Java Operators
Unary prefix operators

- Some operators can be used with one variable only
  - `++expr` Preincrement
  - `--expr` Predecrement
  - `+-` Unary plus and unary minus
  - `!` Logical negation (not)
  - `~` Bitwise complement (invert every bit)
Ternary operator

- \( \text{varName} = \text{boolean-expr} \, ? \, \text{expression-1} : \text{expression-2} \)
- This is like if-then-else
- If the \( \text{boolean-expr} \) evaluates to true, \( \text{varName} \) is set to \( \text{expression-1} \), else it is \( \text{expression-2} \)
- Example: \( \text{max} = a > b \, ? \, a : b \); sets the variable \( \text{max} \) to the larger of \( a \) and \( b \)
- \( \text{expression-1} \) and \( \text{expression-2} \) need not be the same type, but either result must be useable
- *The ternary operator is right associative!*
- Personally, I think the ternary operator is confusing – use it at your own risk
Assignment operators

- You can assign the same value to multiple variables simultaneously
  
  \[ a = b = c = 7.5 \times w; \]
  
  - assigns \( 7.5 \times w \) to \( c \), then assigns \( c \) to \( b \), then assigns \( b \) to \( a \) – if all these assignments are legal

- Assignment is right associative

- You can use all standard arithmetic operators in the shorthand assignment \texttt{operator=}
  
  - Example: \texttt{variable += expression; variable /= expression;}
Object creation and casting

• Use **new** to create a new instance of a class
• Use **(type)** to cast/convert a variable to the given **type**
  – Casting tells the compiler the variable actually has the given **type**
  – For primitive types, you can only cast between numeric types (**int**, **float**, **short**, **double**, **long**, **char**) – you may lose precision
  – For objects, you can only cast a superclass into a subclass
  – A cast essentially tells the compiler how to interpret the (binary) memory when the object is stored
  – If your cast is unsuccessful, you will get a **ClassCastException**
Relational operators

• Have already seen the standard comparison operators for numbers, <, <=, >=, >

• For objects, we can check whether an object is a given object type using `instanceof`
  
  ```java
  if (Animal instanceof Dog) { ... }
  ```
  
  – Useful if we’d like to do a cast
  – Sometimes, we want to check the specific type of a given object and do something different depending on the type
What you need to know

• You should understand what each operator does
• Parameter lists, array indexing, casting, `++` and `--`, and the dot operator are done first
  – In particular, a cast refers to the one following entity, so to cast the result of an expression you need extra parentheses
  – Example 1: `variable = (type)(expression);`
  – Example 2: `variable = ((type)variable).method();`
• In arithmetic, the unary operators `+` and `–` are done first, then multiplication and division, then addition and subtraction
• All assignment operators are done last
• For anything else, it’s a good idea to use parentheses anyway