Session 02:
Variables and Calculations

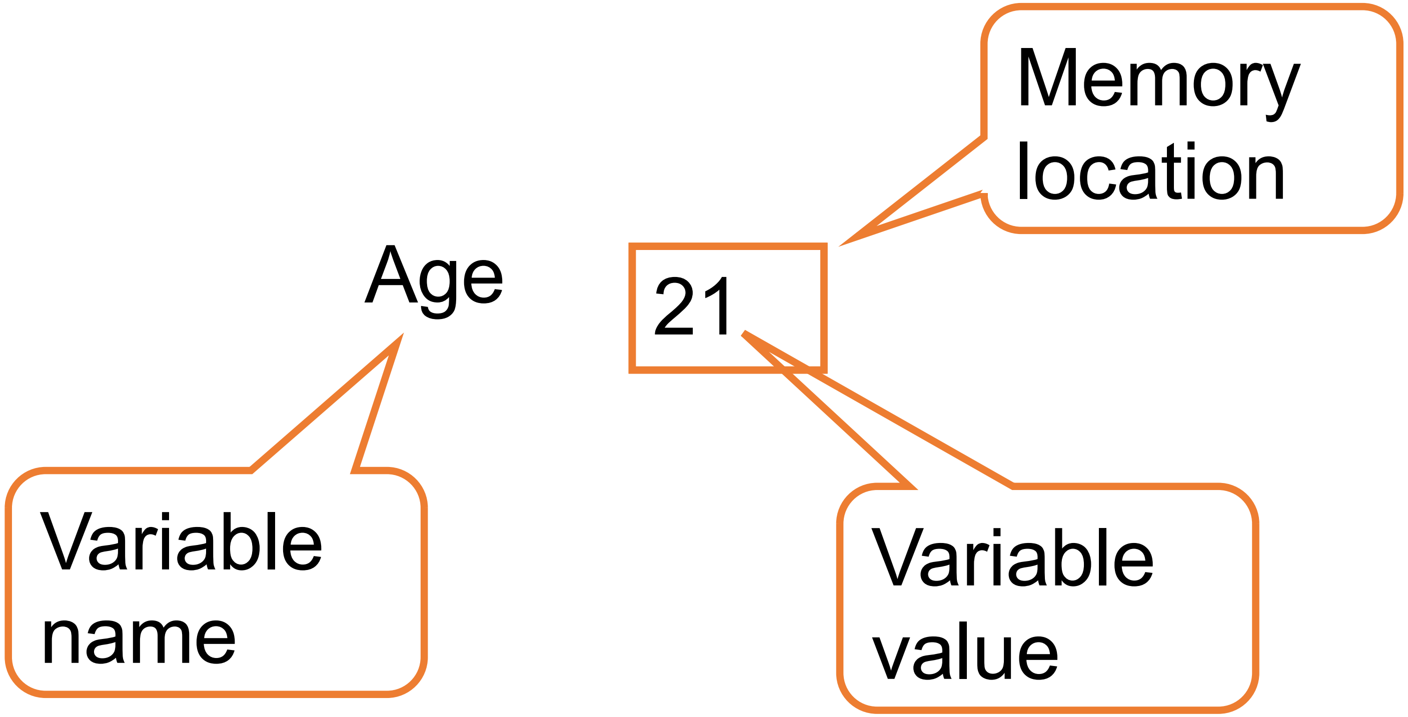
Dr Tomasz Kurowski
t.j.kurowski@cranfield.ac.uk

Lecture outline

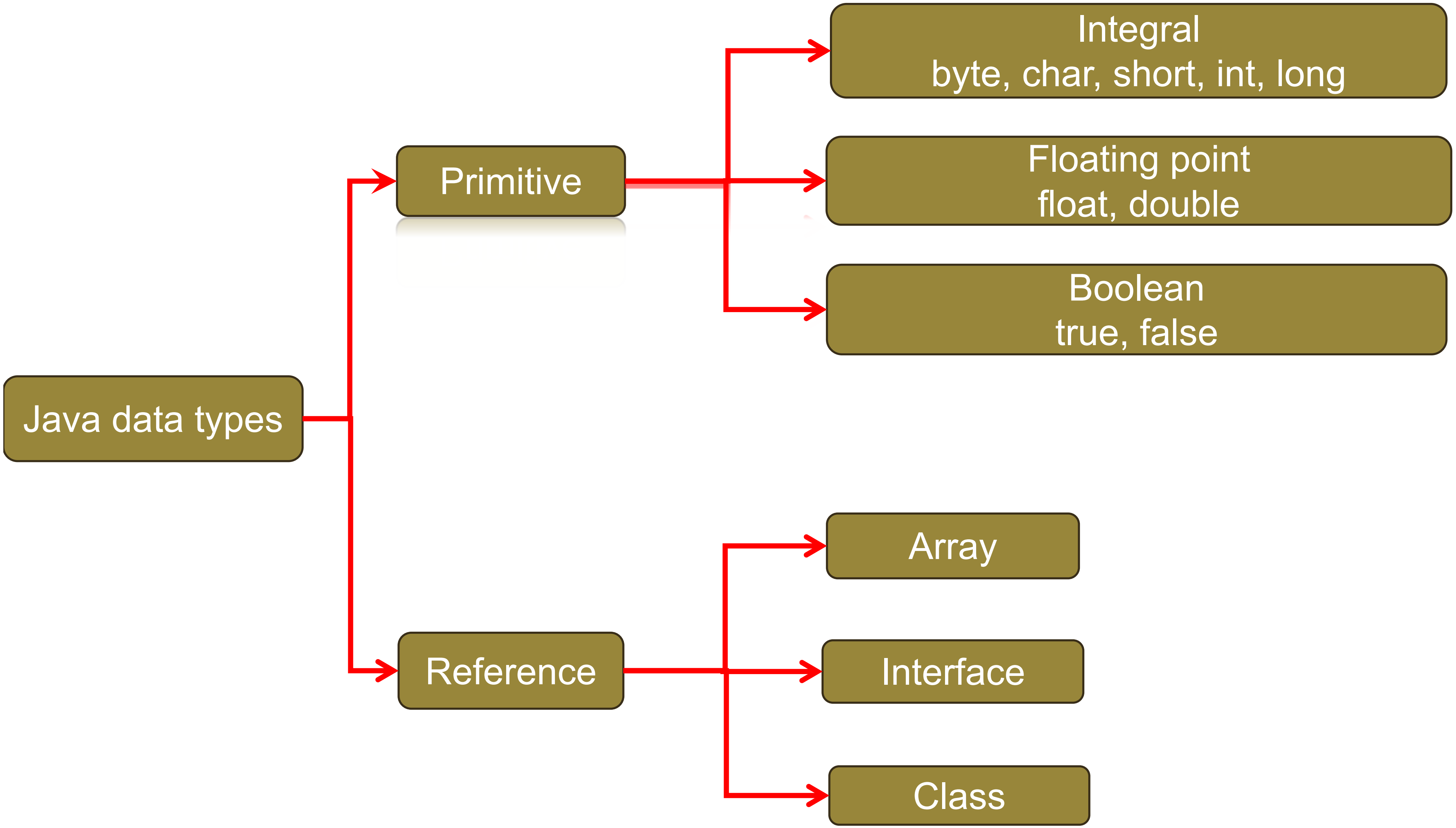
- Numeric data types
  - Byte, short, int, long, float
  - Declaring numeric data
  - Arithmetic operations
- Character data types

Variables

- A place in the computer’s memory to store a number.



Java data types



Java data types

Primitive

Integral
byte, char, short, int, long

Floating point
float, double

Boolean
true, false

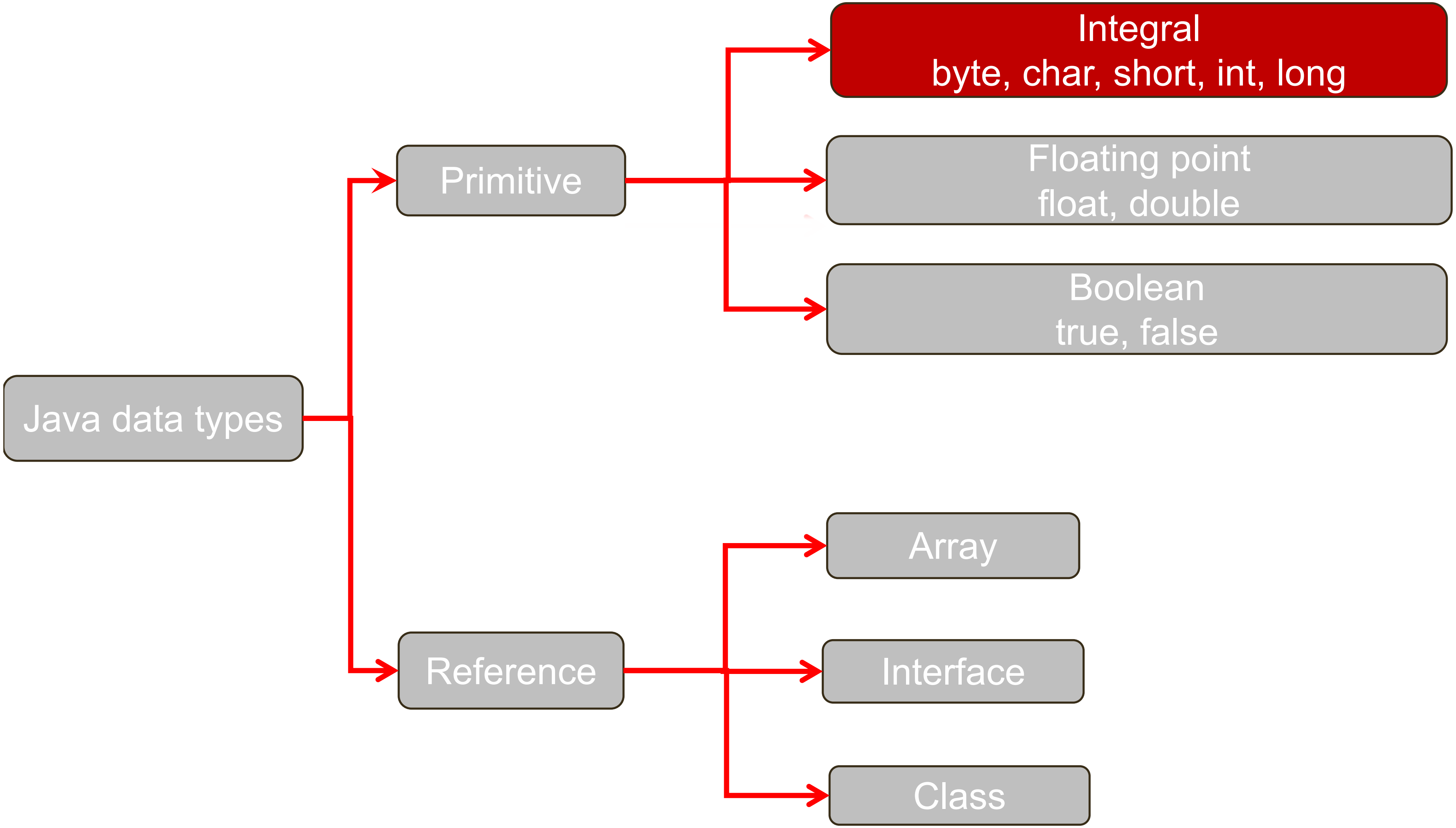
Reference

Array

Interface

Class

Java data types



Java data types

Primitive

Integral
byte, char, short, int, long

Floating point
float, double

Boolean
true, false

Reference

Array

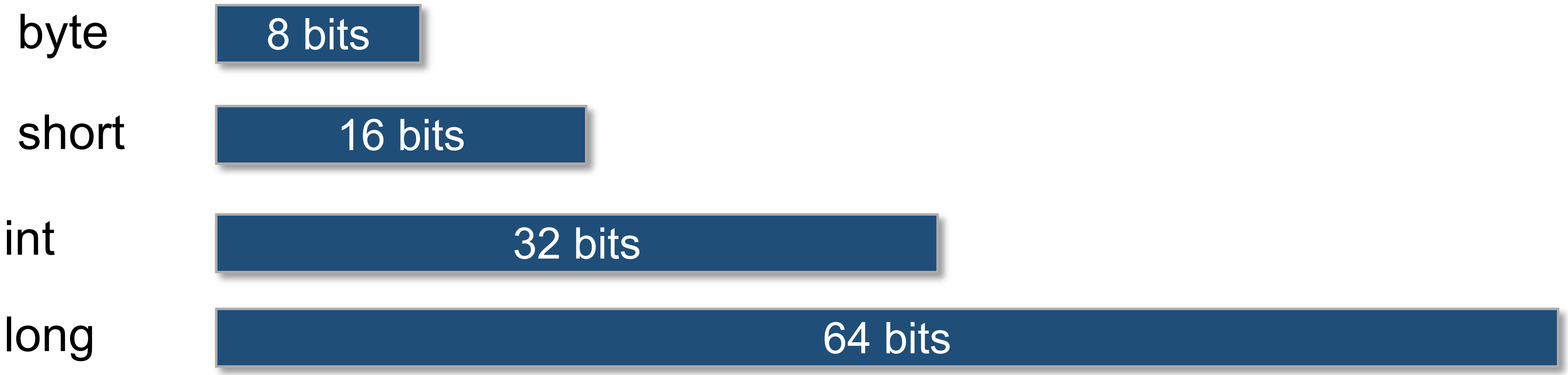
Interface

Class

Integral Types

- byte, short, int and long → refer to integer values
  - 22 16 546
- Negative integers:
  - -45 -925

Integral Types



- int is the most commonly used. It ranges from -2147483648 to + 2147483647

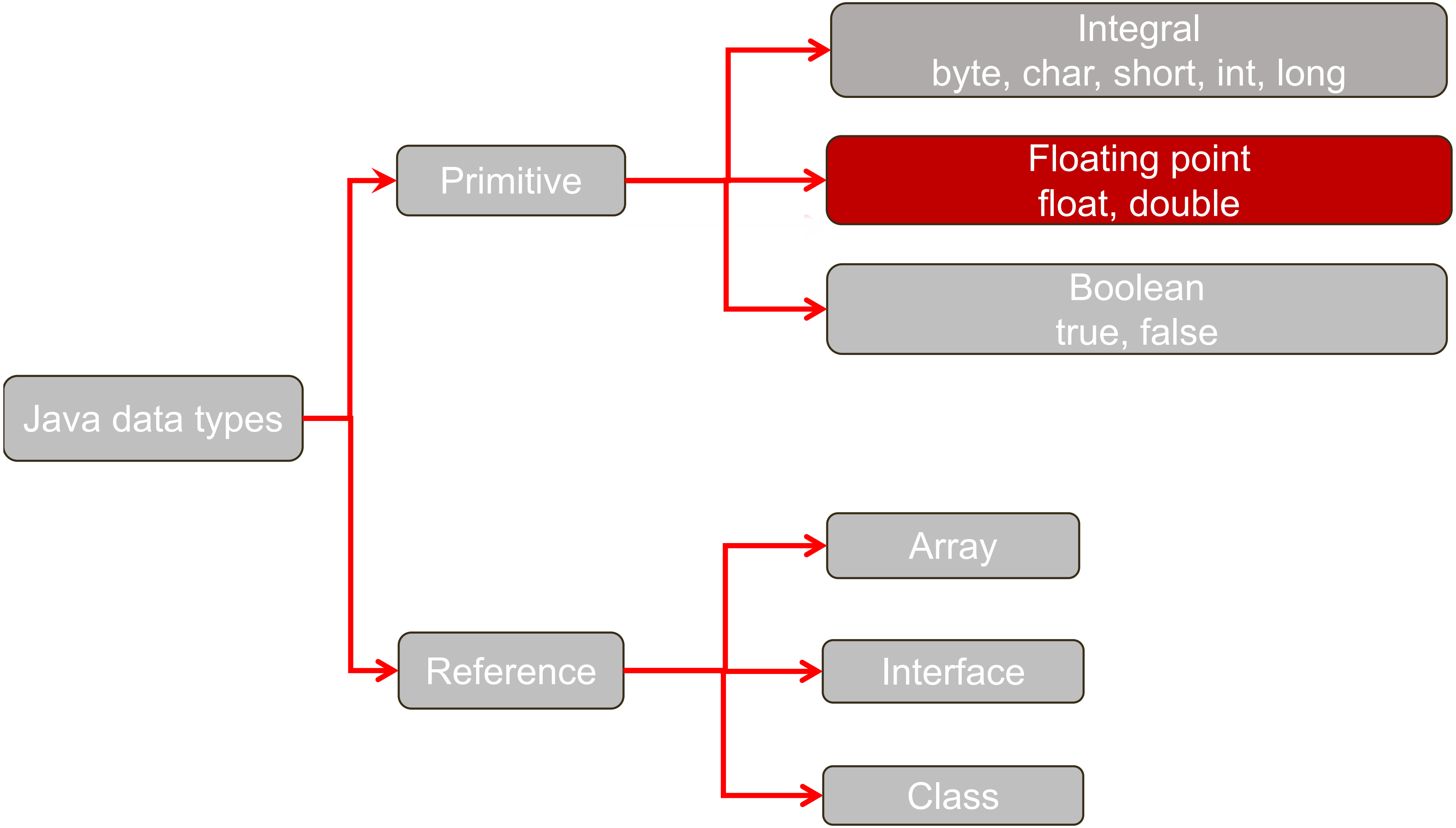
Integer Data Types

Type	Size	Min Value	Max Value
byte	8 bits	-128	+127
short	16 bits	-32768	+32767
int	32 bits	-2147483648	+2147483647
long	64 bits	- 2^64	+2^64

Integers (int)

- Integer data types store whole numbers
    - The number of students
    - The number of pixels
    - The number of books sold

Java data types



Java data types

Primitive

Integral
byte, char, short, int, long

Floating point
float, double

Boolean
true, false

Reference

Array

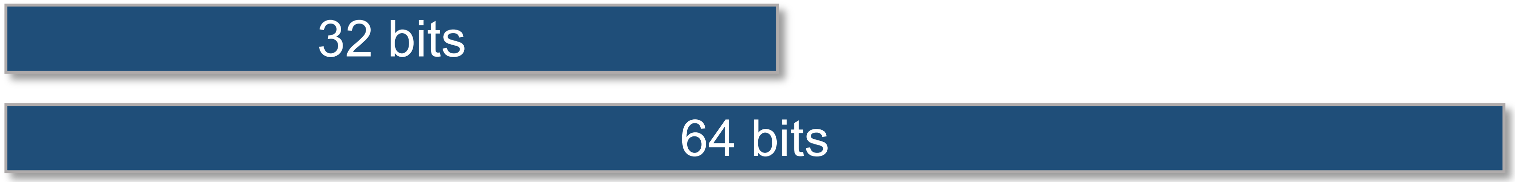
Interface

Class

Floating-Point Types

- To represent real (decimal) numbers
  - 18.0 12.87 4. .8 12.654981

- float
- double



32 bits

64 bits

Floating-point types

- Floating points can also have an exponent value

Java Notation	Scientific Notation
1.7453E-12	1.7453 x 10^{-12}
3.6524E4	3.6524 x 10^{4}
7E20	7 x 10^{20}

Floating point literal

- The default literal that a complier would assume is of type `double`.
- To write literal of type float, end the number with the letter F (or f)

Literal	Type
0.0	double
0.0f	float
2.001E3F	float
1.8E2	double

Floating Point Precision

- float and double data types store an approximation of a ‘real’ number. PC’s have special math FPU
- floats use 32 bits of storage, doubles use 64 bits.
- Try the following small piece of code
  - *double x = 5.02;*
  - *double y = 0.01;*
  - *double z = x + y;*
  - *System.out.println (z);*
- Result will be something like 5.02999999….
- Inherent problem with any real arithmetic – use integers for precision.
- Use ‘*double*’ – only use ‘*float’* to reduce memory requirements – e.g. large ‘array of real numbers

Variable name

- Start with letter (A to Z or a to z)
- Can contain letters or digits
- Can contain _ or $
- Any length name – be sensible!
- Java syntax is case sensitive.
- Conventions
  - camelCase (CamelCase for class names!)
  - snake_case

Declaring variables

```
int length;
int breadth;
int area;
```

```
int length, breadth, area;
```

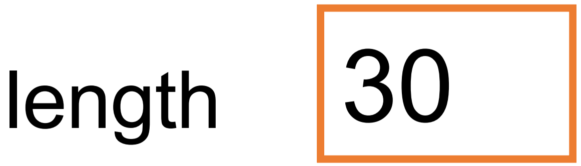
The Assignment Statement

```
length = 20;
```

- `=`
- “is assigned the value”
- The variables named **`length`** is assigned the value 20

Assignment Statement

```
int length;

length = 20;

length = 30;
```



length

30

Program

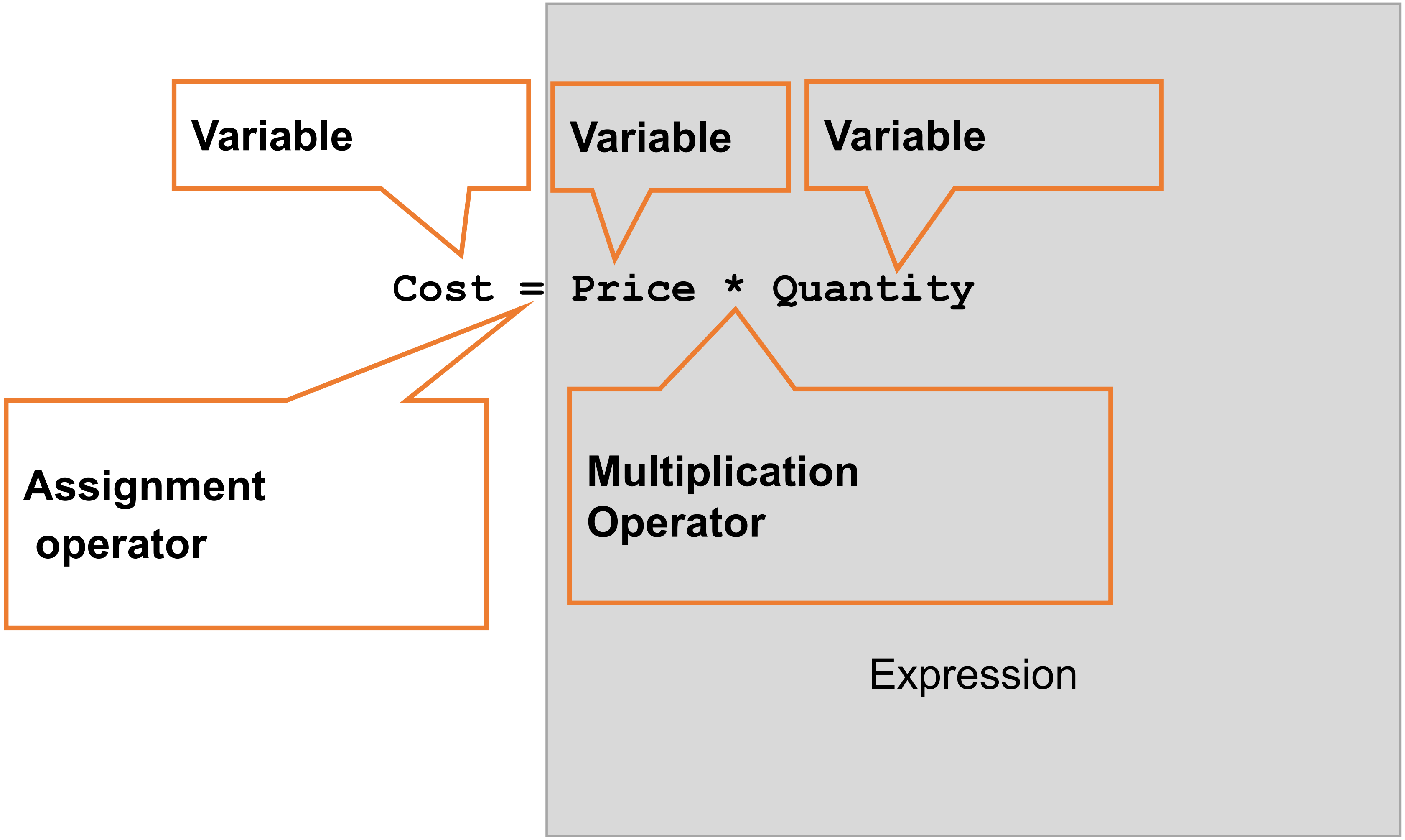
```
public class Calculation {
  public static void main(String[] args) {
     int length;
     int width;
     int area;

     length= 20;
     width= 10;
     area=length*width;
   System.out.println ("Area is "+ area);
   }
}
```

Calculations and operations

- Variable = expression;

```
area = length * width;
```

Operators



Variable
Variable
Variable
Cost = Price * Quantity
Assignment operator
Multiplication Operator
Expression

Arithmetic Operators

Operator	Meaning	Precedence
()	parenthesis	0
*****	multiply	1
/	divide	1
%	remainder	1
+	add	2
-	subtract	2

Example of precedence

```
Total = 10 + 15 * 2 / 4
Total = 10 + 15 * 2 / 4
Total = 10 + 30 / 4
Total = 10 + 7.5
Total = 17.5
```

Example of precedence

$$X= \frac{a + b}{a - b}$$

Algebraic

$X= a + b / a - b$

Java Code

$X= (a + b) / (a - b)$

Java Code

Type Conversion

```
int someInt;    //can only hold int values
double someDouble; // can only hold double values
```

- if we try:

```
someDouble = 12;
```

Java refuses to store anything other than **`double`** value in **`someDouble`**. The complier converts `12` to **`12.0`** and then stores it into **`someDouble`**

Type Conversion

```
int someInt;    //can only hold int values
double someDouble; // can only hold double values
```

- if we try:

```
someInt = 4.8;
someDouble = 12;
```

The Fractional part is truncated(cut-off) so the results are:

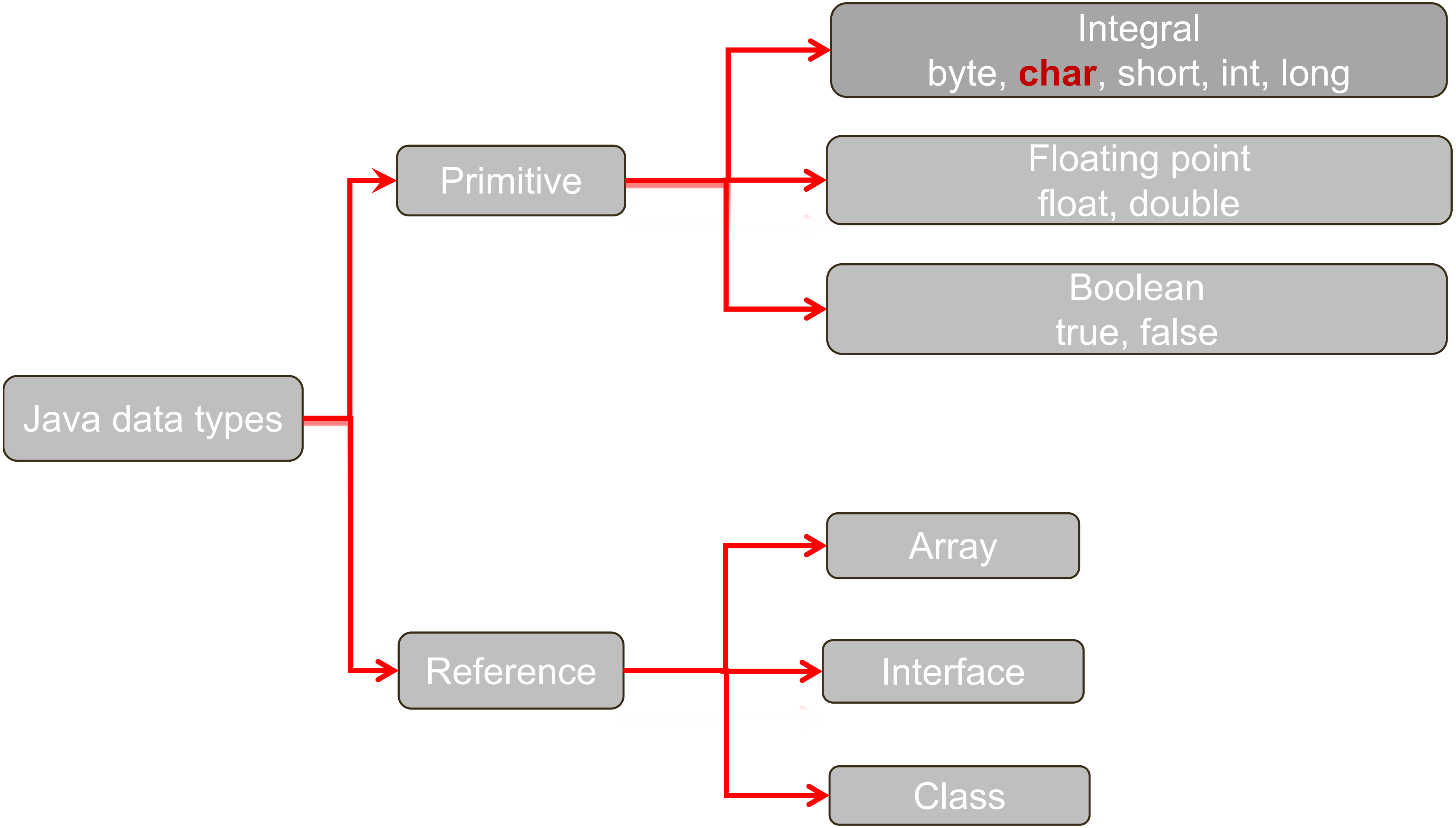
```
someDouble => 12.0;
someInt => 4;
```

Type Casting

- Type casting is the explicit conversion of a value from one data type to another

```
someDouble = (double) (3 * someInt + 2);
someInt = (int)(5.2 / 4.5 - 2.3);
```

Java data types



Character Data Type

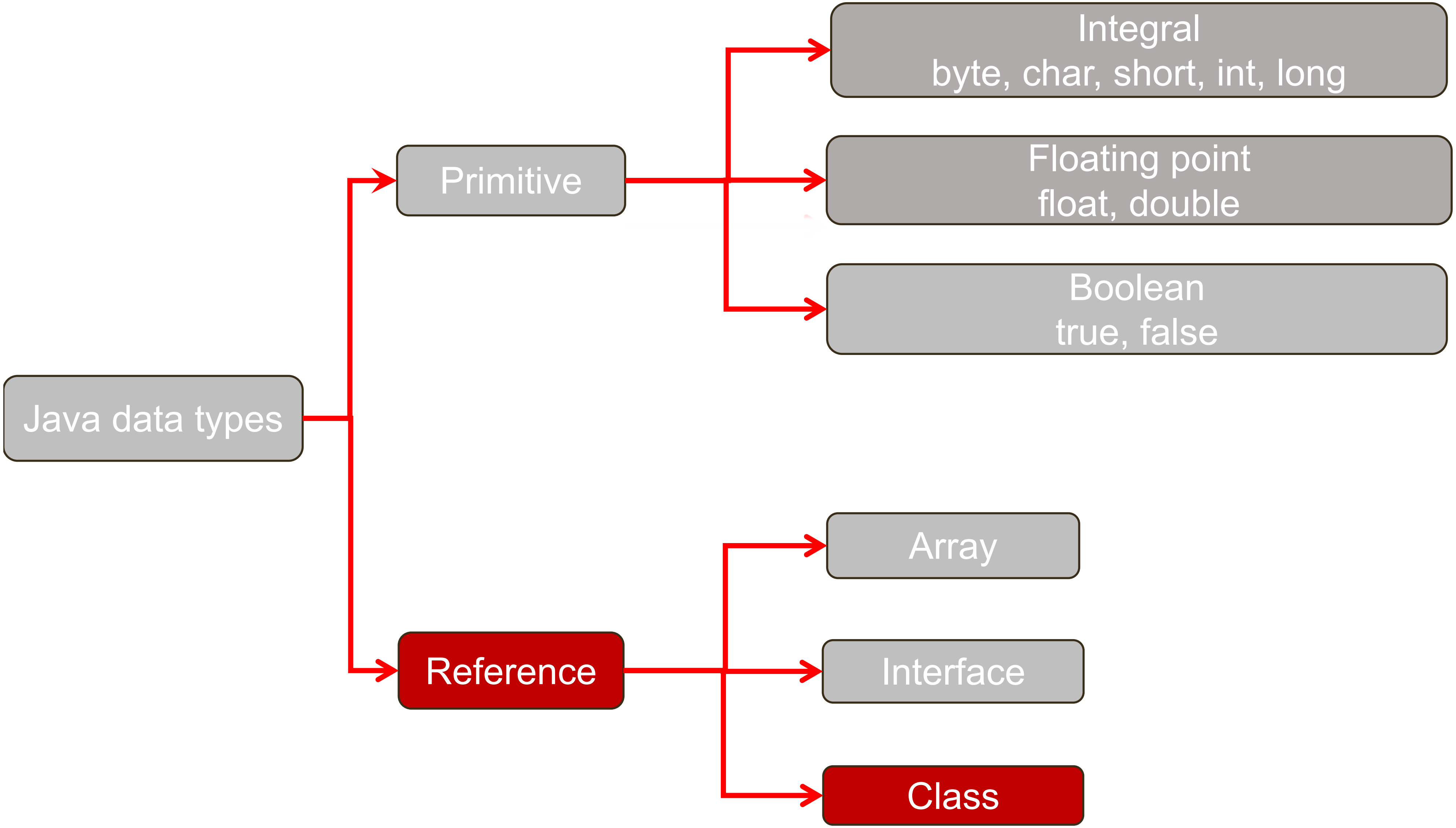
- Java has a character data type: `char`
- `char` variable holds a single character

```
char letter = 'A';
char numChar = '4';
```

  - Special characters
    - `char tab = ‘\t’;`

- Chars require 2 bytes of storage to allow space for other character sets

Java data types



Java data types

Primitive

Integral
byte, char, short, int, long

Floating point
float, double

Boolean
true, false

Reference

Array

Interface

Class

Simple Strings

- Java has a String – a class, not a simple type.
- For now, we can declare a string :

```
String s = “Hello World ”;
```

- Strings can be concatenated with the ‘+’ sign e.g:

```
String t = “from Cranfield”;
System.out.println (s + t);
System.out.println ( s + “from Cranfield”);
```

String type

- **`System.out.println ()`** – library method with several different versions provided, each expecting different parameter type :

```
i = 7;
System.out.println (“Hello number ”);      // String version
System.out.println (i);                  // Integer version
```

- Most classes have an inherent ‘toString()’ method :

```
System.out.println (Integer.toString(i)); // OK
```

Displaying variables

String

Variable

```
System.out.println(“Answer is = “ + cost);
```

String concatenation operator

Variables and Constants

- Declaring and Initializing in One Step
  - `int x = 1;`
  - `double radius = 1.4;`
  - `float f = 1.4f;`
  - `char ch = ‘A’;`
- Constants – note keyword ‘final’
  - *final datatype CONSTANTNAME = VALUE;*
  - `final double PI = 3.14159;`
  - `final int SIZE = 3;`

Integer Maths

- Integer division deals with whole numbers

```
int  i=9;   int j = 4;     int k;     float f;
k = i / j;             // Result of 9/4 = 2 (fractional part is lost)
```

- The % operator lets us catch the remainder of an integer division. Again, result always a whole number

```
k = i % j;   // Remainder of 9 / 4 = 1.
```

- To capture a ‘real’ result, cast i or j before operation

```
f = (float) i / j;// Answer in f = 2.25
f = i / j;          // Cast after division, result = 2.00
```