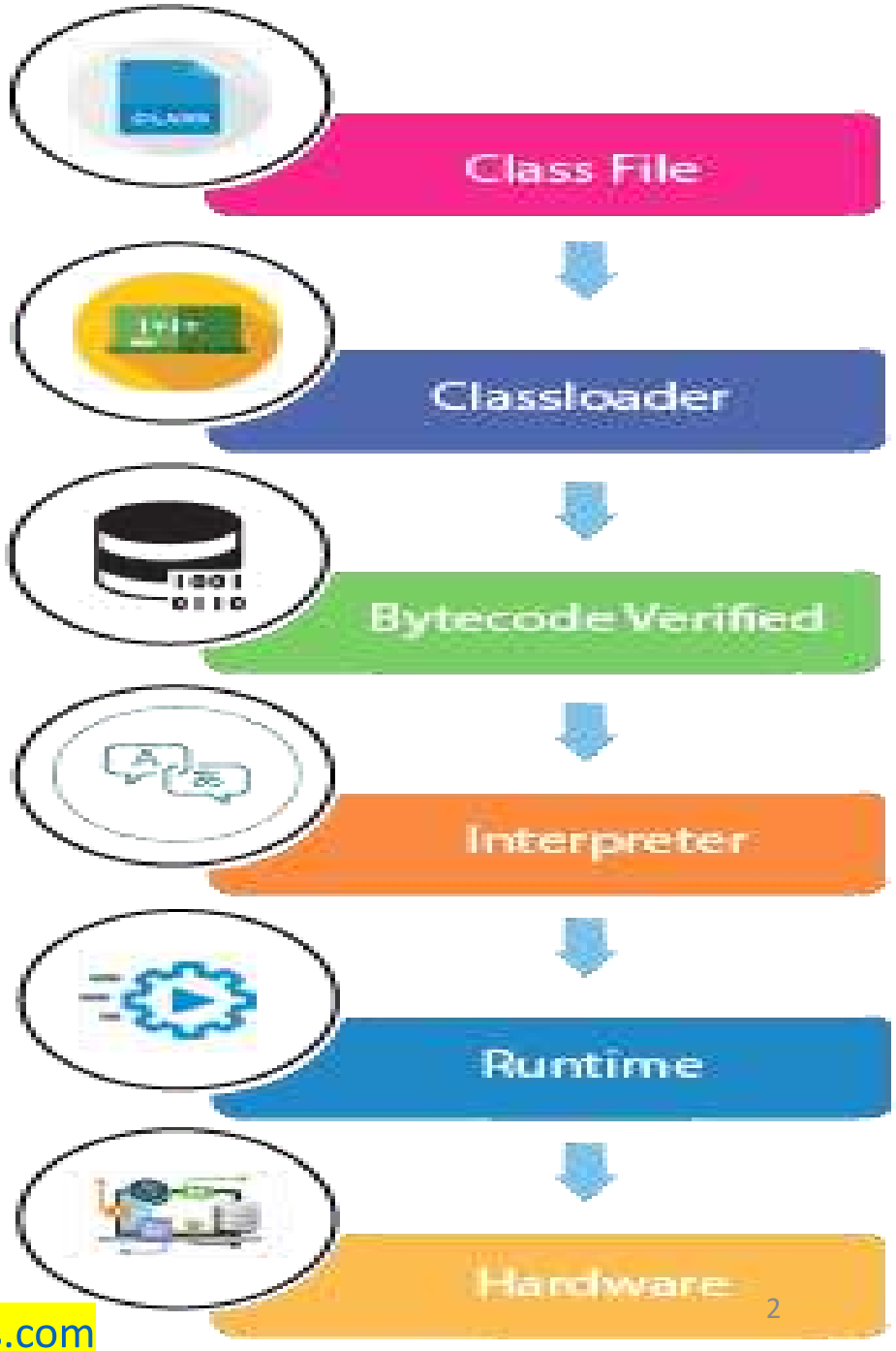
- Java Programs with executions
- Java Operators and Expressions
- Java Data Types
- Java Declaring (Creating) Variables, Display Values from Variables
- Java Comments, Variable and Naming Rules
- Java Compilation Flow, Parameters used in Java Program

Email id:
info@curiousinfotechsolutions.com

Compilation Flow







Java doesn't support default arguments like C++.

Java does not support header files like C++.

- Java uses the import keyword to include different classes and methods

C++ Program Example	Java Program Example

#include <iostream>
using namespace std;
int main() {
  cout << "Hello C++";
  return 0;
}
	class Simple{
	  public static void main(String args[]){
	  System.out.println("Hello Java");
	  }
	}

File: Simple.cpp	File: Simple.java

https://www.onlinegdb.com/online_java_compiler

Parameters used in Java Program

- **class** keyword is used to declare a class in Java.
- **public** keyword is an access modifier that represents visibility. It means it is visible to all.
- **static** is a keyword. If we declare any method as static, it is known as the static method. The core advantage of the static method is that there is no need to create an object to invoke the static method. The main() method is executed by the JVM, so it doesn't require creating an object to invoke the main() method. So, it saves memory.
- **void** is the return type of the method. It means it doesn't return any value.
- **main** represents the starting point of the program.
- **String[] args** or **String args[]** is used for command line argument.
- **System.out.println()** is used to print statement. Here, System is a class, out is an object of the PrintStream class, println() is a method of the PrintStream class.

Tips of Java Programming

- Every line of code that runs in Java must be inside a class.
- A class should always start with an uppercase first letter.
- Java is case-sensitive: "MyClass" and "myclass" has different meaning.
- The name of the java file must match the class name. When saving the file, save it using the class name and add ".java" to the end of the filename.
- The main() method is required Any code inside the main() method will be executed.
- The curly braces {} marks the beginning and the end of a block of code.
- Each code statement must end with a semicolon.

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
- Multi-line Comments

Java Single-line Comments

- Single-line comments start with two forward slashes (//).
- Any text between // and the end of the line is ignored by Java (will not be executed).

- This example uses a single-line comment before a line of code:

- Example-1

```
// This is input statement
System.out.println("Hello World");
```

- Example-2

```
// This is a comment System.out.println("Hello World");
```

Example-3

```
System.out.println("Hello World"); // This is a comment
```

Java Multi-line Comments

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

- Any text between /* and */ will be ignored by Java.

- Example

```
/* The code below will print the words Hello World
to the screen, and it is amazing */
System.out.println("Hello World");
```

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' or '@' . Char values are surrounded by single quotes.
  - boolean - stores values with two states: true or false

The general rules for naming variables

1. Names can contain letters, digits, underscores, and dollar signs
2. Names must begin with a letter
3. Names should start with a lowercase letter and it cannot contain whitespace
4. Names can also begin with $ and _
5. Names are case sensitive ("myVar" and "myvar" are different variables)
6. Reserved words (like Java **keywords**, such as int or boolean) cannot be used as names.

Declaring (Creating) Variables

- To create a variable, you must specify the type and assign it a value:

- Syntax-1
  - type variableName;
  - variableName=value;

Syntax-2

type variableName = value;

- Where type is one of Java's types (such as int or String), and variableName is the name of the variable (such as x or name). The equal sign is used to assign values to the variable.

Examples of Variables


```
String myName;
myName = "Vivek";
System.out.println(myName);
```

```
int myNum;
myNum = 15;
System.out.println(myNum);
```

```
float myFNum;
myFNum = 5.99f;
System.out.println(myFNum);
```

```
char myLetter;
myLetter = 'V';
System.out.println(myLetter);
```

Imp Notes

- During Declaring (Creating) Variables, there is also possible to initializing Variable with value.
  - int myNum = 5;
    - System.out.println(myNum);
  - float myFloatNum = 5.99f;
    - *System.out.println(myFloatNum);*
  - *char myLetter = 'D';*
    - System.out.println(myLetter);
  - boolean myBool = true;
    - System.out.println(myBool);
  - String myText = "Hello";
    - System.out.println(myText);

Output
5
5.99
D
true
Hello

Display Values of Variables

Display Welcome	System.out.print("Welcome");
Display Welcome Welcome	System.out.print("Welcome Welcome");
Display Welcome Welcome in New Line	System.out.print("Welcome \nWelcome");
Display Welcome Welcome with two print in New Line	System.out.print("Welcome\n"); System.out.print("Welcome");
Difference between print and println	System.out.println("Welcome"); System.out.print("Welcome");
Add two strings Wel and Come	Sytem.out.println("Wel"+"Come");
Let A=10 and display Value of A	System.out.println("A="+A);

Tip to Declare Many Variables

- int x = 5, y = 6, z = 50;
- System.out.println("x="+x + "y="+y + "z="+z);
- System.out.println("x="+x , "y="+y , "z="+z);

Java Identifiers

- All Java variables must be identified with PDF que 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
    - // Good
    - int minutesPerHour = 60;
  - // OK, but not so easy to understand what m actually is
    - int m = 60;

Java Data Types

a variable in Java must be a specified  a type

- 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**

Difference between **primitive** and **non-primitive** data types

- Primitive types are predefined (already defined) in Java. Non-primitive types are created by the programmer and is not defined by Java (except for String).
- Non-primitive types can be used to call methods to perform certain operations, while primitive types cannot.
- A primitive type has always a value, while non-primitive types can be null.
- A primitive type starts with a lowercase letter, while non-primitive types starts with an uppercase letter.
- The size of a primitive type depends on the data type, while non-primitive types have all the same size.

Primitive 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

Tips for Primitive Data Types

- byte myNum1 = 100;
  - System.out.println(myNum1);
- short myNum2 = 5000;
  - System.out.println(myNum2);
- int myNum3 = 100000;
  - System.out.println(myNum3);
- long myNum4 = 15000000000L;
  - System.out.println(myNum4);
- float myNum5 = 5.75f;
  - System.out.println(myNum5);
- double myNum6 = 19.99d;
  - System.out.println(myNum6);

Use float or double?

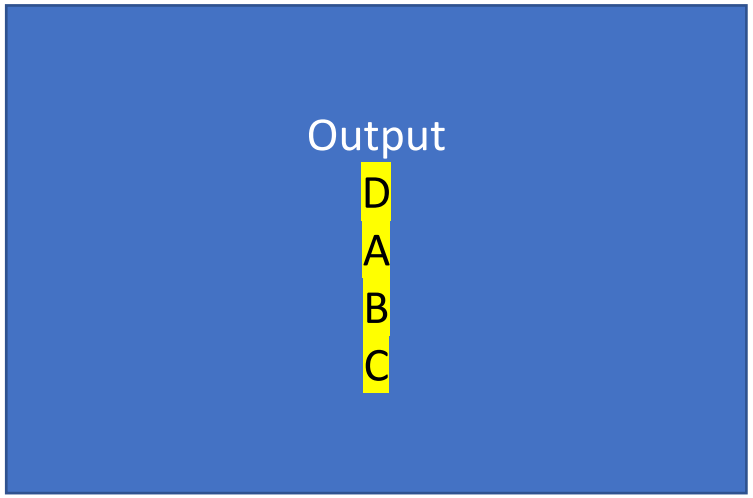
- The precision of a floating point value indicates how many digits the value can have after the decimal point.
- The precision of float is only six or seven decimal digits, while double variables have a precision of about 15 digits. Therefore, it is safer to use double for most calculations.
- Example:
  - float f1 = 22.0F/7.0F;
  - double d1 = 22.0D/7.0D;
  - System.out.println(f1);
  - System.out.println(d1);

Output

3.142857
3.142857142857143

Tips: Characters Data Types

- char myGrade = 'D';
- System.out.println(myGrade);
- char myVar1 = 65, myVar2 = 66, myVar3 = 67;
- System.out.println(myVar1);
- System.out.println(myVar2);
- System.out.println(myVar3);



Operators and Expressions in Java

- Unary Operator
- Arithmetic Operator
- Shift Operator
- Relational Operator
- Bitwise Operator
- Logical Operator
- Ternary Operator
- Assignment Operator

Operator Type	Category	Precedence
Unary	postfix	expr++ expr--
	prefix	++expr --expr +expr -expr ~ !
Arithmetic	multiplicative	* / %
	additive	+ -
Shift	shift	<< >> >>>
Relational	comparison	< > <= >= instanceof
	equality	== !=
Bitwise	bitwise AND	&
	bitwise exclusive OR	^
	bitwise inclusive OR	|
Logical	logical AND	&&
	logical OR	||
Ternary	ternary	? :
Assignment	assignment	= += -= *= /= %= &= ^= |= <<= >>= >>>=

Unary Operator Example

```
1.  public class Main
2.  {
    1.   public static void main(String args[])
    2.   {
         1.    int x=10;
         2.    System.out.println(x++);//10 (11)
         3.    System.out.println(++x);//12
         4.    System.out.println(x--);//12 (11)
         5.    System.out.println(--x);//10
    3.   }
3.  }
```

```
1.public class OperatorExample
2.{
    1.public static void main(String args[])
    2.{
        1.int a=10;
        2.int b=10;
        3.System.out.println(a++ + ++a);//10+12=22
        4.System.out.println(b++ + b++);//10+11=21
    3.  }
3.}
```

Java Arithmetic Operator Example

```
1.public class OperatorExample
2.{
    1.public static void main(String args[])
    2.{
        1.int a=10;
        2.int b=5;
        3.System.out.println(a+b);//15
        4.System.out.println(a-b);//5
        5.System.out.println(a*b);//50
        6.System.out.println(a/b);//2
        7.System.out.println(a%b);//0
    3.}
3.}
```

Java Shift Operator

```
1.public class OperatorExample
2.{
    1.public static void main(String args[])
    2.{
        1. System.out.println(10>>2);
        2. System.out.println(10<<2);
    3.}
3.}
```

Java Logical Operator

```
1. public class OperatorExample
2. {
    1. public static void main(String args[])
    2. {
        1. int a=10;
        2. int b=5;
        3. int c=20;
        4. System.out.println(a<b&&a<c);//false && true = false
        5. System.out.println(a>b||a>c);//true || false= true
    3. }
3. }
```

Java Ternary Operator

```
1.public class OperatorExample
2.{
    1.public static void main(String args[])
    2.{
        1.int a=2;
        2.int b=5;
        3.int min=(a<b)?a:b;
        4.System.out.println(min);
    3.}
3.}
```

Java Assignment Operator

```
1. public class OperatorExample
2. {
    1. public static void main(String args[])
    2. {
        1. int a=10;
        2. int b=20;
        3. a+=4;//a=a+4 (a=10+4)
        4. b-=4;//b=b-4 (b=20-4)
        5. System.out.println(a);
        6. System.out.println(b);
    3. }
```