

Lecture Chapter 1 Java Fundamentals

Basics

- **Comments**

- `//`
- `/* */`
- `/** */`

appendix A Java language

`javadoc` automatically generate documentation

- **Identifiers**

- **Keywords**

- **Variables**

- **Primitive Types**

- ★ `int`
- ★ `double`
- ★ `char`
- ★ `boolean`

- **Wrapper Classes**

- ★ `Integer`
- ★ `Double`
- ★ `Character`
- ★ `Boolean`
- ★ `Strings`

All classes use reference variables to hold the objects derived from those classes

- ▶ `String`

- ❖ `immutable`

- ▶ `StringBuilder`

- ❖ `mutable`
- ❖ `single tasks`

- ▶ `StringBuffer`

- ❖ `mutable`
- ❖ `synchronized, i.e., multiple tasks concurrently`

- **literal constants**

- `4 193`
- `true false`
- `1.95e3 1.95e-3`
- `'a'`
- `"a" "charles "`
- `\n \\ System.out.println("Hello, \nhe said\n", let's go");`

- **Named Constants**

- `final float DEFAULT_RADIUS = 1.0;`
- `Math.PI Math.e Math.E`

- **Assignments**
 - `A = B = C = D = 51;` right associative

Operators are generally left associative

- **Arithmetic Operators**
 - `+` binary, unary
 - `-` binary, unary

} unary has higher precedence than binary

} `+` & `-` have equal precedence

 - `*`
 - `/`
 - `%`

} Equal Precedence

- **Relational Operators** comparison

- `<`
- `<=`
- `>`
- `>=`

Relational & Equality Operators combine two variables together to form a relational expression which can yield a boolean value, i.e., `A < B` yields either **TRUE** or **FALSE** depending upon the values held by `A` & `B` and the specific operator used

- **Equality Operators**

- `==`
- `!=`

- **Logical Operators**

- `&&` AND
- `||` Inclusive OR
- `!` NOT
- `^` Exclusive OR

Logical Operators can be used to combine two relational expressions together to form larger logical expressions which, in turn yields a boolean value for the larger logical expression.

- **Implicit Conversions**

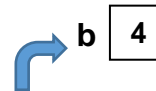
- `int` → `long` → `float` → `double`

- **Explicit Conversions**

- `cast` `double volume = 9.7;`
- `System.out.println((int) volume);`

- **Multiple Assignments**

- `a = 5 + (b = 4);` → `a = (5 + (b = 4))` → `a = (5 + 4)` → `a = 9`



- **Combined Operators**

- `+=` `a += 5;` → `a = a + 5;`
- `-=` `a -= 5;` →
- `*=`
- `/=`
- `%=`
- `++`
- `--`