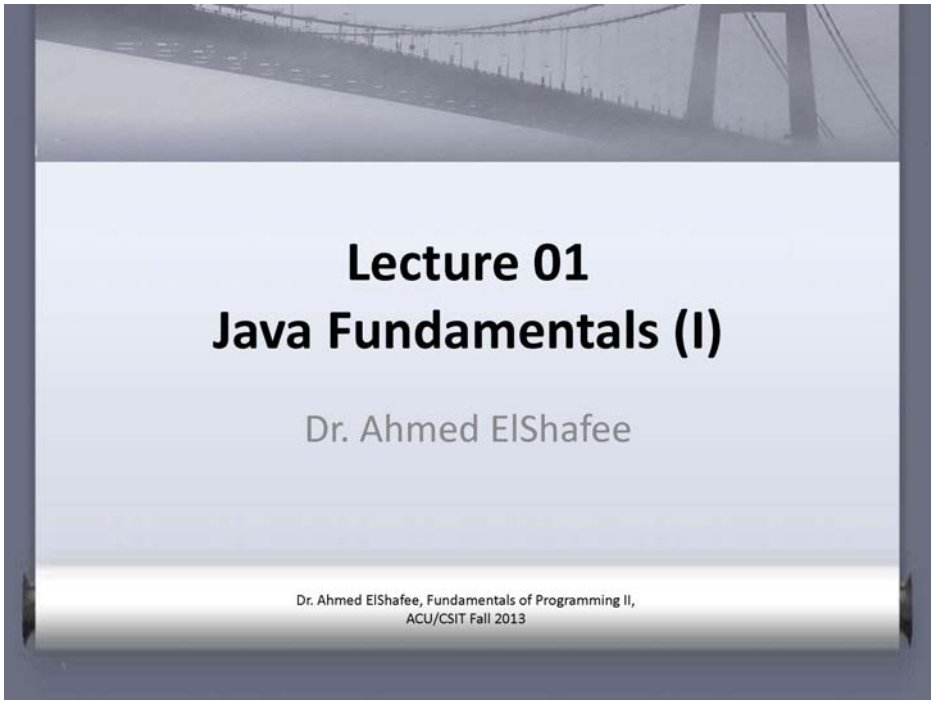

Agenda

1. Java program structure
2. Basic Syntax
3. Java Identifiers
4. Java Keywords
5. Comments in Java
6. Java Basic Data Types
7. Java Basic Operators
8. Java IO
9. Casting
10. **Java - Number Class**

Java program structure

- a simple code that would print the words *Hello World*.

```
public class MyFirstJavaProgram{

   /* This is my first java program.
    * This will print 'Hello World' as the output
    */

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


Basic Syntax

- Case Sensitivity –
  - Java is case sensitive which means identifier Hello and hello would have different meaning in Java.
- Class Names –
  - For all class names the first letter should be in Upper Case.
  - 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

- Method Names –
  - All method names should start with a Lower Case letter.
  - If several words are used to form the name of the method, then each inner word's first letter should be in Upper Case.
  - Example public void myMethodName()

- Program File Name –
  - Name of the program file should exactly match the class name.
  - When saving the file you should save it using the class name (Remember java is case sensitive) and append '.java' to the end of the name. (if the file name and the class name do not match your program will not compile).
  - Example : Assume 'MyFirstJavaProgram' is the class name. Then the file should be saved as 'MyFirstJavaProgram.java'
- **public static void main(String args[])** –
  - java program processing starts from the main() method which is a mandatory part of every java program..

Java Identifiers

- All java components require names.
- Names used for classes, variables and methods are called identifiers.
- In java there are several points to remember about identifiers. They are as follows:
  - All identifiers should begin with a letter (A to Z or a to z), currency character ($) or an underscore (-).
  - After the first character identifiers can have any combination of characters.
  - A key word cannot be used as an identifier.
  - Most importantly identifiers are case sensitive.
  - Examples of legal identifiers:age, $salary, _value, __1_value
  - Examples of illegal identifiers : 123abc, -salary

Java Keywords:

abstract	assert	boolean	break
byte	case	catch	char
class	const	continue	default
do	double	else	enum
extends	final	finally	float
for	goto	if	implements
import	instanceof	int	interface
long	native	new	package
private	protected	public	return
short	static	strictfp	super
switch	synchronized	this	throw
throws	transient	try	void
volatile	while		

Comments in Java

```
public class MyFirstJavaProgram{

  /* This is my first java program.
   * This will print 'Hello World' as the output
   * This is an example of multi-line comments.
   */

   public static void main(String []args){
      // This is an example of single line comment
      /* This is also an example of single line comment. */
      System.out.println("Hello World");
   }
}
```


Java Basic Data Types

- Variables are reserved memory locations to store values. This means that when you create a variable you reserve some space in memory.
- Based on the data type of a variable, the operating system allocates memory and decides what can be stored in the reserved memory.
- Therefore, by assigning different data types to variables, you can store integers, decimals, or characters in these variables.
- There are two data types available in Java:
  - Primitive Data Types
  - Reference/Object Data Types

1. Primitive Data Types:

- There are eight primitive data types supported by Java. Primitive data types are predefined by the language and named by a key word, default value is zero.
- **byte:** Byte data type is a 8-bit signed two's complement integer.
- **short:** Short data type is a 16-bit signed two's complement integer.
- **int:** Int data type is a 32-bit signed two's complement integer.
- **long:** Long data type is a 64-bit signed two's complement integer.
- **float:** Float data type is a single-precision 32-bit IEEE 754 floating point.

- **double:** double data type is a double-precision 64-bit IEEE 754 floating point.
- **boolean:** boolean data type represents one bit of information.
- **char:** char data type is a single 16-bit Unicode character.

2. Reference Data Types:

- Class objects, and various type of array variables come under reference data type.
- Default value of any reference variable is null.
- A reference variable can be used to refer to any object of the declared type or any compatible type.
- Example : Animal animal = new Animal("giraffe");

Java Literals:

- A literal is a source code representation of a fixed value. They are represented directly in the code without any computation.
- Literals can be assigned to any primitive type variable. For example:

```
byte a = 68;
char a = 'A'
```

- byte, int, long, and short can be expressed in decimal(base 10),hexadecimal(base 16) or octal(base 8) number systems as well.
- Prefix 0 is used to indicates octal and prefix 0x indicates hexadecimal when using these number systems for literals. For example:


```
int decimal = 100;
int octal = 0144;
int hexa = 0x64;
```

- string literals in Java are a sequence of characters between a pair of double quotes.

```
"Hello World"
"two\nlines"
"\"This is in quotes\""
```

- String and char types of literals can contain any Unicode characters. For example:

```
char a = '\u0001';
String a = "\u0001";
```


- special escape sequences for String and char literals as well.

Notation	Character represented
\n	Newline (0x0a)
\r	Carriage return (0x0d)
\f	Formfeed (0x0c)
\b	Backspace (0x08)
\s	Space (0x20)
\t	tab
\"	Double quote
\'	Single quote
\\	backslash
\ddd	Octal character (ddd)
\uxxxx	Hexadecimal UNICODE character (xxxx)

Java Basic Operators

- Java provides a rich set of operators to manipulate variables. We can divide all the Java operators into the following groups:
  - Arithmetic Operators
  - Relational Operators
  - Bitwise Operators
  - Logical Operators
  - Assignment Operators
  - Misc Operators

The Arithmetic Operators:

- Arithmetic operators are used in mathematical expressions in the same way that they are used in algebra. The following table lists the arithmetic operators:
- Assume integer variable A holds 10 and variable B holds 20 then:

Operator	Description	Example
+	Addition - Adds values on either side of the operator	A + B will give 30
-	Subtraction - Subtracts right hand operand from left hand operand	A - B will give -10
*	Multiplication - Multiplies values on either side of the operator	A * B will give 200
/	Division - Divides left hand operand by right hand operand	B / A will give 2
%	Modulus - Divides left hand operand by right hand operand and returns remainder	B % A will give 0
++	Increment - Increase the value of operand by 1	B++ gives 21
--	Decrement - Decrease the value of operand by 1	B-- gives 19


```
class MySecondJavaProgram{
public static void main(String args[]) {
  int a = 10;
  int b = 20;
  int c = 25;
  int d = 25;
  System.out.println("a + b = " + (a + b) );
  System.out.println("a - b = " + (a - b) );
  System.out.println("a * b = " + (a * b) );
  System.out.println("b / a = " + (b / a) );
  System.out.println("b % a = " + (b % a) );
  System.out.println("c % a = " + (c % a) );
  System.out.println("a++   = " +  (a++) );
  System.out.println("b--   = " +  (a--) );
```

```
 // Check the difference in d++ and ++d
   System.out.println("d++   = " +  (d++) );
   System.out.println("++d   = " +  (++d) );
}
}
```

```
a + b = 30
a - b = -10
a * b = 200
b / a = 2
b % a = 0
c % a = 5
a++   = 10
b--   = 11
d++   = 25
++d   = 27
```


The Relational Operators:

Operator	Description	Example
==	Checks if the value of two operands are equal or not, if yes then condition becomes true.	(A == B) is not true.
!=	Checks if the value of two operands are equal or not, if values are not equal then condition becomes true.	(A != B) is true.
>	Checks if the value of left operand is greater than the value of right operand, if yes then condition becomes true.	(A > B) is not true.
<	Checks if the value of left operand is less than the value of right operand, if yes then condition becomes true.	(A < B) is true.
>=	Checks if the value of left operand is greater than or equal to the value of right operand, if yes then condition becomes true.	(A >= B) is not true.
<=	Checks if the value of left operand is less than or equal to the value of right operand, if yes then condition becomes true.	(A <= B) is true.

```
class MyThirdJavaProgram{
public static void main(String args[]) {
  int a = 10;
  int b = 20;
  System.out.println("a == b = " + (a == b) );
  System.out.println("a != b = " + (a != b) );
  System.out.println("a > b = " + (a > b) );
  System.out.println("a < b = " + (a < b) );
  System.out.println("b >= a = " + (b >= a) );
  System.out.println("b <= a = " + (b <= a) );
}
}
```



```
a == b = false
a != b = true
a > b = false
a < b = true
b >= a = true
b <= a = false
```

- Bitwise operators
  - And &
  - Or |
  - Xor ^
  - not ~
  - Left shift <<
  - Right shift >>


```
public class mythirdjavaprogram4{
  public static void main(String args[]) {
    int a = 60;            /* 60 = 0011 1100 */
    int b = 13;            /* 13 = 0000 1101 */
    int c = 0;
    c = a & b;     /* 12 = 0000 1100 */
    System.out.println("a & b = " + c );
    c = a | b;     /* 61 = 0011 1101 */
    System.out.println("a | b = " + c );
    c = a ^ b;     /* 49 = 0011 0001 */
    System.out.println("a ^ b = " + c );
    c = ~a;        /*-61 = 1100 0011 */
    System.out.println("~a = " + c );
    c = a << 2;    /* 240 = 1111 0000 */
    System.out.println("a << 2 = " + c );
    c = a >> 2;    /* 215 = 1111 */
    System.out.println("a >> 2  = " + c );
    c = a >>> 2;   /* 215 = 0000 1111 */
    System.out.println("a >>> 2 = " + c );
  }
}
```


- Logical operator
  - && Called Logical AND operator.
  - || Called Logical OR Operator.
  - ! Called Logical NOT Operator.

- **The Assignment Operators:**

operator	description
=	Simple assignment operator, Assigns values from right side operands to left side operand
+=	Add AND assignment operator,
-=	Subtract AND assignment operator,
*=	Multiply AND assignment operator
/=	Divide AND assignment operator, It
%=	Modulus AND assignment operator,
<<=	Left shift AND assignment operator
>>=	Right shift AND assignment operator
&=	Bitwise AND assignment operator
^=	bitwise exclusive OR and assignment operator
|=	bitwise inclusive OR and assignment operator

- **Misc Operators**
  - **Conditional Operator (? :):**
  - **instanceOf Operator:**

Java IO

print & println

```
String first_name="zaki";
String family_name="gedan";
System.out.println(first_name + " " + family_name);
System.out.print(first_name + " " + family_name);
```

```
String first_name="zaki";
String family_name="gedan";
Char space=" ";
System.out.println(first_name + space + family_name);
System.out.print(first_name + space + family_name);
```


printf (formatted string)

Data type	symbol	function
char	%c	Sysetm.out.printf("this is char: %c", ch);
String	%s	Sysetm.out.printf("my name is %s %s", first_name, last_name);
integers	%d	Sysetm.out.printf("%d + %d = %d", num1, num2, answer);
double	%f	Sysetm.out.printf("%f + %f = %f", num1, num2, answer);

Data type	symbol	function
short	%d	Sysetm.out.printf("%d + %d = %d", num1, num2, answer);
float	%f	Sysetm.out.printf("%f + %f = %f", num1, num2, answer);
byte	%d	Sysetm.out.printf("%d + %d = %d", num1, num2, answer);
Boolean	%b	Sysetm.out.printf("yes this is %b", 1);
Hexadecimal	%x	Sysetm.out.printf("0x%x + 0x%x = 0x%x", num1, num2, answer);
Octal	%o	Sysetm.out.printf("%o + %o = %o", num1, num2, answer);

Scanners

1. import library

```
import java.util.Scanner;
```

2. Create object from scanner

```
Scanner user_input = new Scanner(System.in);
```

3. Accepting string input from user

```
String family_name = user_input.next( );
```

* You can accept any type of inputs from the user as follows

```
x = user_input.nextXYZ()
```

where XYZ is one of BigDecimal, BigInteger, Boolean, Byte, Float, or Short


```
class MyThirdJavaProgram2{
public static void main(String[] args) {
      String first_name, last_name, comment;
      int id,level, gained_credit_hours;
      float GPA;
      Scanner s=new Scanner(System.in);
      System.out.printf("\nEnter Your ID (ineteger):");
      id=s.nextInt();
      System.out.printf("\nEnter Your first name:");
      first_name=s.next();
      System.out.printf("\nEnter Your family name:");
      last_name=s.next();
      System.out.printf("\nEnter Your Level(1,2,3,4):");
      level=s.nextInt();
      System.out.printf("\nEnter Your Gained Credit hours till now (integer):");
```



```
 gained_credit_hours=s.nextInt();
System.out.printf("\nEnter Your your GPA (float):");
      GPA=s.nextFloat();
      System.out.printf("\ntype your comments (ended by Enter):");
      comment=s.next();

System.out.printf("\n**************************************************"
);
      System.out.printf("\nID\t%d\nname\t%s %s\nLevel\t%d\nCH\t%d\nGPA\t%f",
            id, first_name, last_name, level, gained_credit_hours, GPA);
      System.out.println("\nCommnet:\n"+comment);
   }
}
```



```
Enter Your ID (ineteger):4091001
Enter Your first name:mosfata
Enter Your family name:diaa
Enter Your Level(1,2,3,4):4
Enter Your Gained Credit hours till now (integer):3
Enter Your your GPA (float):3.5
type your comments (ended by Enter):no comment
**************************************************
ID      4091001
name    mosfata diaa
Level   4
CH      123
GPA     3.500000
Commnet:
no
```

JOptionPane

1. Import library

```
import javax.swing.JOptionPane;
```

2. Show message

```
JOptionPane.showMessageDialog( null, full_name );
```

3. Accept input string

```
first_name = JOptionPane.showInputDialog("First Name","Enter Your Family Name");
```


Casting

Converting string to integer

```
String string_num="10";
int num= Integer.parseInt( string_num );
```

Converting int to string

```
int n=10;
String str="this number equals to: "+n;
```

```
int n=10;
String strn=Integer.toString(n);
String str="this number equals to: "+strn;
```

That applies to all types of numerical variables


```
class MyThirdJavaProgram3{
public static void main(String[] args) {
    String user_input;
    float Width, hight, area;
    user_input=JOptionPane.showInputDialog("Width","0");
    Width=Float.parseFloat(user_input);
    user_input=JOptionPane.showInputDialog("Hight","0");
    hight=Float.parseFloat(user_input);
    area=Width*hight;
    JOptionPane.showMessageDialog(null, "Rectangle Area = "+area,"Rectangle",
JOptionPane.INFORMATION_MESSAGE);
        }
}
```