COMP90041

Programming and Software Development

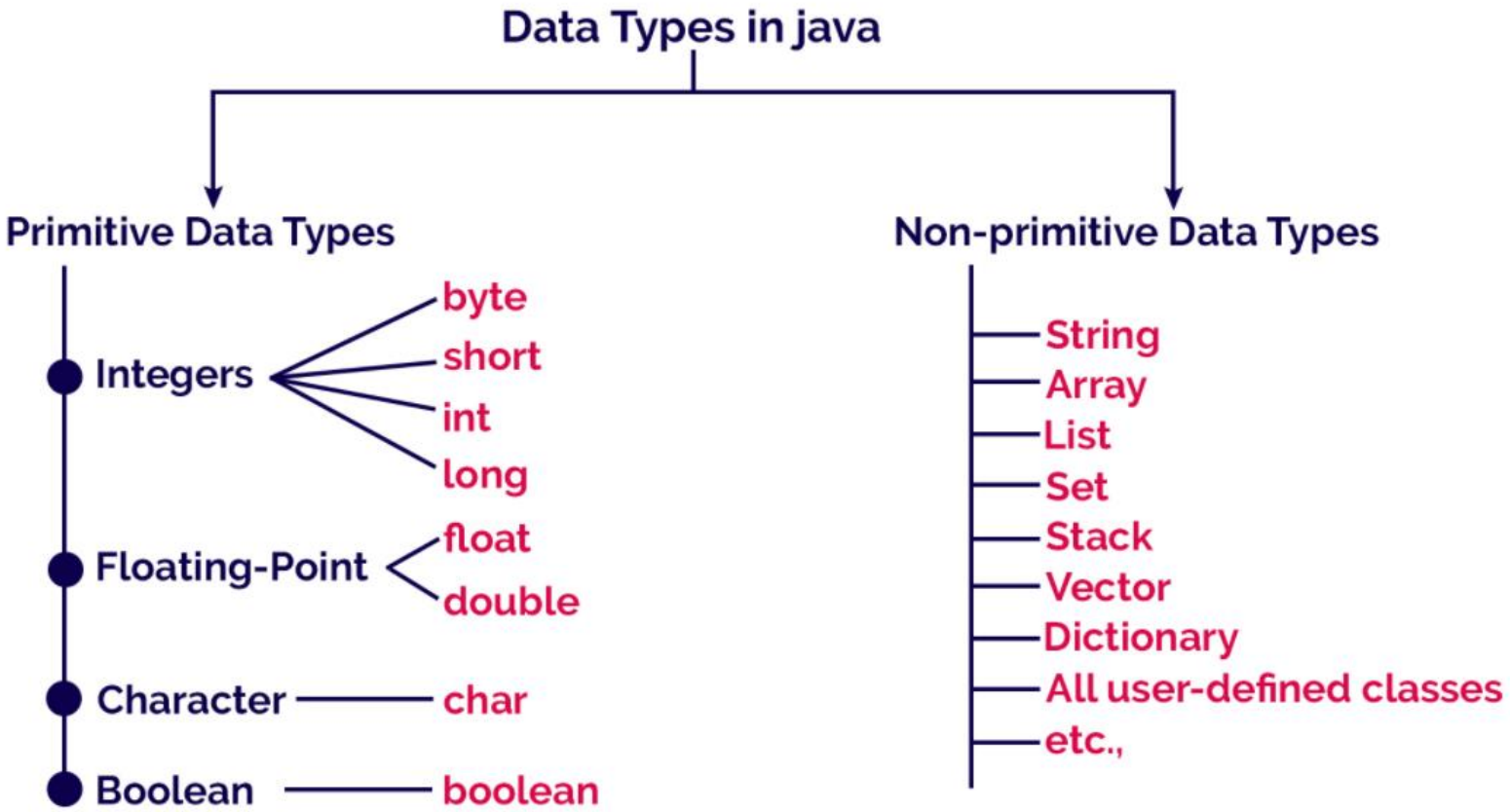
Semester 2, 2021

Lab 2

Andrew Naughton

andrew.naughton@unimelb.edu.au

Primitive types



Primitive types

Data type	Meaning	Memory size	Range	Default Value
byte	Whole numbers	1 byte	-128 to +127	0
short	Whole numbers	2 bytes	-32768 to +32767	0
int	Whole numbers	4 bytes	-2,147,483,648 to +2,147,483,647	0
long	Whole numbers	8 bytes	-9,223,372,036,854,775,808 to +9,223,372,036,854,775,807	0L
float	Fractional numbers	4 bytes	-	0.0f
double	Fractional numbers	8 bytes	-	0.0d
char	Single character	2 bytes	0 to 65535	\u0000
boolean	unsigned char	1 bit	0 or 1	0 (false)

Operations (for numbers)

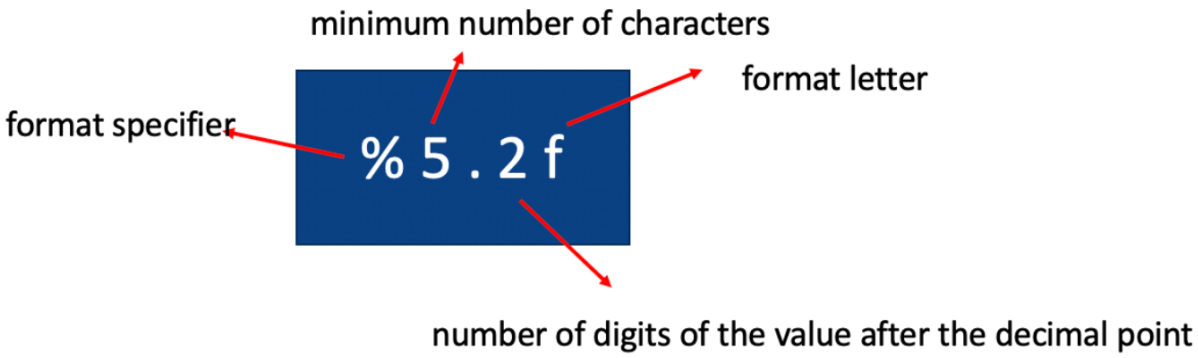
+ - * / % < <= > >= == !=

Operator	Notes	Examples
/	True divide unless both operands are integers, then it is integer divide	True divide: 4.0 / 3 = 1.334 Integer divide: 4 / 3 = 1
%	Remainder or modulo	10 % 3 = 1
==	Checks for equality	(3 * 8) == (4 * 6)

printf – formatted output

- Form:
  ```
  System.out.printf(format-string, args...);
  ```
- *E.g.*:
  ```
  System.out.printf("Average:  %5.2f", average);
  ```

  - If the number if negative, the value will be left-justified, otherwise right-justified



printf – formatted output

Format letter	Date type	Notes
d	int	
s	String	Capital S will uppercase all letters in String
c	char	Capital C will uppercase the letter
n	-	Creates platform specific newline character. Use %n instead of \n for greater compatibility

Print 5.7889 as 5.78

Reading console input

```
import java.util.Scanner;
Scanner keyboard = new Scanner(System.in);
String line = keyboard.nextLine();
```

What	Type	Expression
One word	`String`	`keyboard.next()`
One integer	`int`	`keyboard.nextInt()`
One double	`double`	`keyboard.nextDouble()`

- After `next`, `nextInt`, or `nextDouble`, `nextLine` just reads rest of current line (maybe nothing!)

Read the following input:
9
August
2021

Handling command line inputs

- When your program is run, it can be given arguments on the command line
  - first command line argument: `args[0]`
  - second command line argument: `args[1]`
  - third command line argument: `args[2]`, *etc.*.
- Each of these is a `String`
- To converts string to int: `Integer.parseInt(string)`

if-else

- Form:
  `if (expr)` *Statement*$_1$ `else` *Statement*$_2$
- Executes *Statement*$_1$ if the *expr* is `true`, else executes Statement$_2$

- Java also has an if-else expression:

  expr$_1$ `?` *expr*$_2$ `:` *expr*$_3$

  - If *expr*$_1$ is `true` value is *expr*$_2$
  - If *expr*$_1$ is `false` value is *expr*$_3$

Print PASS or FAIL based on user's mark