Java Program Structure

- In the Java programming language:
  - A program is made up of one or more classes
  - A class contains one or more methods
  - A method contains program statements
- A Java application always contains a method called `main`

Java comments can take two forms:

```
// comment runs to the end of the line
/* comment runs to the terminating symbol, even across line breaks */
/* Here is a nicer looking (?)
 * comment format that many programmers use */
```

Variable name

- Can be made up of letters, digits, underscore character (_)
- Cannot begin with a digit
- Java is case sensitive, so `Total` and `total` are different identifiers
- Use meaningful variable names!!
Variable and declaration

- Variable must be declared: specify variable's name and type
- Multiple variables can be declared in one declaration statement

```
int total;
```

Type

- A type is a set of values along with a set of operations on those values
- E.g., the set of integers
  \( \{ \ldots, -2, -1, 0, 1, 2, \ldots \} \)
  along with the arithmetic operations
  \(+, -, \times, \div, \%\)

Primitive Data

Eight primitive data types in Java:

- Four types of integers:
  - byte, short, int, long
- Two types of floating point numbers:
  - float, double
- One character type:
  - char
- One logical type called boolean:
  - boolean

Arithmetic Expressions

- *Arithmetic expressions* compute numeric results and use arithmetic operators:
  - Addition: \(+\)
  - Subtraction: \(-\)
  - Multiplication: \(\times\)
  - Division: \(/\)
  - Remainder: \(\%\)

Integer division and remainder

- If both operands to the division operator (\(/\)) are integers, the result is an integer (the fractional part is discarded)

  \[
  \begin{align*}
  14 & / 3 = 4 \\
  8 & / 12 = 0
  \end{align*}
  \]

- The remainder operator (\(\%\)) returns the remainder after dividing the second operand into the first

  \[
  \begin{align*}
  14 & \% 3 = 2 \\
  8 & \% 12 = 8
  \end{align*}
  \]

Arithmetic Expressions

- *Arithmetic expressions* compute numeric results and use arithmetic operators:
  - Addition: \(+\)
  - Subtraction: \(-\)
  - Multiplication: \(\times\)
  - Division: \(/\)
  - Remainder: \(\%\)

- If either or both operands to an arithmetic operator are *double*, the result is a *double*
### Type conversion
- **Arithmetic promotion:** automatic
- **Casting:** an explicit conversion

\[
\begin{align*}
5 & \div (\text{double}) 9 \\
5 & \div 9.0
\end{align*}
\]

### Variable and declaration
- Variable must be **declared:** specify variable's name and type
- Multiple variables can be declared in one declaration statement

```c
int total;
int count, temp, result;
```

### Assignment
- Expression on the right is evaluated and the result is stored in the variable on the left
- Expression type must be **same as or narrower than** the variable type

```c
int total;  //declaration
total = 55;  //assignment
```

```
total  55
```
Declaration and assignment

- Variable must be declared: specify variable’s name and type
- Multiple variables can be declared in one declaration statement
- Can declare a variable and assign a value to it in one statement

```java
int total;
int count, tmp, result;
int sum = 0;
int base = 32, max = 149;
```

Constant

- Variable has one value in the entire program
- Compiler will issue an error if you try to change a constant
- Use the modifier (keyword) `final` to declare a constant

```java
final int MIN_HEIGHT = 69;
```

- Why specify constants:
  - give names to otherwise unclear literal values
  - facilitate changes to the code

The Math class

Collection of basic math functions
Examples:

```java
double tmp = Math.exp(1);
tmp = 3 * Math.sin(2);
tmp = Math.random(); // in [0,1]
tmp = Math.floor(Math.random());
```

Boolean

- Represent conditions or states `true` or `false`
- Only two valid values for boolean type: `true, false`

```java
boolean done = false;
```

Boolean Expressions

- Expressions with relational operators give boolean results:
  - `==` equal to
  - `!=` not equal to
  - `<` less than
  - `>` greater than
  - `<=` less than or equal to
  - `>=` greater than or equal to
- Logical operators:
  - `&&` (and) `||` (or) `!` (not)