FUNDAMENTALS PROGRAMMING STRUCTURES

A SIMPLE JAVA PROGRAM

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

COMMENTS

- //…
- /*…*/

DATA TYPES

- Java is a **strongly typed language**.
- There are eight **primitive types** in Java.

INTEGERS

Type	Storage Requirement
int	4 bytes
short	2 bytes
long	8 bytes
byte	1 byte

FLOATING-POINT TYPES

Type	Storage Requirement
float	4 bytes
double	8 bytes

THE CHARACTER TYPE

- Single quotes are used to denote char constants.
- Unicode characters have codes between 0 and 65535.

Escape Sequence	Name	Unicode Value
\b	backspace	\u0008
\t	tab	\u0009
\n	linefeed	\u000a
\r	carriage return	\u000d
\”	double quote	\u0022
\’	single quote	\u0027
\\	backslash	\u005c

THE BOOLEAN TYPE

- The boolean type has 2 values: **true**, **false**.

VARIABLES

- You declare a variable by placing the **type** first, followed by the **name** of the variable.

```
double salary;
int vacationDays;
long earthPopulation;
char yesChar;
boolean done;
```

ASSIGNMENTS & INITIALIZATION

- After you declare a variable, you must explicitly initialize it by means of an assignment statement - you can never use the values of uninitialized variables.
- You assign to a previously declared variable using the variable name on the left, an equal sign (=), and then some Java expression that has an appropriate value on the right.

```
int vacationDays; // this is a declaration
vacationDays = 12; // this is an assignment
```

INCREMENT & DECREMENT OPERATORS

- x++
- ++x
- x--
- --x

RATIONAL & BOOLEAN OPERATORS

- ==
- !=
- >,>=
- <,<=
- &&
- ||
- ?

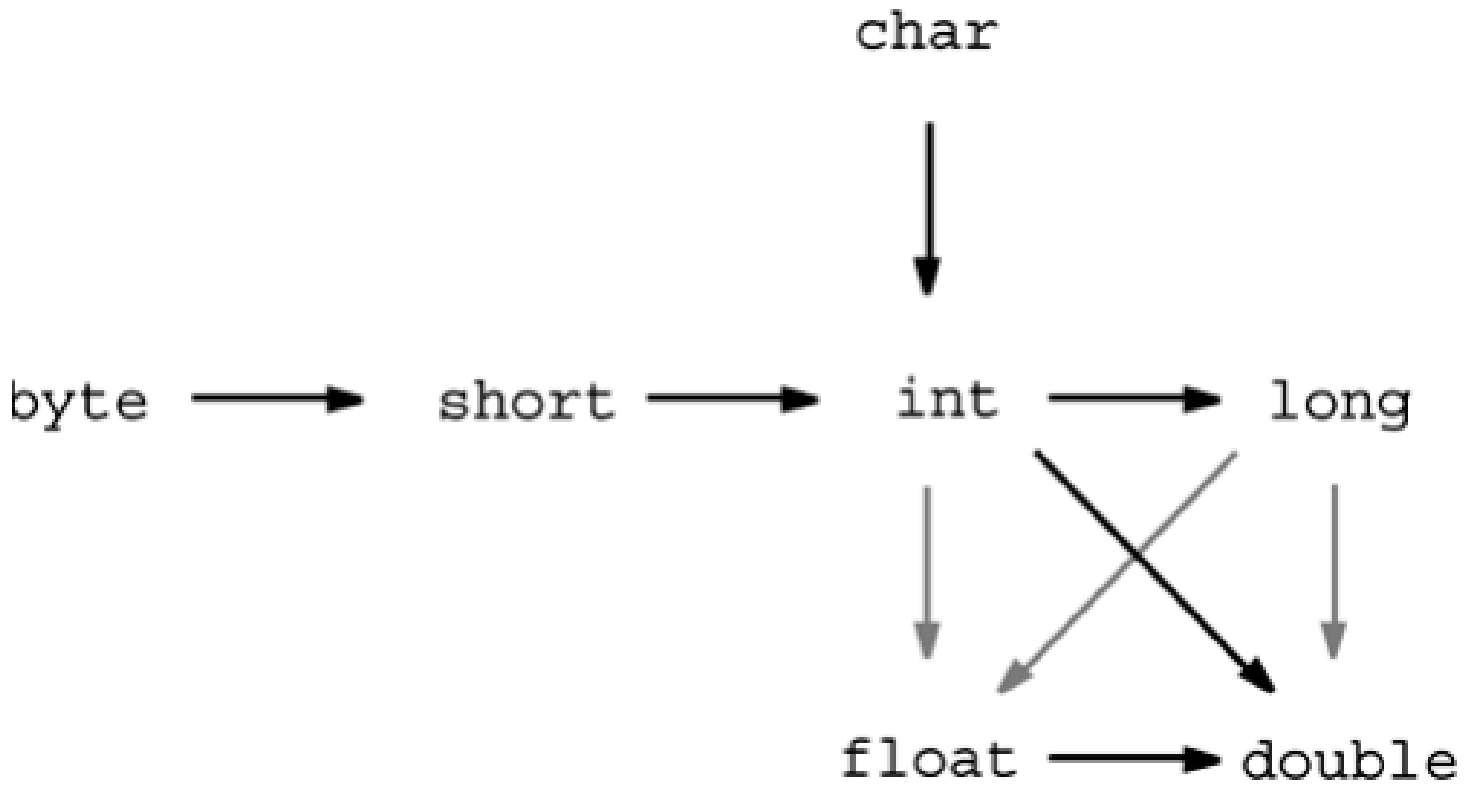
BITWISE OPERATORS

- &
- |
- ^
- ~

MATHEMATICAL FUNCTIONS & CONSTANTS

- Math.sqrt()
- Math.pow()
- Math.sin()
- Math.cos()
- Math.tan()
- Math.log()
- Math.PI
- Math.E
- …

CONVERSIONS BETWEEN NUMERIC TYPES



CASTS

- Conversions where loss of information is possible are done by means of **casts**.

```
double x = 9.997;
int nx = (int)x;
```

STRING (1)

- Strings are sequences of characters.

```
String e = ""; // an empty string
String greeting = "Hello";
```

STRING (2)

- **Concatenate**: Java allows you to use the + sign to join (concatenate) two strings together.

```
String expletive = "Expletive";
String PG13 = "deleted";
String message = expletive + PG13;
```

STRING (3)

- String denotes a sequence of Unicode characters. It is possible to get at individual characters of a string.

```
String greeting = "Hello";
char last = greeting.charAt(4); // fourth is 'o'
```

- The String class gives no methods that let you change a character in an existing string.

STRING (4)

- **Equality**: To test whether or not two strings are equal, use the equals method.

```
s.equals(t)
"Hello".equalsIgnoreCase("hello")
```

- Do not use the == operator to test if two strings are equal.

STRING (5)

- char charAt(int index)
- int compareTo(String other)
- boolean endsWith(String suffix)
- boolean equals(Object other)
- boolean equalsIgnoreCase(String other)
- int indexOf(String str)
- int indexOf(String str, int fromIndex)

STRING (6)

- int lastIndexOf(String str)
- int lastIndexOf(String str, int fromIndex)
- int length()
- String replace(char oldChar, char newChar)
- boolean startsWith(String prefix)
- String substring(int beginIndex)
- String substring(int beginIndex, int endIndex)

STRING (7)

- String toLowerCase()
- String toUpperCase()
- String trim()

READING INPUT

- Console.
- Graphic.

FORMATTING OUTPUT

The **NumberFormat** class in the **java.text** package has three methods that yield standard formatters for:

- numbers
- currency values
- percentage values

THE NUMBERFORMAT CLASS

- static NumberFormat getCurrencyInstance()
- static NumberFormat getNumberInstance()
- static NumberFormat getPercentInstance()
- void setMaximumFractionDigits(int digits)
- void setMaximumIntegerDigits(int digits)
- void setMinimumFractionDigits(int digits)
- void setMinimumIntegerDigits(int digits)

BLOCK SCOPE

- A block or compound statement is any number of simple Java statements that are surrounded by a pair of braces.
- Blocks define the scope of your variables.
- Blocks can be nested inside another.
- It is not possible to declare identically named variables in two nested blocks.

CONDITION STATEMENTS (1)

- If statement:

```
if (condition){
    statement1;
    statement2;
    …
}
```

CONDITION STATEMENTS (2)

- If/else statement:

```
if (condition1)        {
    …
}
else if (contidion2) {
    …
}…
else{
    …
}
```

INDETERMINATE LOOPS (1)

- While statement:

```
while (condition)    {
    statement1
    statement2
    …
}
```

INDETERMINATE LOOPS (2)

- Do/while statement:

```
do{
    statement1
    statement2
    …
}
while(condition);
```

DETERMINATE LOOPS

- For statement:

```
for(statement1;expression1;expression2){
    statement2;
    …
}
```

MULTIPLE SELECTION

- Switch statement:

```
switch(input){
    case value1:
        …
        break;
    …
    default
        …
        break;
}
```

BREAKING CONTROL FLOW

- break.
- continue.

ARRAYS

- An array is a data structure that stores a collection of values of thesame type. You access each individual value through an integer index.
- You declare an array variable by specifying the array type - which is the element type followed by [] - and the array variable name.
- You use the new operator to create the array.

ARRAY INITIALIZERS & ANONYMOUS ARRAYS

- Java has a shorthand to create an array object and supply initial values at the same time.

```
int[] smallPrimes = { 2, 3, 5, 7, 11, 13 };
```

- You can even initialize an anonymous array.

```
new int[] { 17, 19, 23, 29, 31, 37 }
```

COPYING ARRAYS (1)

- You can copy one array variable into another, but then both variables refer to the same array.

```
int[] luckyNumbers = smallPrimes;
```

COPYING ARRAYS (2)

- If you actually want to copy all values of one array into another, you have to use the arraycopy method in the System class.

System.arraycopy(from, fromIndex, to, toIndex, count);