

2.2 Primitive Data Types

```
public class Addition {
    public static void main(String[] args) {
        int a = Integer.parseInt(args[0]);
        int b = Integer.parseInt(args[1]);
        int c = a + b;
        System.out.println(c);
    }
}
```

"Primitive" Data Types

Data type.

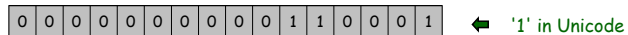
- A set of values.
- A set of operations on those values.

Data Type	Description	Examples	Common Operations
char	character	'A' '@'	compare
String	sequence of characters	"Hello, World" "CS is fun"	concatenation, compare
int	integer	17 12345	add, subtract, multiply, remainder
double	floating point number	3.1415 2.17	add, subtract, multiply, divide
boolean	truth value	true false	not, and, or, xor

Text

The `String` data type.

- A sequence of Unicode characters.
- Each character internally stored as a sequence of 16 bits:



- Not technically a primitive type, but special language support.
- **Ex:** generate subdivisions of a ruler.

```
public class Ruler {
    public static void main(String[] args) {
        String ruler1 = "1 ";
        String ruler2 = ruler1 + "2 " + ruler1;
        String ruler3 = ruler2 + "3 " + ruler2;
        String ruler4 = ruler3 + "4 " + ruler3;
        String ruler5 = ruler4 + "5 " + ruler4;
        System.out.println(ruler5);
    }
}
```

1
1 2 1
1 2 1 3 1 2 1
↑
string concatenation

Ruler

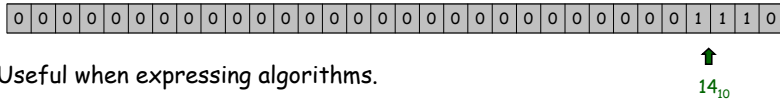
```
% javac Ruler.java
% java Ruler
1 2 1 3 1 2 1 4 1 2 1 3 1 2 1 5 1 2 1 3 1 2 1 4 1 2 1 3 1 2 1
```



Integers

The `int` data type.

- Represents integers between -2^{31} and $2^{31} - 1$.
- Internally stored as a sequence of 32 bits:



- Useful when expressing algorithms.

```
public class IntOps {
    public static void main(String[] args) {
        int a = 1234;
        int b = 99;
        int sum = a + b;    1333
        int prod = a * b;  122166
        int quot = a / b;  12
        int rem = a % b;   46
    }
}
```

1234 = 12*99 + 46

Floating Point Numbers

The `double` data type.

- Represents floating point numbers.
- Internally stored as a sequence of 64 bits using scientific notation.
- Useful in scientific applications.
- Ex: solve quadratic equation $x^2 + bx + c = 0$.

$$\text{roots} = \frac{-b \pm \sqrt{b^2 - 4c}}{2}$$

```
public class Quadratic {
    public static void main(String[] args) {
        double b = Double.parseDouble(args[0]);
        double c = Double.parseDouble(args[1]);

        double sqrt = Math.sqrt(b*b - 4.0*c);
        double root1 = (-b + sqrt) / 2.0;
        double root2 = (-b - sqrt) / 2.0;

        System.out.println(root1);
        System.out.println(root2);
    }
}
```

Annotations:
 ← read coefficients from command line (points to args[0] and args[1])
 ← calculate roots (points to the sqrt and root calculations)
 ← print them out (points to the println statements)

Command Line Arguments

Command line arguments.

- Simple method for processing a small amount of user input.

```
% java Quadratic -3.0 2.0
2.0
1.0
command line arguments

% java Quadratic -1.0 -1.0
1.618033988749895
-0.6180339887498949
golden ratio

% java Quadratic 1.0 1.0
NaN
NaN
not a number

% java Quadratic 1.0 hello
java.lang.NumberFormatException: hello

% java Quadratic 1.0
java.lang.ArrayIndexOutOfBoundsException
```

$x^2 - 3x + 2$

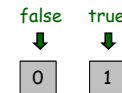
$x^2 - x - 1$

$x^2 + x + 1$

Booleans and Comparisons

The `boolean` data type.

- Has two values: `true` or `false`.
- Internally represented as one bit.
- Useful to control logic and flow of a program.



Logic Operators			
a	b	a && b	a b
false	false	false	false
false	true	false	true
true	false	false	true
true	true	true	true

Logic Operators	
a	!a
false	true
true	false

Comparison Operators			
op	Description	true	false
==	equal	2 == 2	2 == 3
!=	not equal	2 != 3	2 != 2
<	less	2 < 3	3 < 2
<=	less or equal	2 <= 3	3 <= 2
>	greater	3 > 2	2 > 3
>=	greater or equal	3 >= 3	2 >= 3

Booleans and Comparisons

Is year y a leap year?

- Yes if divisible by 400, or divisible by 4 but not 100.

```
public class LeapYear {
    public static void main(String[] args) {
        int y = Integer.parseInt(args[0]);
        boolean isLeapYear;

        // divisible by 4 but not 100
        isLeapYear = (y % 4 == 0) && (y % 100 != 0);

        // or divisible by 400
        isLeapYear = isLeapYear || (y % 400 == 0);

        System.out.println(isLeapYear);
    }
}
```

```
% java LeapYear 2004
true
% java LeapYear 1900
false
% java LeapYear 2000
true
```

9

Type Conversion

Type conversion: convert from one type of data to another.

- Automatic: no loss of precision; or with Strings.
- Explicit: cast; or method.

Ex: generate a pseudo-random number between 0 and N-1.

- `Math.random` outputs a double between 0.0 and 1.0.

```
public class RandomInteger {
    public static void main(String[] args) {
        int N = Integer.parseInt(args[0]);
        double r = Math.random(); // String to int (method)
        int n = (int) (r * N);
        // double to int (cast)      int to double (automatic)
        System.out.println("random integer is: " + n);
        // int to String (automatic)
    }
}
```

10

Summary

A data type is a set of values and operations on those values.

- String: text processing.
- double, int: mathematical calculator.
- boolean: basis for decision making.

Be aware.

- Declare type of values.
- Convert between types when necessary.

11