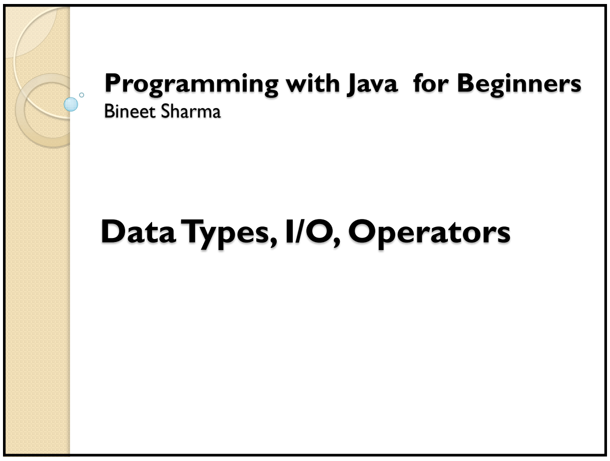
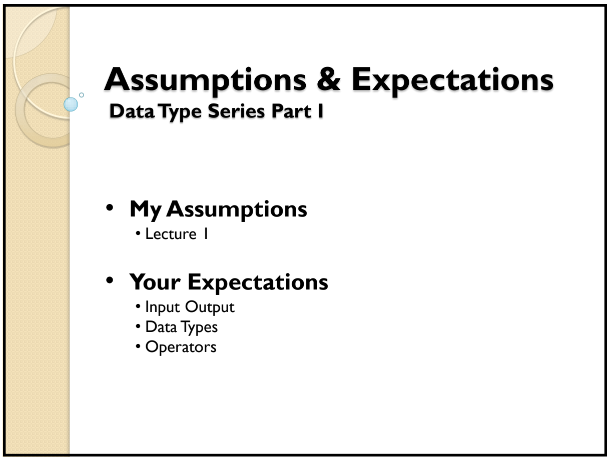


Programming with Java for Beginners

Bineet Sharma

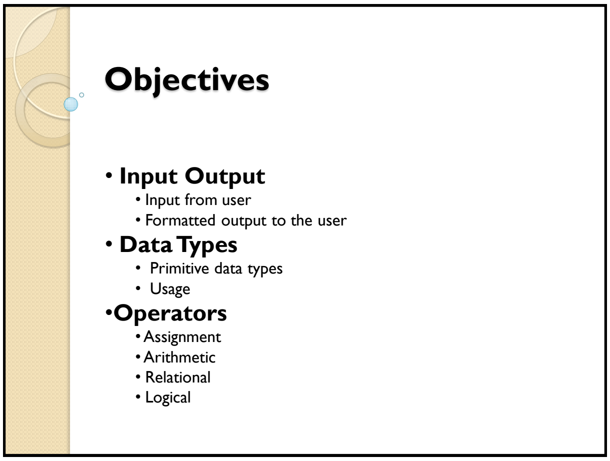
Data Types, I/O, Operators



Assumptions & Expectations

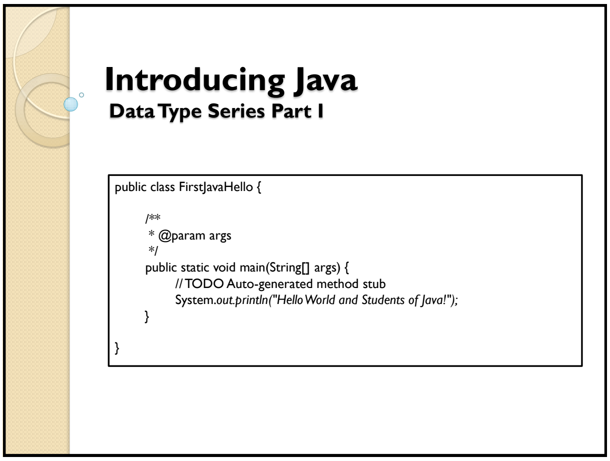
Data Type Series Part I

- **My Assumptions**
  - Lecture 1
- **Your Expectations**
  - Input Output
  - Data Types
  - Operators



Objectives

- **Input Output**
  - Input from user
  - Formatted output to the user
- **Data Types**
  - Primitive data types
  - Usage
- **Operators**
  - Assignment
  - Arithmetic
  - Relational
  - Logical



Introducing Java

Data Type Series Part I

```
public class FirstJavaHello {

        /**
         * @param args
         */
        public static void main(String[] args) {
                // TODO Auto-generated method stub
                System.out.println("Hello World and Students of Java!");
        }

}
```

Introducing Java

Data Type Series Part I

Dissecting the program (Comments in Java):

a) First lines are comments, which are ignored by compiler.
b) Needed for maintenance purposes.
c) Use // for single line, /* */ for multiline.
d) Over commenting is not good.
e) Must maintain comments as well.

```
//PROGRAM:                  FirstJavaHello.java
//Developer:                Bineet Sharma
//Date:                     11/22/2012
//Description:              Write a program to illustrate Java capability
//Command line arguments:
/*
   Known Issues:
   Revision:
*/
public class FirstJavaHello {
       …
}
```

Introducing Java

Data Type Series Part I

Dissecting the program (Java Class):

a) Every code in Java is inside a class. It has a body within {}.
b) In this case, the class name is name of the application as well.
c) This is a public class that is why the file name must be the name of class as well FirstJavaHello.java (one public class per file).
d) *javac* compiles the .java file and produces .class byte code.

```
// ….
//Revision:

public class FirstJavaHello {
      /**
       * @param args
       */
      public static void main(String[] args) {
            // TODO Auto-generated method stub
            System.out.println("Hello World and Students of Java!");
      }
}
```

Introducing Java

Data Type Series Part I

Dissecting the program (method *main*):

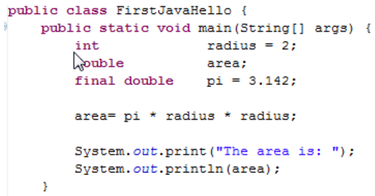
a) Java program starts executing with a method called **main**(). It has a body within {}. Is unique, and only one per application.
b) **System.out** is an object which knows how to display a character in a terminal.
c) **println**: is the message sent to the System.out object. The Strings in quotation marks is a **parameter** to println **method** and contains characters to be printed.

```
public class FirstJavaHello {
      /**
       * @param args
       */
      public static void main(String[] args) {
            // TODO Auto-generated method stub
            System.out.println("Hello World and Students of Java!");
      }
}
```

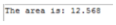
Introducing Java

Data Type Series Part I

How do you determine the area of a circle?



```
public class FirstJavaHello {
    public static void main(String[] args) {
        int             radius = 2;
        double          area;
        final double    pi = 3.142;

        area= pi * radius * radius;

        System.out.print("The area is: ");
        System.out.println(area);
    }
```



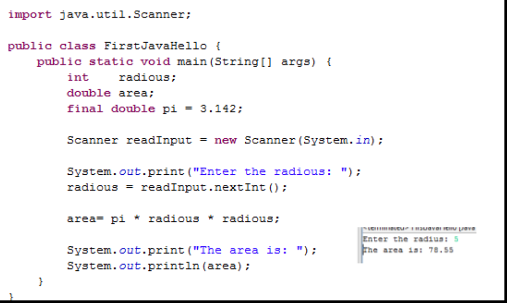
```
The area is: 12.568
```

Can we improve on this? What is missing?

Introducing Java

Data Type Series Part I

What is the improvement? How does this program work?



```
import java.util.Scanner;

public class FirstJavaHello {
    public static void main(String[] args) {
        int     radious;
        double area;
        final double pi = 3.142;

        Scanner readInput = new Scanner(System.in);

        System.out.print("Enter the radious: ");
        radious = readInput.nextInt();

        area= pi * radious * radious;

        System.out.print("The area is: ");
        System.out.println(area);
    }
}
```

```
Enter the radius: 5
The area is: 78.55
```

Input/Output in Java

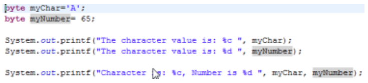
Data Type Series Part I

- **Input Operation**
  - Copy data from input device usually a keyboard.
  - Performed by using methods in library like Scanner package.
  - Format specifier specifics to the data type is used.
  - For example: %d for integer, %f for float.
- **Output Operation**
  - Display information stored in memory to output device usually a screen.
  - Performed by output methods in System.out package.
  - Format specifier specifics to the data type is used.
  - For example: %d for integer, %f for float.
- **Use Scanner, System.out packages**

print, printf, and println methods

Data Type Series Part I

- **Display information in std output device**
- **Syntax:**
  - printf(FormatControlString);
  - printf(FormatControlString, datalist);
- **FormatControlString:**
  - contains characters, format specifier, and escape sequences
- **Datalist:**
  - contains any constants, variables, expressions, and function calls separated by commas



```
byte myChar='A';
byte myNumber= 65;

System.out.printf("The character value is: %c ", myChar);
System.out.printf("The character value is: %d ", myNumber);

System.out.printf("Character is: %c, Number is %d ", myChar, myNumber);
```

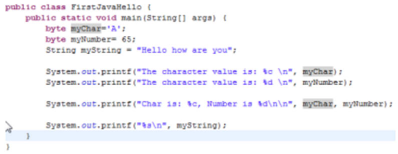
printf() Format Specifier

Data Type Series Part I

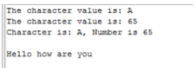
Format Specifier	Data Type
%d	signed decimal integer
%c	single character
%s	string or character array
%f	floating point (decimal notation)
%e	double (floatng point - exponential notation)
%g	floating point (%f or %e whichever is shorter)
%u	unsigned decimal integer
%x	unsigned hexadecimal integer (uses “abcdef”)
%o	unsigned octal integer
l	prefix used with %d, %u, %x, %o to specify long integer for example %ld

printf() Example

Data Type Series Part I



```
public class FirstJavaHello {
    public static void main(String[] args) {
        byte myChar='A';
        byte myNumber= 65;
        String myString = "Hello how are you";

        System.out.printf("The character value is: %c \n", myChar);
        System.out.printf("The character value is: %d \n", myNumber);

        System.out.printf("Char is: %c, Number is %d\n\n", myChar, myNumber);

        System.out.printf("%s\n", myString);
    }
}
```



```
The character value is: A
The character value is: 65
Character is: A, Number is 65

Hello how are you
```

Escape sequence

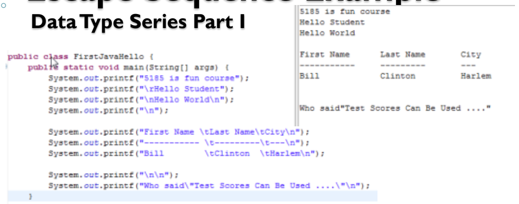
Data Type Series Part I

- Special meaning if used in printf() method
- Causes escape from normal interpretation of format specifier string

Cursor Control	ASCII value	Formatters
\n	010	newline (starts a new line)
\t	009	tab (moves cursor 8 spaces to right)
\r	013	carriage return (moves cursor to the beginning of the next line)
\b	008	backspace (moves cursor one space to left)
\f	012	form feed (advances printer paper one page)
\v	011	vertical tab
\’	039	single quote
\0	000	null character
\\	092	backslash
\”	034	double quote
\?	063	question

Escape Sequence Example

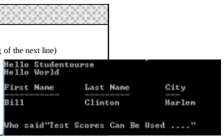
Data Type Series Part I



```
public class FirstJavaHello {
    public static void main(String[] args) {
        System.out.printf("5185 is fun course");
        System.out.printf("\rHello Student");
        System.out.printf("\nHello World\n");
        System.out.printf("\n");

        System.out.printf("First Name \tLast Name\tCity\n");
        System.out.printf("----------- \t---------\t---\n");
        System.out.printf("Bill        \tClinton  \tHarlem\n");

        System.out.printf("\n\n");
        System.out.printf("Who said\"Test Scores Can Be Used ....\"\n");
    }
```

```
5185 is fun course
Hello Student
Hello World

First Name      Last Name       City
-----------     ---------       ---
Bill            Clinton         Harlem


Who said"Test Scores Can Be Used ...."
```

Cursor Control	ASCII value	Formatters
\n	010	newline (starts a new line)
\t	009	tab (moves cursor 8 spaces to right)
\r	013	carriage return (moves cursor to the beginning of the next line)
\b	008	backspace (moves cursor one space to left)
\f	012	form feed (advances printer paper one page)
\v	011	vertical tab
\’	039	single quote
\0	000	null character
\\	092	backslash
\”	034	double quote
\?	063	question



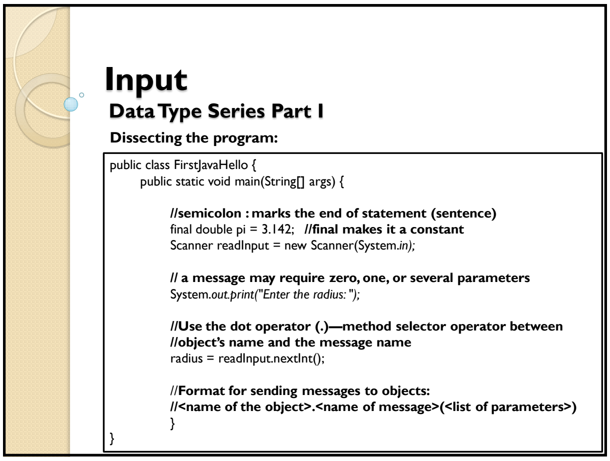
```
Hello Studentourse
Hello World

First Name      Last Name       City
-----------     ---------       ---
Bill            Clinton         Harlem

Who said"Test Scores Can Be Used ...."
```

Input

Data Type Series Part I

Dissecting the program:

```
import java.util.Scanner; // where to find the class needed later in the program
                              // Scanner object provides input  functionality
public class FirstJavaHello {
        public static void main(String[] args) {
                //declare, define and initialize primitive data types
                int    radius;          //a variable, can hold different value when run
                double area;
                final double pi = 3.142;
                // instantiate the scanner object, readInput is reference
                // (a variable) to scanner
                Scanner readInput = new Scanner(System.in);

                System.out.print("Enter the radius: ");
                radius = readInput.nextInt();  //use the reference and send message
                                                 //by calling a method nextInt()
                ……
        }
}
```

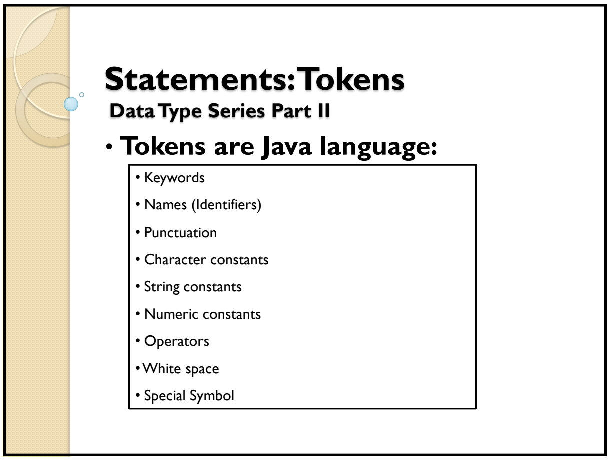


Input

Data Type Series Part I

Dissecting the program:

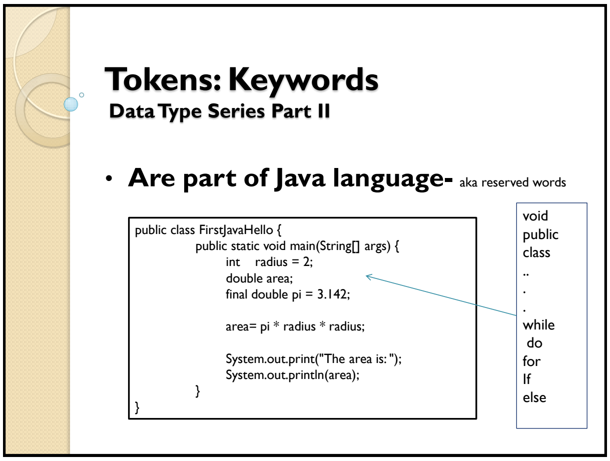
```
public class FirstJavaHello {
      public static void main(String[] args) {

            //semicolon : marks the end of statement (sentence)
            final double pi = 3.142;   //final makes it a constant
            Scanner readInput = new Scanner(System.in);

            // a message may require zero, one, or several parameters
            System.out.print("Enter the radius: ");

            //Use the dot operator (.)—method selector operator between
            //object's name and the message name
            radius = readInput.nextInt();

            //Format for sending messages to objects:
            //<name of the object>.<name of message>(<list of parameters>)
            }
}
```



Statements: Tokens

Data Type Series Part II

- **Tokens are Java language:**
  - Keywords
  - Names (Identifiers)
  - Punctuation
  - Character constants
  - String constants
  - Numeric constants
  - Operators
  - White space
  - Special Symbol



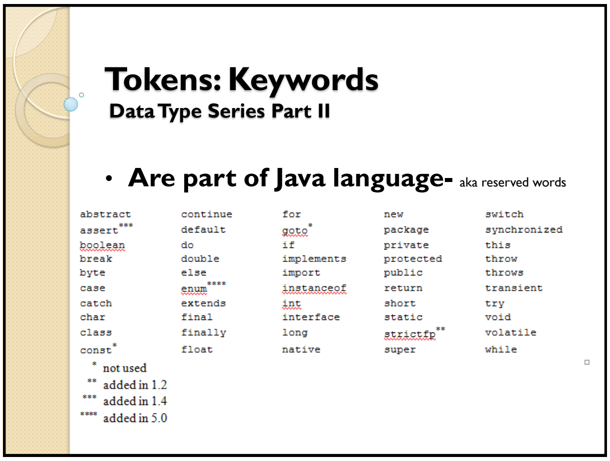
Tokens: Keywords

Data Type Series Part II

- **Are part of Java language-** aka reserved words

```
public class FirstJavaHello {
            public static void main(String[] args) {
                  int    radius = 2;
                  double area;
                  final double pi = 3.142;

                  area= pi * radius * radius;

                  System.out.print("The area is: ");
                  System.out.println(area);
            }
}
```

void
public
class
..
.
.
while
do
for
If
else



Tokens: Keywords

Data Type Series Part II

- **Are part of Java language-** aka reserved words

abstract
assert***
boolean
break
byte
case
catch
char
class
const*

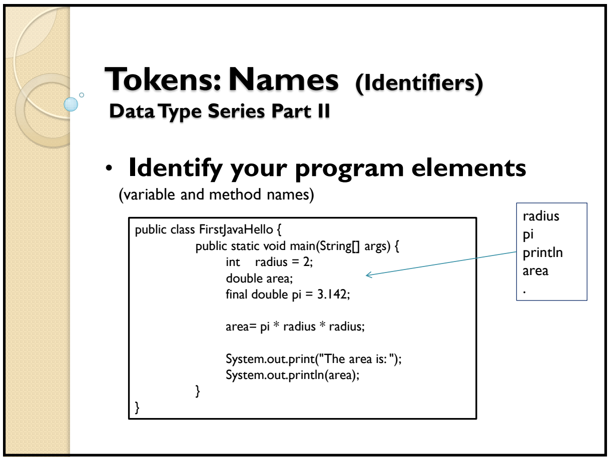
continue
default
do
double
else
enum****
extends
final
finally
float

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

* not used
** added in 1.2
*** added in 1.4
**** added in 5.0



Tokens: Names (Identifiers)

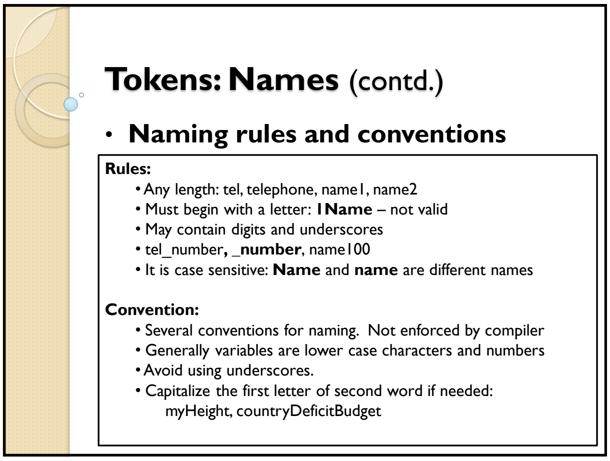
Data Type Series Part II

- **Identify your program elements**

(variable and method names)

```
public class FirstJavaHello {
            public static void main(String[] args) {
                  int    radius = 2;
                  double area;
                  final double pi = 3.142;

                  area= pi * radius * radius;

                  System.out.print("The area is: ");
                  System.out.println(area);
            }
}
```

radius
pi
println
area
.



Tokens: Names (contd.)

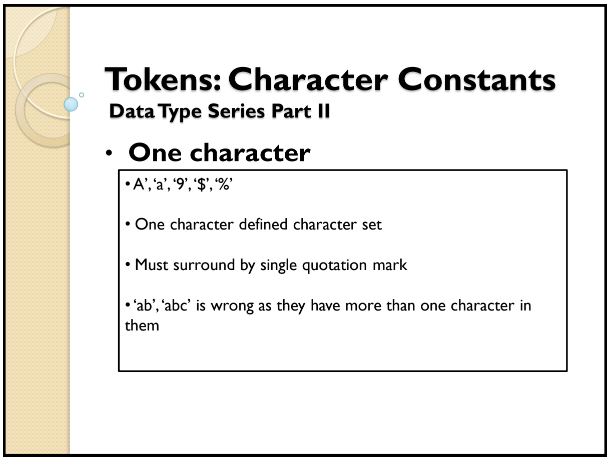
- **Naming rules and conventions**

Rules:

- Any length: tel, telephone, name1, name2
- Must begin with a letter: **1Name** – not valid
- May contain digits and underscores
- tel_number**, _number**, name100
- It is case sensitive: **Name** and **name** are different names

Convention:

- Several conventions for naming. Not enforced by compiler
- Generally variables are lower case characters and numbers
- Avoid using underscores.
- Capitalize the first letter of second word if needed:
  myHeight, countryDeficitBudget

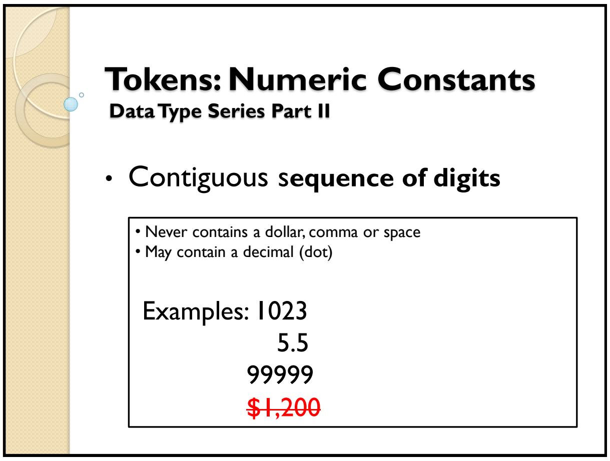


Tokens: Character Constants

Data Type Series Part II

- **One character**

- A', 'a', '9', '$', '%'
- One character defined character set
- Must surround by single quotation mark
- 'ab', 'abc' is wrong as they have more than one character in them



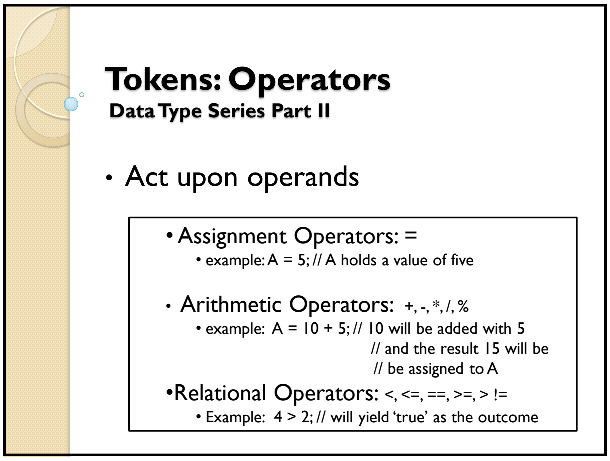
Tokens: Numeric Constants

Data Type Series Part II

- Contiguous sequence of digits

- Never contains a dollar, comma or space
- May contain a decimal (dot)

Examples: 1023
5.5
99999
~~$1,200~~

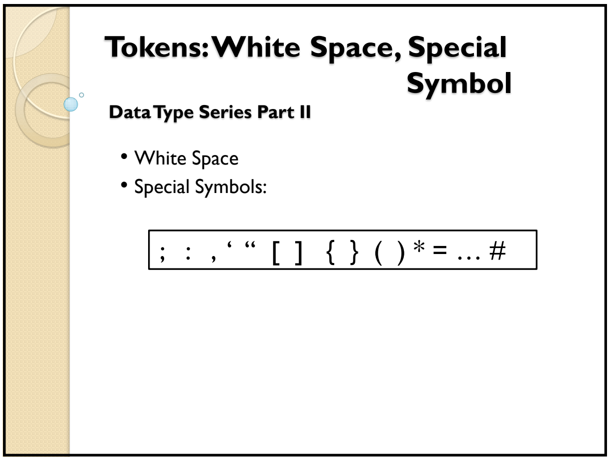


Tokens: Operators

Data Type Series Part II

- Act upon operands

- Assignment Operators: =
  - example: A = 5; // A holds a value of five
- Arithmetic Operators: +, -, *, /, %
  - example: A = 10 + 5; // 10 will be added with 5 // and the result 15 will be // be assigned to A
- Relational Operators: <, <=, ==, >=, > !=
  - Example: 4 > 2; // will yield 'true' as the outcome

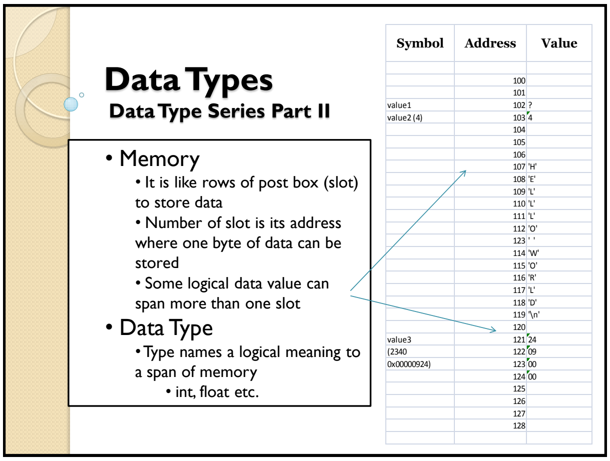


Tokens: White Space, Special Symbol

Data Type Series Part II

- White Space
- Special Symbols:

; : , ' " [ ] { } ( ) * = … #

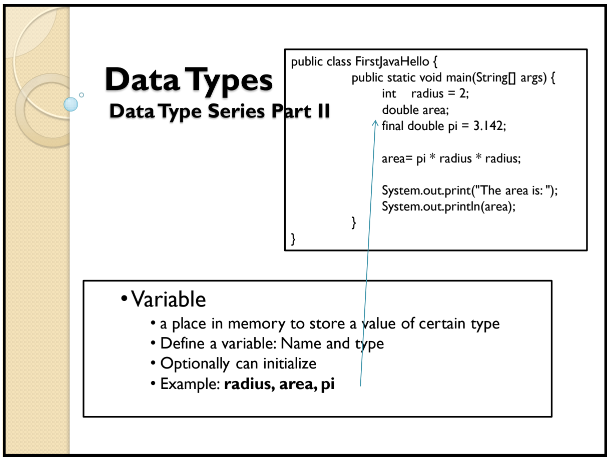


Data Types

Data Type Series Part II

- Memory
  - It is like rows of post box (slot) to store data
  - Number of slot is its address where one byte of data can be stored
  - Some logical data value can span more than one slot
- Data Type
  - Type names a logical meaning to a span of memory
    - int, float etc.

Symbol	Address	Value
	100	
	101	
value1	102	?
value2 (4)	103	4
	104	
	105	
	106	
	107	'H'
	108	'E'
	109	'L'
	110	'L'
	111	'L'
	112	'O'
	123	' '
	114	'W'
	115	'O'
	116	'R'
	117	'L'
	118	'D'
	119	'\n'
	120	
value3	121	24
(2340	122	09
0x00000924)	123	00
	124	00
	125	
	126	
	127	
	128	



Data Types

Data Type Series Part II

```
public class FirstJavaHello {
        public static void main(String[] args) {
            int    radius = 2;
            double area;
            final double pi = 3.142;

            area= pi * radius * radius;

            System.out.print("The area is: ");
            System.out.println(area);
        }
}
```

- Variable
  - a place in memory to store a value of certain type
  - Define a variable: Name and type
  - Optionally can initialize
  - Example: **radius, area, pi**

Data Types

Data Type Series Part II

```
public class FirstJavaHello {
                public static void main(String[] args) {
                        int     radius = 2;
                        double area;
                        final double pi = 3.142;

                        area= pi * radius * radius;

                        System.out.print("The area is: ");
                        System.out.println(area);
        }
```

- Guarantees values and precision
- Allows flexibility for programmers
- Fractional part can be 10e-38 to 10e38

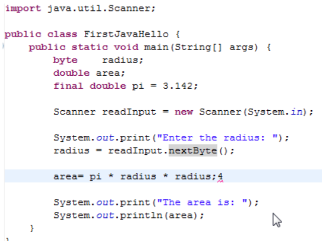
Type	Explanation
byte	A 8-bit (1-**byte**) integer value (-128 to 127)
short	A 16-bit (2-**byte**) integer value (-32768 to 32767)
int	A 32-bit (4-**byte**) integer value (-2147483648 to 2147483647)
long	A 64-bit (8-**byte**) integer value (-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807)
float	A 32-bit (4-**byte**) floating-point value (7 digits precision)
double	A 64-bit (8-**byte**) floating-point value (14 digits precision)
char	A 16-bit character **using** the Unicode encoding scheme
boolean	A **true** or **false** value

Data Types

Data Type Series Part II

- Byte: Used for small numbers and characters

  -127 to 128



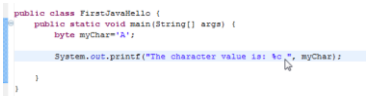
```
import java.util.Scanner;

public class FirstJavaHello {
    public static void main(String[] args) {
        byte    radius;
        double area;
        final double pi = 3.142;

        Scanner readInput = new Scanner(System.in);

        System.out.print("Enter the radius: ");
        radius = readInput.nextByte();

        area= pi * radius * radius;4

        System.out.print("The area is: ");
        System.out.println(area);
    }
```

Data Types

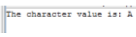
Data Type Series Part II

- Byte: Used for small numbers and characters

  -127 to 128



```
public class FirstJavaHello {
    public static void main(String[] args) {
        byte myChar='A';

        System.out.printf("The character value is: %c ", myChar);

    }
}
```

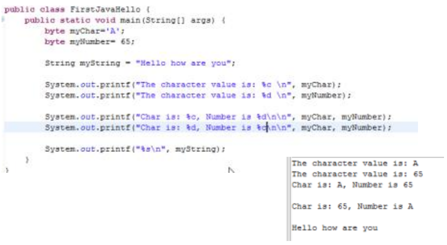


```
The character value is: A
```

Data Types

Data Type Series Part II

- Mixing number and characters



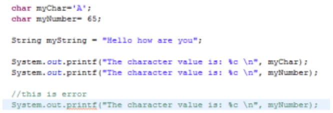
```
public class FirstJavaHello {
    public static void main(String[] args) {
        byte myChar='A';
        byte myNumber= 65;

        String myString = "Hello how are you";

        System.out.printf("The character value is: %c \n", myChar);
        System.out.printf("The character value is: %d \n", myNumber);

        System.out.printf("Char is: %c, Number is %d\n\n", myChar, myNumber);
        System.out.printf("Char is: %d, Number is %c\n\n", myChar, myNumber);

        System.out.printf("%s\n", myString);
    }
}
```

```
The character value is: A
The character value is: 65
Char is: A, Number is 65

Char is: 65, Number is A

Hello how are you
```

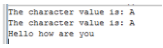
Data Types

Data Type Series Part II

- Character: To support Unicode

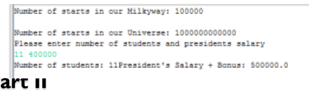


```
char myChar='A';
char myNumber= 65;

String myString = "Hello how are you";

System.out.printf("The character value is: %c \n", myChar);
System.out.printf("The character value is: %c \n", myNumber);

//this is error
System.out.printf("The character value is: %c \n", myNumber);
```



```
The character value is: A
The character value is: A
Hello how are you
```

Data Types

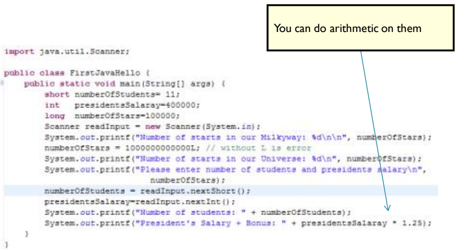


```
Number of starts in our Milkyway: 100000

Number of starts in our Universe: 1000000000000
Please enter number of students and presidents salary
11 400000
Number of students: 11President's Salary + Bonus: 500000.0
```

Data Type Series Part II

- Regular, Short and Long Integers

You can do arithmetic on them



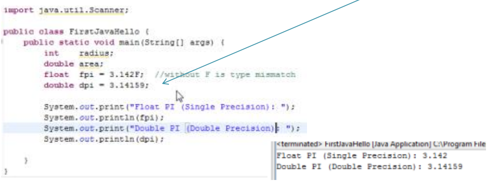
```
import java.util.Scanner;

public class FirstJavaHello {
    public static void main(String[] args) {
        short numberOfStudents= 11;
        int   presidentsSalaray=400000;
        long  numberOfStars=100000;
        Scanner readInput = new Scanner(System.in);
        System.out.printf("Number of starts in our Milkyway: %d\n\n", numberOfStars);
        numberOfStars = 1000000000000L; // without L is error
        System.out.printf("Number of starts in our Universe: %d\n", numberOfStars);
        System.out.printf("Please enter number of students and presidents salary\n",
                          numberOfStars);
        numberOfStudents = readInput.nextShort();
        presidentsSalaray=readInput.nextInt();
        System.out.printf("Number of students: " + numberOfStudents);
        System.out.printf("President's Salary + Bonus: " + presidentsSalaray * 1.25);
    }
}
```

Data Types

Data Type Series Part II

- Real numbers: Floating and Double

You can do arithmetic on them



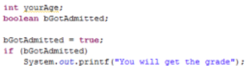
```
import java.util.Scanner;

public class FirstJavaHello {
    public static void main(String[] args) {
        int    radius;
        double area;
        float  fpi = 3.142F;  //without F is type mismatch
        double dpi = 3.14159;

        System.out.print("Float PI (Single Precision): ");
        System.out.println(fpi);
        System.out.print("Double PI (Double Precision): ");
        System.out.println(dpi);

    }
}
```

```
<terminated> FirstJavaHello [Java Application] C:\Program File
Float PI (Single Precision): 3.142
Double PI (Double Precision): 3.14159
```

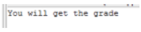
Data Types

Data Type Series Part II

- boolean



```
int yourAge;
boolean bGotAdmitted;

bGotAdmitted = true;
if (bGotAdmitted)
    System.out.printf("You will get the grade");
```



```
You will get the grade
```

Data Types
Data Type Series Part II

•Variables: memory location to store data that has a name. Like a post office box number.

- Format to declare variables:
  - data_type variable_name;
  - data_type vaiable_name1, variable_name2;
- Variables are implicitly initialized to zero

```
public class FirstJavaHello {
        public static void main(String[] args) {
                int  value1;              //declaration which is definition also
                int   value2, sum;    // multiple variables can be defined
        }
}
```

Data Types
Data Type Series Part II

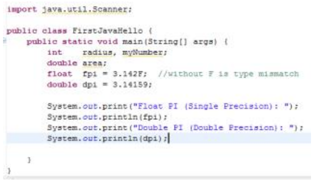
- Initializing Variables

- Declarations create variables, but do not provide a value
- Initialization provides the value:
  - **variable_name = expression**

```
public class FirstJavaHello {
        public static void main(String[] args) {
                int  value1;                    // declaration which is definition also
                value1 = 324;                 // assignment
                int   value2 = 6;              // definition which is initialization also
                                                   //explicitly initialize;
}
```

Assignment Operator:
Data Type Series Part III

- Assigns the value from right operand to the left
- ‘=‘ operator is used as assignment operator

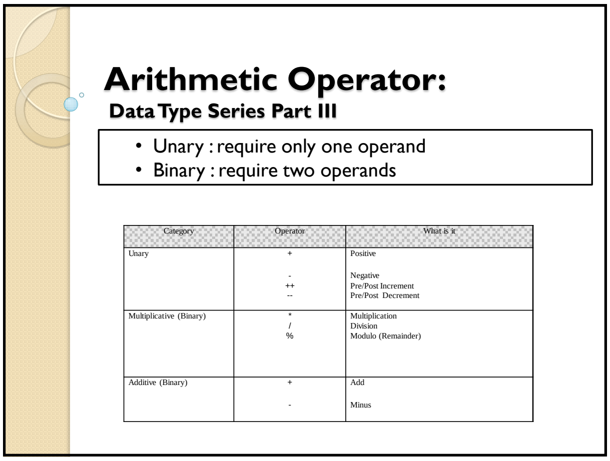


```
import java.util.Scanner;

public class FirstJavaHello {
    public static void main(String[] args) {
        int    radius, myNumber;
        double area;
        float  fpi = 3.142F;  //without F is type mismatch
        double dpi = 3.14159;

        System.out.print("Float PI (Single Precision): ");
        System.out.println(fpi);
        System.out.print("Double PI (Double Precision): ");
        System.out.println(dpi);

    }
}
```

Assignment Operator:
Data Type Series Part III

- Expression is combination of:
  operators & operands
- General form of expression is:
  Variable = Expression;

- A = B;
- myAge = 24;
- yourSalary = GetYourSalary();
- yourTakehomePay = yourSalary * yourTaxRate;
- int temp = 55;

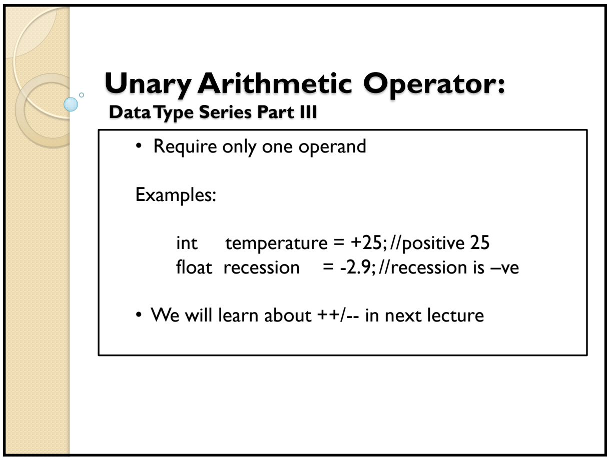


Arithmetic Operator:

Data Type Series Part III

- Unary : require only one operand
- Binary : require two operands

Category	Operator	What is it
Unary	+ - ++ --	Positive Negative Pre/Post Increment Pre/Post Decrement
Multiplicative (Binary)	* / %	Multiplication Division Modulo (Remainder)
Additive (Binary)	+ -	Add Minus



Unary Arithmetic Operator:

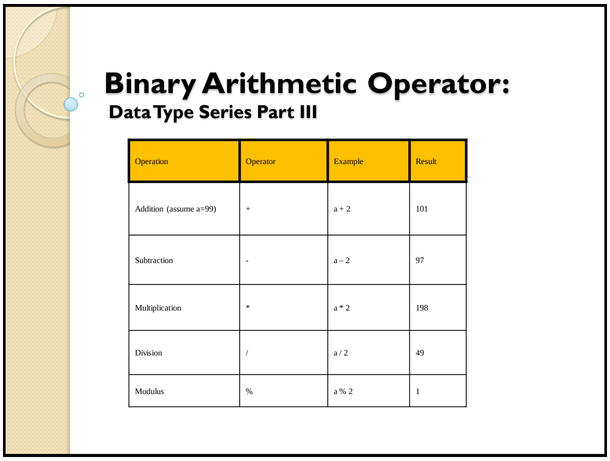
Data Type Series Part III

- Require only one operand

Examples:

```
int     temperature = +25; //positive 25
float  recession    = -2.9; //recession is –ve
```

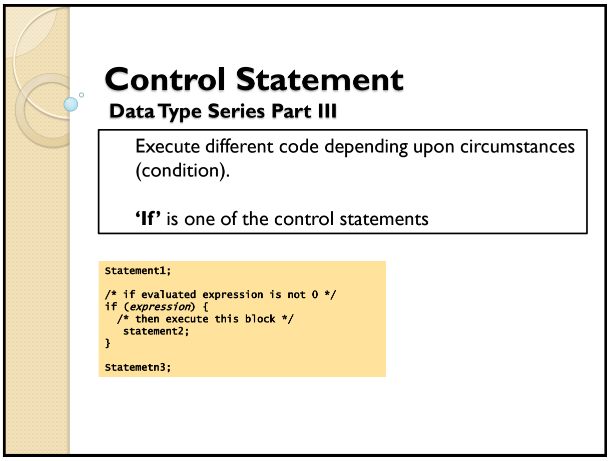
- We will learn about ++/-- in next lecture



Binary Arithmetic Operator:

Data Type Series Part III

Operation	Operator	Example	Result
Addition (assume a=99)	+	a + 2	101
Subtraction	-	a – 2	97
Multiplication	*	a * 2	198
Division	/	a / 2	49
Modulus	%	a % 2	1



Control Statement

Data Type Series Part III

Execute different code depending upon circumstances (condition).

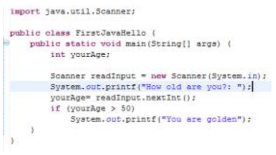
'If' is one of the control statements

```
Statement1;

/* if evaluated expression is not 0 */
if (expression) {
  /* then execute this block */
   statement2;
}

Statemetn3;
```

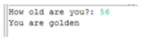
Control Statement

Data Type Series Part III

Example of ‘**if**’ statement



```
import java.util.Scanner;

public class FirstJavaHello {
    public static void main(String[] args) {
        int yourAge;

        Scanner readInput = new Scanner(System.in);
        System.out.printf("How old are you?: ");
        yourAge= readInput.nextInt();
        if (yourAge > 50)
            System.out.printf("You are golden");
    }
}
```



```
How old are you?: 56
You are golden
```

Control Statement

Data Type Series Part III

‘**if**’ has another form

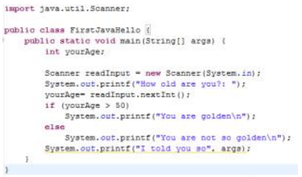
if ..else

```
Statement1;

/* if evaluated expression is not 0 */
if (expression) {
  /* then execute this block */
   statement2;
}
else
{
   statement3;
}

Statemetnt4;
```

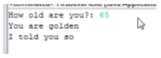
Control Statement

Data Type Series Part III

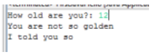
Example of ‘**if .. else**’ statement



```
import java.util.Scanner;

public class FirstJavaHello {
    public static void main(String[] args) {
        int yourAge;

        Scanner readInput = new Scanner(System.in);
        System.out.printf("How old are you?: ");
        yourAge= readInput.nextInt();
        if (yourAge > 50)
            System.out.printf("You are golden\n");
        else
            System.out.printf("You are not so golden\n");
        System.out.printf("I told you so", args);
    }
}
```



```
How old are you?: 65
You are golden
I told you so
```

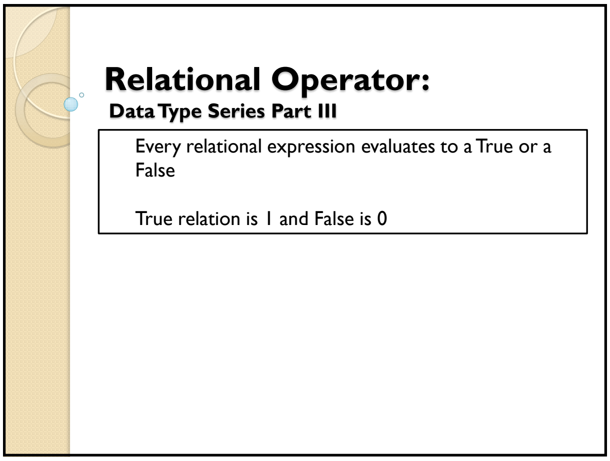


```
How old are you?: 12
You are not so golden
I told you so
```

Relational Operator:

Data Type Series Part III

Operation	Operator	Example	Meaning
Equality relational	= = !=	X == Y X != Y	X is equal to Y X is not equal to Y
Relational	< <= > >=	X < Y X <= Y X > Y X >= Y	X is less than Y X is less than or equal to Y X is greater than y X is greater than or equal to Y

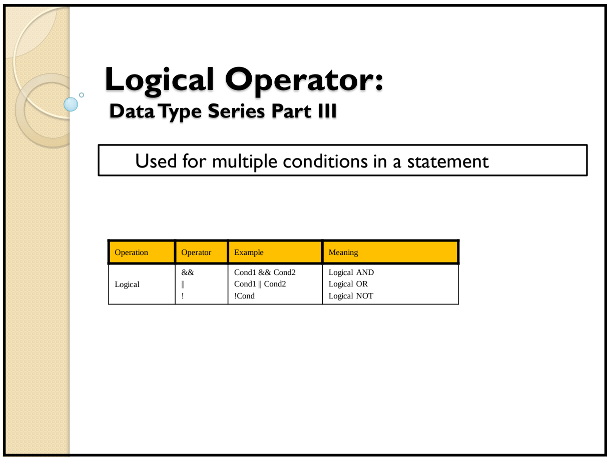


Relational Operator:

Data Type Series Part III

Every relational expression evaluates to a True or a False

True relation is 1 and False is 0

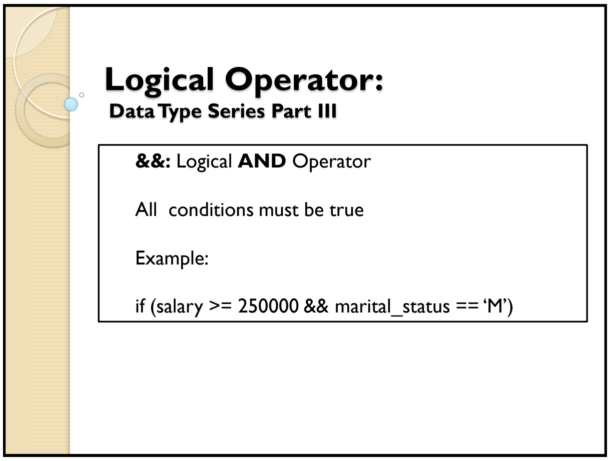


Logical Operator:

Data Type Series Part III

Used for multiple conditions in a statement

Operation	Operator	Example	Meaning
Logical	&& || !	Cond1 && Cond2 Cond1 || Cond2 !Cond	Logical AND Logical OR Logical NOT



Logical Operator:

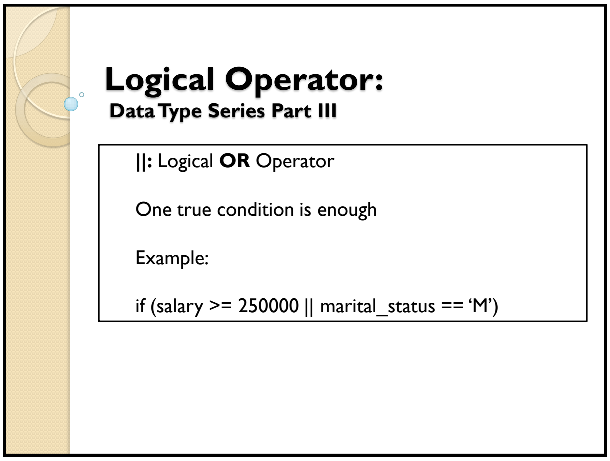
Data Type Series Part III

&&: Logical **AND** Operator

All conditions must be true

Example:

if (salary >= 250000 && marital_status == ‘M’)



Logical Operator:

Data Type Series Part III

||: Logical **OR** Operator

One true condition is enough

Example:

if (salary >= 250000 || marital_status == ‘M’)

Logical Operator:

Data Type Series Part III

!: Logical **NOT** Operator

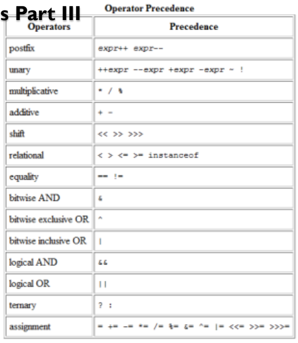
One true condition is enough

Example:

if (!(marital_status == ‘M’))

Operators:

Data Type Series Part III



Operator Precedence

Operators	Precedence
postfix	`expr++ expr--`
unary	`++expr --expr +expr -expr ~ !`
multiplicative	`* / %`
additive	`+ -`
shift	`<< >> >>>`
relational	`< > <= >= instanceof`
equality	`== !=`
bitwise AND	`&`
bitwise exclusive OR	`^`
bitwise inclusive OR	`|`
logical AND	`&&`
logical OR	`||`
ternary	`? :`
assignment	`= += -= *= /= %= &= ^= |= <<= >>= >>>=`

Data Type, IO, Operators

Data Type Series Part III

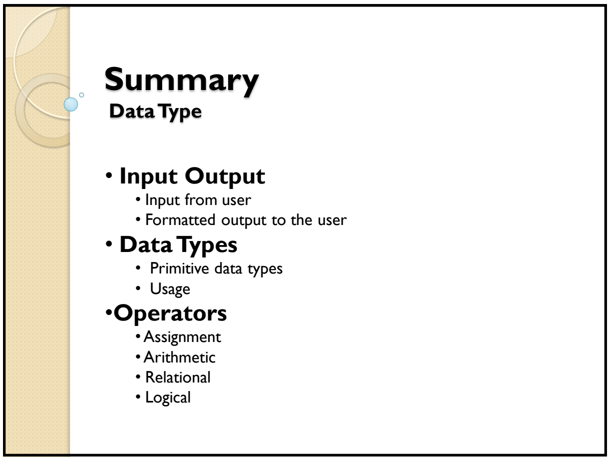
- **Demo**

```
import java.util.Scanner;

public class Hello {
  public static void main(String [] args) {
    int age;

    Scanner readInput = new Scanner(System.in);

    System.out.print("Please enter your age: ");
    age = readInput.nextInt();

    if (age >= 13)
       if (age <= 19)
         System.out.printf ("You are a teenager");
  }
}
```

Vocabulary We Used

Data Type Series

- ✓ byte, char, int, long, double, float, short, boolean
- ✓ package
- ✓ printf()
- ✓ Assignment, Arithmetic, Logical, Relational Operators
- ✓ Control statements
- ✓ if
- ✓ if .. else
- ✓ Variable, declare, implicit, explicit initialize



Summary

Data Type

- **Input Output**
  - Input from user
  - Formatted output to the user
- **Data Types**
  - Primitive data types
  - Usage
- **Operators**
  - Assignment
  - Arithmetic
  - Relational
  - Logical