Lecture 2

Object Oriented Concepts and Programming (CSC244)

By

Dr. Tabbasum Naz

tabbasum.naz@ciitlahore.edu.pk

A Simple Java Program

```
1  // Fig. 2.1: Welcome1.java
2  // A first program in Java.
3
4  public class Welcome1 {
5
6     // main method begins execution of Java application
7     public static void main( String args[] )
8     {
9        System.out.println( "Welcome to Java Programming!" );
10
11    } // end method main
12
13 } // end class Welcome1
```

```
Welcome to Java Programming!
```

Java Basic Types

Type	Contains	Default	Size	Range
boolean	true or false	false	1 bit	NA
char	Unicode character unsigned	\u0000	16 bits or 2 bytes	0 to 2^{16}-1 or \u0000 to \uFFFF
byte	Signed integer	0	8 bit or 1 byte	-2^{7} to 2^{7}-1 or -128 to 127
short	Signed integer	0	16 bit or 2 bytes	-2^{15} to 2^{15}-1 or -32768 to 32767
int	Signed integer	0	32 bit or 4 bytes	-2^{31} to 2^{31}-1 or -2147483648 to 2147483647
long	Signed integer	0	64 bit or 8 bytes	-2^{63} to 2^{63}-1 or -9223372036854775808 to 9223372036854775807
float	IEEE 754 floating point single-precision	0.0f	32 bit or 4 bytes	±1.4E-45 to ±3.4028235E+38
double	IEEE 754 floating point double-precision	0.0	64 bit or 8 bytes	±439E-324 to ±1.7976931348623157E+308

Naming Guidelines

- The rules for variable names and identifiers follow the same conventions as C:
  - Variables and methods must start with lowercase

```
double x;
int sessionalMarks=0;
public  void accountDetails();
```

  - `studentMarks` is preferred to student-marks
  - Class names must start with uppercase

    public class `HelloWorld{}`

Naming Guidelines (Contd)

- Lower Camel Case: Each new word begins with a capital letter except the first letter of the name is in lowercase (e.g., `hasChildren`, `customerFirstName, customerLastName`).
- Using proper naming convention increases the readability of your program.
- Use pronounceable names
- Be brief
- `calculateArea` instead of `calculateTheAreaOfRectangle`

Java for Beginners

- Java class name is also called identifier
- Reserved words or keywords are reserved by java and cannot be used as identifiers
- Identifier may contain letters, dollar sign, underscores, digits and letters
- Identifier does not begin with a digit
- Identifier does not contain spaces

Valid Identifiers	Invalid Identifiers
_Hello	1Name
Hello_World	Hello World
Name1	

Java Reserved Words

- Reserved words cannot be used as identifiers.

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

Escape Sequences in Java

- Backslash (\) is called an escape character.
- When a backslash apperas in string of characters. Java combines the next character with the backslash to form an escape sequence.

Escape sequence	Description
`\n`	Newline. Position the screen cursor to the beginning of the next line.
`\t`	Horizontal tab. Move the screen cursor to the next tab stop.
`\r`	Carriage return. Position the screen cursor to the beginning of the current line; do not advance to the next line. Any characters output after the carriage return overwrite the characters previously output on that line.
`\\`	Backslash. Used to print a backslash character.

`\"`	Double quote. Used to print a double-quote character. For example,
	`System.out.println( "\"in quotes\"" );`
	displays
	`"in quotes"`

Arithmetic Operators

Java operation	Arithmetic operator	Algebraic expression	Java expression
Addition	+	$f + 7$	`f + 7`
Subtraction	-	$p - c$	`p - c`
Multiplication	*	bm	`b * m`
Division	/	x / y or $\frac{x}{y}$ or $x \div y$	`x / y`
Remainder	%	$r \bmod s$	`r % s`

Fig. 2.11 | Arithmetic operators.

Precedence of Arithmetic Operators

Operator(s)	Operation(s)	Order of evaluation (precedence)
*	Multiplication	Evaluated first. If there are several operators of this type, they are evaluated from left to right.
/	Division	
%	Remainder	
+	Addition	Evaluated next. If there are several operators of this type, they are evaluated from left to right.
-	Subtraction	

Fig. 2.12 | Precedence of arithmetic operators.

Precedence of Arithmetic Operators (Contd)

Algebra: $z = pr\%q + w/x - y$

Java:

```
z = p * r % q + w / x - y;
```

6 1 2 4 3 5

$(y = ax^2 + bx + c)$

```
y = a * x * x + b * x + c;
```

6 1 2 4 3 5

Equality and Relational Operators

Standard algebraic equality or relational operator	Java equality or relational operator	Sample Java condition	Meaning of Java condition
Equality operators			
=	==	x == y	x is equal to y
≠	!=	x != y	x is not equal to y
Relational operators			
>	>	x > y	x is greater than y
<	<	x < y	x is less than y
≥	>=	x >= y	x is greater than or equal to y
≤	<=	x <= y	x is less than or equal to y

Fig. 2.14 | Equality and relational operators. (Part 2 of 2.)

Basic Addition Program

```
import java.util.Scanner ;
public class Addition
{
    public static void main (String[] args)
    {
            Scanner input = new Scanner (System.in);
            int num1, num2, sum;
            System.out.println (“Enter first number:");
            num1 = input.nextInt();
            System.out.println (“Enter second number:");
            num2 = input.nextInt();
            sum = num1 + num2 ;
            System.out.printf ("sum = %d\n", sum);
    }
}
```

Simple Program with Equality and Relational Operators

```
1  // Fig. 2.15: Comparison.java
2  // Compare integers using if statements, relational operators
3  // and equality operators.
4  import java.util.Scanner; // program uses class Scanner
5
6  public class Comparison
7  {
8     // main method begins execution of Java application
9     public static void main( String args[] )
10    {
11       // create Scanner to obtain input from command window
12       Scanner input = new Scanner( System.in );
13
14       int number1; // first number to compare
15       int number2; // second number to compare
16
17       System.out.print( "Enter first integer: " ); // prompt
18       number1 = input.nextInt(); // read first number from user
```

Fig. 2.15 | Equality and relational operators. (Part 1 of 2.)

Simple Program with Equality and Relational Operators (Contd)

```
19
20          System.out.print( "Enter second integer: " ); // prompt
21          number2 = input.nextInt(); // read second number from user
22
23          if ( number1 == number2 )
24             System.out.printf( "%d == %d\n", number1, number2 );
25
26          if ( number1 != number2 )
27             System.out.printf( "%d != %d\n", number1, number2 );
28
29          if ( number1 < number2 )
30             System.out.printf( "%d < %d\n", number1, number2 );
31
32          if ( number1 > number2 )
33             System.out.printf( "%d > %d\n", number1, number2 );
34
35          if ( number1 <= number2 )
36             System.out.printf( "%d <= %d\n", number1, number2 );
37
38          if ( number1 >= number2 )
39             System.out.printf( "%d >= %d\n", number1, number2 );
40
41       } // end method main
42
43    } // end class Comparison
```

Output

```
Enter first integer: 777
Enter second integer: 777
777 == 777
777 <= 777
777 >= 777
```

```
Enter first integer: 1000
Enter second integer: 2000
1000 != 2000
1000 < 2000
1000 <= 2000
```

```
Enter first integer: 2000
Enter second integer: 1000
2000 != 1000
2000 > 1000
2000 >= 1000
```

Fig. 2.15 | Equality and relational operators. (Part 2 of 2.)

Precedence and Associatively of Operations

Operators	Associativity	Type
* / %	left to right	multiplicative
+ -	left to right	additive
< <= > >=	left to right	relational
== !=	left to right	equality
=	right to left	assignment

Fig. 2.16 | Precedence and associativity of operations discussed.

Thank You