Building Blocks

Data Types

Primitive Types

Keyword	Type	Min	Max	Default	Example
`boolean`	`true` or `false`	-	-	`false`	`true`
`byte`	8-bit integral value	-128	127	`0`	`118`
`short`	16-bit integral value	-32,768	32,767	`0`	`-202`
`int`	32-bit integral value	-2,147,483,648	2,174,483,647	`0`	`5106`
`long`	64-bit integral value	-2^{63}	$2^{63} - 1$	`0L`	`5106L`
`float`	32-bit floating value	-	-	`0.0f`	`511.183f`
`double`	64-bit floating value	-	-	`0.0`	`511.183`
`char`	16-bit Unicode value	0	65,535	`\u0000`	`'c'`

To keep in mind...

- in Java, boolean `true` and `false` are completely unrelated to `1` and `0` !
- all numeric types are signed (allow negative numbers)
- float requires `f` (or `F`) at the end

```
float x = 2.7;    // DOES NOT COMPILE
float x = 2.7f;  // OK
```

- long requires `l` (or and preferably `L`) at the end

```
long a = 298374612936;   // DOES NOT COMPILE
long a = 298374612936L;  // OK
```

- bit size of boolean is not specified (depends on JVM)

Supported digital formats

- *base 10* (digits 0-9), "normal" numbers
- *octal* (digits 0-7), uses 0 as a prefix (e.g. 017)
- *hexadecimal* (digits 0-9 and letters A-F/a-f), uses 0x or 0X as a prefix
    - format is case insensitive (e.g. 0xFF, 0XFF, 0Xff, 0xff, etc.)
- *binary* (digits 0 and 1), uses 0b or 0B as prefix (e.g. 0b10, 0B11, etc.)

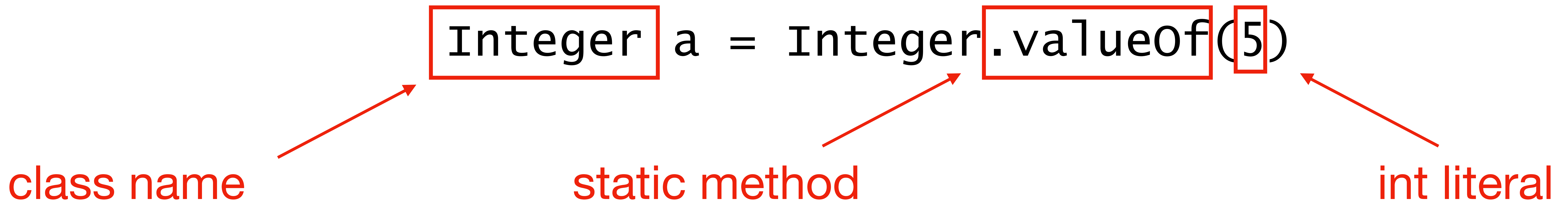
```
// for readability the use of underscore (_) is allowed
// but NOT in the beginning, at the end, right before of after the decimal point

int a = 1_000_000;             // normal usage
int b = 1_2;                   // OK, but not very useful
int c = 1_______2;             // even less useful, but still OK
double d = 1_000_000.000_001;  // OK and makes sense

double x = _10.1;   // NOK
double y = 10.1_;   // NOK
double z = 10_.1;   // NOK
double w = 10._1;   // NOK
```

Wrapper Classes

- primitives are not objects, and sometimes we prefer to work with objects
- each primitive has a *wrapper class*
  - an object type which corresponds to the primitive
- most common way to create an object from the primitive
  - use static method `valueOf()` :



```
Integer a = Integer.valueOf(5)
```

Wrapper Classes

Primitive Type	Wrapper Class	Example
`boolean`	`Boolean`	`Boolean.valueOf(true);`
`byte`	`Byte`	`Byte.valueOf((byte) 12);`
`short`	`Short`	`Short.valueOf((short) 12);`
`int`	`Integer`	`Integer.valueOf(12)`
`long`	`Long`	`Long.valueOf(12L)`
`float`	`Float`	`Float.valueOf(12.0F)`
`double`	`Double`	`Double.valueOf(12.0)`
`char`	`Character`	`Character.valueOf('c')`

```
// valueOf() can be used to convert String into wrapper class
Integer n = Integer.valueOf("12");


// wrapper classes come with some useful methods, e.g.
int m = Integer.parseInt("101");


// before Java 9, this was possible (might appear on OCA exam)
Integer p = new Integer(5);
```

```
// wrapper classes offer many useful helper methods

// byteValue(), shortValue(), intValue(), floatValue(), doubleValue()

// booleanValue(), charValue()


Double d = Double.valueOf(314.67);

System.out.println(d.byteValue());     // 58       (wrap: 314-256=58)

System.out.println(d.intValue());      // 314

System.out.println(d.doubleValue());   // 314.67
```

Strings

- Strings (e.g. "Hello World") are <u>not</u> primitive types in Java
- but they are commonly use like primitives

```
String greeting = "Hello";

String name = "John Wayne";

System.out.println(greeting + ", " + name + "!")   // Hello, John Wayne!
```