Topic Covered:
Introduction to Object Oriented Programming, Introduction to Java, Installation of Java, Running your first program, General Structure of Program, Variables and Data Types.

OBJECT ORIENTED PROGRAMMING

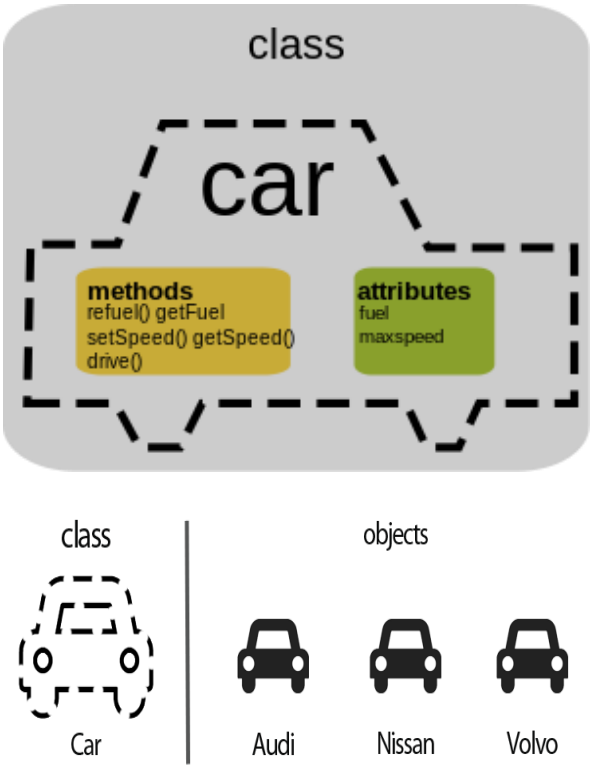
Programming paradigm based on the concept of objects, which may contain data, in the form of fields, often known as attributes; and code, in the form of procedures, often known as methods. Object-Oriented Programming (OOP) is a programming language model, which uses Objects and data as core components. The main idea behind OOP is to represent the data and logic with objects instead of actions or functions.

CLASS

The class defines the shape and nature of an object. A class is the template or blueprint from which objects are actually made. A class is also defined as a new data type, a user defined type which contain two things the Data Member and Methods. A class defines the properties and behaviors for objects. The description of a number of similar object is also called as a class. Classes is logical in nature.

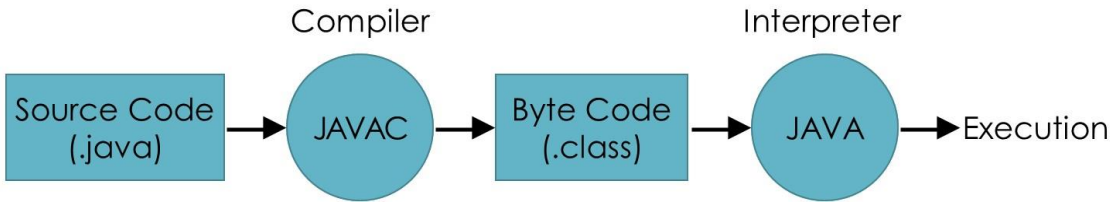
OBJECT

An object is an instance of a class. An object represents an entity in the real world that can be distinctly identified. For example, a student, a desk, a circle, a button, and even a loan can all be viewed as objects. Object to object communication is done via methods.
Software objects also have a state and a behavior:
The state of an object (also known as its properties or attributes) is represented by data fields with their current values. A circle object, for example, has a data field radius, which is the property that characterizes a circle. A dog has states -- color, name, breed.
The behavior of an object (also known as its actions) is defined by methods. To invoke a method on an object is to ask the object to perform an action. For example, you may define methods named getArea() and getPerimeter() for circle objects. A dog has behaviors -- wagging the tail, barking, eating.



INTRODUCTION TO JAVA

Java® is an object-oriented programming language that resembles C++ in many respects. One of the major differences is that Java® programs are intended to be architecture-neutral i.e. a Java® program should, in theory, be able to run on a Unix® workstation, a PC or a Macintosh® without recompilation. In C++, we compiled our programs into machine-dependent object code that was linked to produce an executable. By contrast, Java® programs are compiled into machine-independent byte code. The compiled program is then run within a Java® interpreter, which is responsible for executing the byte code instructions. The Java® interpreter is typically referred to as the Java® Virtual Machine, and it must be present on each computer that runs the program.



INSTALLING JAVA

The Java Development Kit (JDK), officially named "Java Platform, Standard Edition (Java SE)" is needed for writing Java programs. The JDK is freely available from Sun Microsystems (now part of Oracle). The mother site for JDK (Java SE) is http://www.oracle.com/technetwork/java/javase/overview/index.html.

JRE (Java Runtime) is needed for running Java programs. JDK (Java Development Kit), which includes JRE plus the development tools (such as compiler and debugger), is need for writing (developing) as well as running Java programs. In other words, JRE is a subset of JDK. Since you are supposed to write Java Programs, you should install JDK, which includes JRE.

Run the downloaded installer (e.g., "jdk-11.0.{x}_windows-x64_bin.exe"), which installs both the JDK.
By default, JDK is installed in directory "C:\Program Files\Java\jdk-11.0.{x}", where {x} denotes the upgrade number. Accept the defaults and follow the screen instructions to install JDK. Change this directory to “C:\jdk” for your ease so you can quickly access it.

WRITING YOUR FIRST PROGRAM:

In Java, every application begins with a class name, and that class must match the filename. Let's create our first Java file, called MyClass.java, which can be done in any text editor (like Notepad). The file should contain a "Hello World" message, which is written with the following code:

PROGRAM 1: Simple JAVA Program

```
//First Java Program
class MyClass
{
  public static void main(String[] args)
  {
    System.out.println("Hello World");
  }
}
```

COMPLIING AND RUNNING YOUR PROGRAM:

Save the code in Notepad in the directory c:\jdk\bin and name your file as "MyClass.java". Make sure to give your file name same as class name. 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. Open Command Prompt (cmd.exe), navigate to the directory where you saved your file by typing the following commands:

```
C:\Users\abc>  CD\JDK\BIN
```

To compile the program type "javac MyClass.java":

```
C:\JDK\BIN>  javac  MyClass.java
```

This will compile your code. If there are no errors in the code, the command prompt will take you to the next line. Now, type "java MyClass" to run the file:

```
C:\JDK\BIN>  java  MyClass
```

The output will be shown:

```
Hello World
```

EXAMPLE EXPLAINED

Every line of code that runs in Java must be inside a class. In our example, we named the class **MyClass**. A class should always start with an uppercase first letter. If several words are used to form a name of the class each inner words first letter should be in Upper Case. Example class **MyFirstJavaClass**
Note: Java is case-sensitive: "Hello" and "hello" has different meaning.

THE MAIN METHOD

The main() method is required and you will see it in every Java program:

```
public static void main(String[] args)
```

Any code inside the main() method will be executed every program must contain the main() method.
A block, which consists of 0, 1, or more statements, starts with a left curly brace ({) and ends with a right curly brace (}). Blocks are required for class and method definitions and can be used anywhere else in the program that a statement is legal. Example has two blocks: the class definition and the main method definition. As you can see, nesting blocks within blocks is perfectly legal. The main block is nested completely within the class definition block.

SYSTEM.OUT.PRINTLN()

Inside the main() method, we can use the println() method to print a line of text to the screen:

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

```
//First Java Program
/*First Java Program
Java is Fun */
```

JAVA IDENTIFIERS

All Java variables, method names, class names and other data members must be identified with unique names. These unique names are called identifiers.

EXERCISE 1-1:
Write a program which displays information about you like: Your Name, Fathers Name, Class Roll No., age, and Cell Phone number.

VARIABLES

Java allows you to refer to the data in a program by defining variables, which are named locations in memory where you can store values. A variable can store one data value at a time, but that value might change as the program executes.
Identifiers can be short names (like x and y) or more descriptive names (age, sum, totalVolume).
The general rules for constructing names for variables (unique identifiers) are:

- Names can contain letters, digits, underscores, and dollar signs
- Names should begin with a letter
- Names can also begin with $ and _
- Names are case sensitive ("myVar" and "myvar" are different variables)
- Names should start with a lowercase letter and it cannot contain whitespace. If the variable name consists of more than one word, then each word after the first should begin with a capital letter. For example, these identifiers are conventional Java variable names: number1, highScore, booksToRead, ageInYears, and xAxis.
- Reserved words (like Java keywords, such as int or String) cannot be used as names

DATA TYPES

Java supports eight primitive data types: byte, short, int, long, float, double, char, and boolean. They are called primitive data types because they are part of the core Java language. The data type you specify for a variable tells the compiler how much memory to allocate and the format in which to store the data.

DECLARING VARIABLES

Every variable must be given a name and a data type before it can be used. This is called declaring a variable. The syntax for declaring a variable is:

```
datatype identifier;
OR
datatype identifier1, identifier2, ...;
```

INTEGER DATA TYPES

This group includes **byte**, **short**, **int**, and **long**, which are for whole-valued signed numbers.

Integer Data Type	Size in Bytes	Min Value	Max Value
Byte	1	-128	127
Short	2	-32,768	32,767
Int	4	- 2,147,483,648	2,147,483,647
Long	8	-9,223,372,036,854,775,808	9,223,372,036,854,775,807

FLOATING-POINT DATA TYPES

This group includes **float** and **double**, which represent numbers with fractional precision.

Floating-point Data Type	Size in Bytes	Max Positive Non Zero Value	Max Value
Float	4	$1.40239846 \times 10^{-45}$	$3.40282347 \times 10^{38}$
Double	8	$4.9406564584124654 \times 10^{-324}$	$1.7976931348623157 \times 10^{308}$

CHARACTER DATA TYPE

Character Data Type	Size in Bytes	Min Value	Max Value
Char	2	The character encoded as 0000, the null character	The value FFFF which is a special code for "not a character"

BOOLEAN DATA TYPE

Boolean Data Type The boolean data type can store only two values, which are expressed using the Java reserved words true and false.

THE ASSIGNMENT OPERATOR, INITIAL VALUES, AND LITERALS

When you declare a variable, you can also assign an initial value to the data. To do that, use the assignment operator (=) with the following syntax:

```
datatype variableName = initialValue;
```

This statement is read as "variableName gets initialValue". Or

```
datatype variable1 = initialValue1, variable2 = initialValue2 ;
```

Notice that assignment is right to left. The initial value is assigned to the variable.

EXERCISE 1-2:

Write down a program which has 2 variable 'a' and 'b' assign them any value and evaluate the following equation:

$$x = a^2 + 2ab + b^2$$

Create variable x and Display the result of 'x'

PROGRAM 2: Calculates area of Circle

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

     //define the data we know (CONSTANT)
     final double PI = 3.14159;

     //define other data we use
     double radius;
     double area;

     //give radius a value
     radius = 3.5;

     //perform calculation
     area = PI * radius * radius;

     //output the result
     System.out.println("The area of Circle is " + area );
    }
  }
```

PROGRAM 3: Demonstrates long data type. Computes the number of miles that light will travel in a specified number of days.

```
class Light {
  public static void main(String args[]) {
     int lightspeed;
     long days;
     long seconds;
     long distance;
     // approximate speed of light in miles per second
     lightspeed = 186000;
     days = 1000; // specify number of days here
     seconds = days * 24 * 60 * 60; // convert to seconds
     distance = lightspeed * seconds; // compute distance
     System.out.print("In " + days);
     System.out.print(" days light will travel about ");
     System.out.println(distance + " miles.");
  }
}
```

PROGRAM 4: Demonstrates char variables behave like integers.

```
class CharDemo2 {
  public static void main(String args[]) {
     char ch1;
     ch1 = 'X';
     System.out.println("ch1 contains " + ch1);
     ch1++; // increment ch1
     System.out.println("ch1 is now " + ch1);
  }
}
```

PROGRAM 5: Demonstrate boolean values.

```
class BoolTest {
  public static void main(String args[]) {
     boolean b;
     b = false;
     System.out.println("b is " + b);
     b = true;
     System.out.println("b is " + b);
     // a boolean value can control the if statement
     if(b) System.out.println("This is executed.");
     b = false;
     if(b) System.out.println("This is not executed.");
     // outcome of a relational operator is a boolean value
     System.out.println("10 > 9 is " + (10 > 9));
  }
}
```

PROGRAM 6: Demonstrate dynamic initialization.

```
class DynInit {
  public static void main(String args[]) {
     double a = 3.0, b = 4.0;
     // c is dynamically initialized
     double c = Math.sqrt(a * a + b * b);
     System.out.println("Hypotenuse is " + c);
  }
}
```