CS18000: Problem Solving and Object-Oriented Programming

Primitive Types and Strings

Video 1
Data Types

Numbers and Mathematical Operators

Types

Primitive Types

Values, Variables, and Literals

- Programs (and CPUs) work with *values*
- Values are represented in programs by *literals* and stored in *variables*
- Literals
  - 3, -23, 4.5, 0.23, 3E8, 6.02e+23
  - "Hello there", 'A', true, false
- Variables
  - x, y, a, b, helloMessage, wheel, robot, r1, w27
  - Use letters, digits, and "_" (start with letter)
  - Identify a location in memory

Types

- Variables and literals have *types*
- Examples
  - int
  - double
  - String
- Type is a formal definition
  - Set of values
  - Set of operations on those values

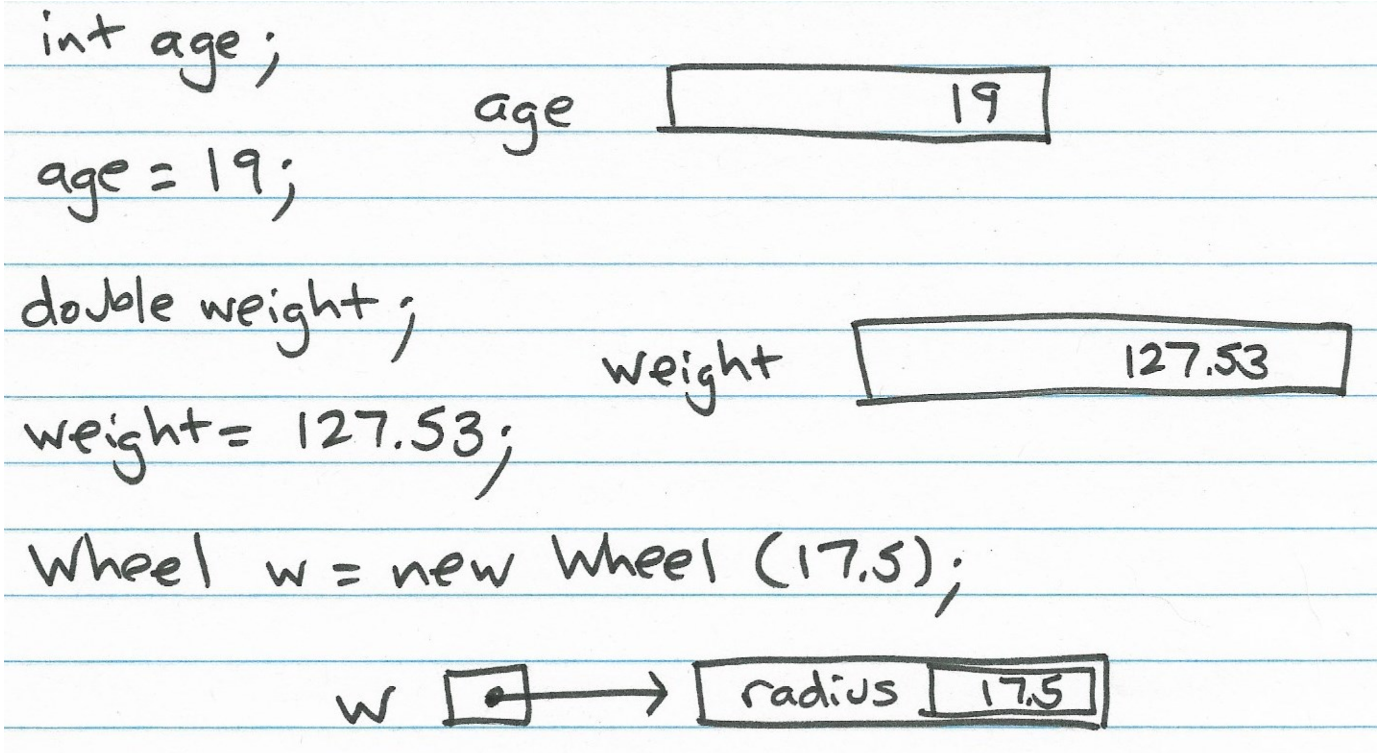
Example Java Type: int

- Set of values: subset of the integers
  - Stored in 4 consecutive bytes: 32 bits
  - Range: -2,147,483,648 to 2,147,483,647
  - Literals: 23, 45, -19, 0
  - Variables declared with “int” reserved word
- Set of operations: standard mathematical
  - +, -, *, /
  - % (mod) remainder 17%3=2

Type Categories in Java

- Primitive Types
  - Built-in to language
  - boolean, byte, short, int, long, float, double, char
  - Occupy enough bits/bytes to store value
- Reference Types
  - Can be defined by the user
  - Hold a “reference” (pointer) to an object

Primitive and Reference Types

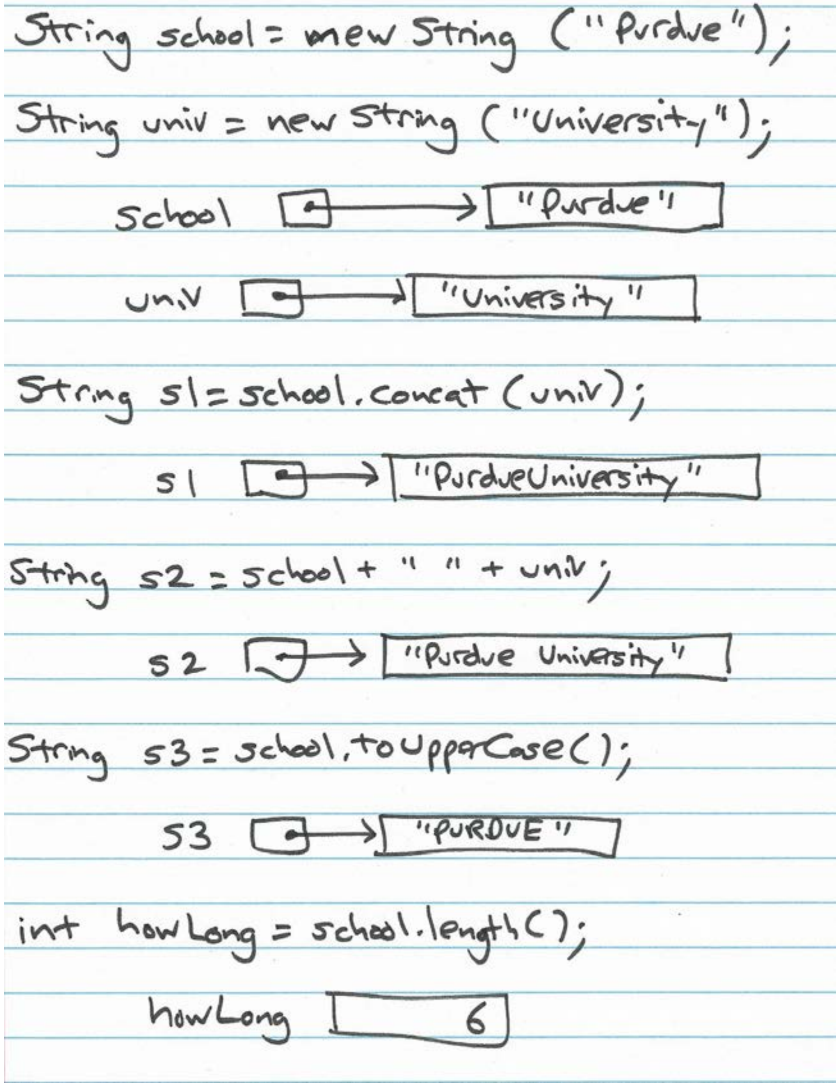


```
int age;

age = 19;

double weight;

weight= 127.53;

Wheel w = new Wheel (17.5);
```

Example Reference Type: String

- Set of values:
  - Sequences of characters
  - Length 0 to 2,147,483,647
- Set of operations:
  - concat() (also + operator)
  - toUpperCase()
  - length()
  - substring()
  - Plus many others

Example Reference Type: String



```
String school = new String ("Purdue");
String univ = new String ("University");
```

```
String s1 = school.concat (univ);
```

```
String s2 = school + " " + univ;
```

```
String s3 = school.toUpperCase();
```

```
int howLong = school.length();
```

Wheel Class

```
public class Wheel {
    double radius;

    Wheel(double radius) {
        this.radius = radius;
    }

    double getCircumference() {
        return 2 * Math.PI * radius;
    }

    double getArea() {
        return Math.PI * radius * radius;
    }

    double getRadius() {
        return radius;
    }
}
```

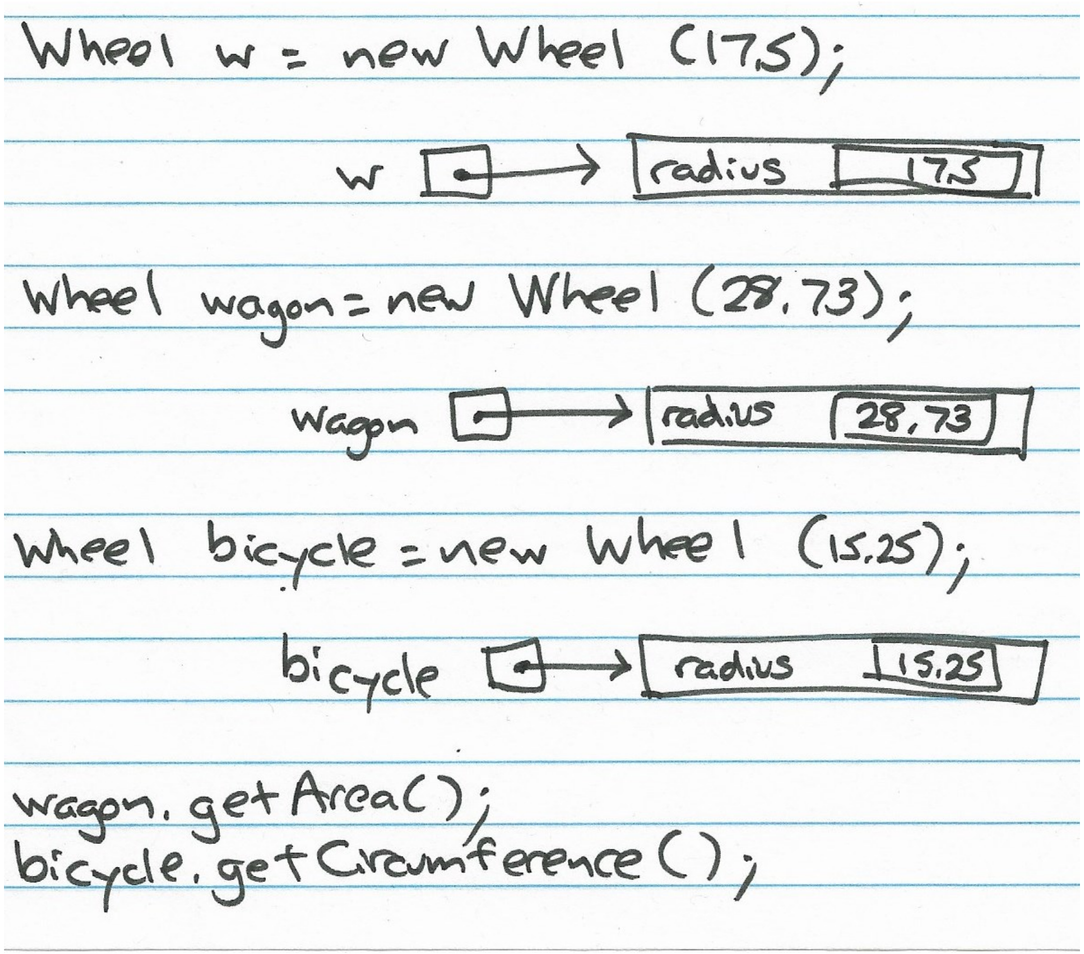
Example Reference Type: Wheel

- Set of values:
  - Limited by memory only
  - A new one created for each “new Wheel(…)”
  - Same or different radius
- Set of operations
  - getArea()
  - getCircumference()
  - getRadius()

Variables and Literals

- Variable
  - Memory location where something can be stored
  - Contents of the location can change: vary
- Literal
  - A value that cannot change
  - Can be stored in a variable
  - Examples: 42, 3.14159, "hello", and 'X'

Variables and Literals



```
Wheel w = new Wheel (17.5);

Wheel wagon = new Wheel (28.73);

Wheel bicycle = new Wheel (15.25);

wagon.getArea();
bicycle.getCircumference();
```

Declarations

- In Java, variables must be declared and given a type
- Java compiler does two things with this information
  - Arranges for space to be allocated for the variable to store a value
  - Ensures that only valid (type-defined) operations can be performed on this variable

Video 2
Primitive Types

Primitive Types

Integer and Real Number Types

Integer Types in Java

- All represent subsets of the integers
- Differ in how many bits used to store the value (and, so, how many values)
  - byte: 8 bits (-128 to 127)
  - short: 16 bits (-32,768 to 32,767)
  - int: 32 bits (-2,147,483,648 to 2,147,483,647)
  - long: 64 bits (18 digits)
- Most popular: int

Operations on Integer Types

- Usual mathematical: +, -, *, /, and % (mod)
- Important note:
  - Divide operation is “integer divide”
  - Result is an integer, even if it “should” be a fraction
  - Called “truncation”
  - Examples:

    10 / 5 is 2

    13 / 4 is 3

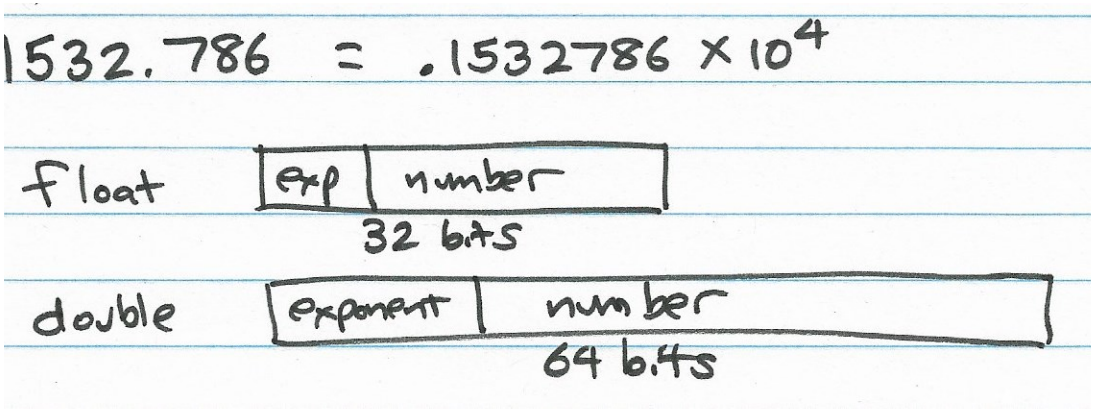
    3 / 2 is 1

    1 / 2 is 0

Real Number Types in Java

- Represent subsets of the real numbers
- Two types, differing in number of bits used
  - float: 32 bits (aka “single precision”)
  - double: 64 bits (aka “double precision”)
- Most popular: double

Real Number Types in Java



$1532.786 = .1532786 \times 10^4$

Operations on Real Types

- Usual mathematical: +, -, *, /
- The Math class includes many others (some also applicable to integers):
  - Math.pow(base, exponent)
  - Math.log10(number)
  - Trig functions, logs, etc.
- See http://docs.oracle.com/javase/6/docs/api/java/lang/Math.html

Declaring Variables: Syntax

- Various possibilities supported…

```
int x;          // declare only
int x = 5;      // and initialize
int x, y;       // two at once
int x = 5, y = 10;
```

- Best practice guidelines…
  - Declare only one variable per line
  - Include a comment describing its purpose

```
int mass;          // mass of the particle
```

Expressions

- Expressions built by combining
  - variables (x, y) and literals (3, 27) with
  - operators (+, -)
- Usual mathematical precedence
  - Multiplication and division first
  - Addition and subtraction second
  - (see chart on next slide)

x = b + c * d - a / b / d; … same as…

x = ((b + (c * d)) - ((a / b) / d));

Precedence (Highest to Lowest)

Category	Operator
Unary	+expr -expr
Multiplicative	* / %
Additive	+ -
Shift	<< >> >>>
Bitwise operators	& then ^ then |
Logical operators	&& then ||
Assignment	=

Complete list: http://docs.oracle.com/javase/tutorial/java/nutsandbolts/operators.html

Video 1
Type Promotion and Characters

You are much smarter than a computer

A computer cannot do this….

3 + 5.0

Type Promotion

- Mixing value types in expressions is allowable
- Values are “promoted” when it makes sense (no data is lost)
  - short to int, float to double
  - integers to reals (e.g., int to double)
- Examples:

  3 + 5.0 evaluated as 3.0 + 5.0 -> 8.0

  1 / 2.0 evaluated as 1.0 / 2.0 -> 0.5
- Promotion is a form of casting…

Casting

- Cast: convert from a value of one type to a value of another type
- Upcast: from “narrower” to “wider” (more bits)
  - short -> int
  - float -> double
  - int -> double
- Downcast: from “wider” to “narrower” (fewer bits)
  - int -> short
  - double -> float
  - double -> int

Casting Rules

- Upcasting is fine
  - Nothing lost
  - Java handles automatically (“promotes”)

```
double x = 1 / 2.0;  // x is assigned 0.5
```

- Downcasting is dangerous
  - Precision is lost
  - Java prevents by default
  - Programmer must override with cast operator

Cast Operator

- Tells Java compiler…
  - “Trust me; the expression can be converted to the indicated type”
- Put type name in parentheses before expression
- Example:

```
int x;
double y = 12.0;
x = y; // “loss of precision” error
x = (int) y; // allowable
x = (int) (y / 3.0); // also allowable
```

Constructors and Fields

- Constructor
  - A special method in a class that is used to “construct” an object when it is being created
  - Called by the “new” operator
  - `new Wheel(15)` invokes the `Wheel(double radius)` method
- Fields (instance or member variables)
  - Variables located inside the class definition that become part of the object
  - Often “initialized” by the constructor using “this”
  - `this.radius = radius`

Primitive Type: char

- Set of possible characters, symbols that make up a String
- Encoded as number: e.g., ‘A’ is 65, ‘B’ is 66, ‘C’ is 67, etc.
- Java uses 16 bits to store each character (65536)
- Historical growth…
  - Initially ASCII standard of 128 characters
  - Extended to Latin-1 character set of 256 characters
  - Now at Unicode character set of 65536 characters

Set of char Values

- ASCII subset
  - Character codes 0-127 (7 bits)
  - Include English alphabet (upper and lower), numbers, punctuation, special characters
- Latin-1 extension
  - Added character codes 128-255 (8 bits)
  - Include non-English (Romanized) characters and punctuation (dipthongs, accents, circumflexes, etc.)
- Unicode 1 extension
  - Added character codes 256-65535 (16 bits)
  - Includes non-Romanized alphabetic characters from Cyrillic, Hebrew, Arabic, Chinese, Japanese, etc.

char Literals

- A single character surrounded by single quotes, for example
  - ‘A’, ‘a’, ‘x’, ‘0’, ‘3’
  - ‘!’, ‘,’, ‘”’, ‘&’
- A special “escape sequence” surrounded by single quotes, for example
  - ‘\t’ (tab)
  - ‘\n’ (newline)
  - ‘\’’ (single quote)
  - ‘\\’ (backslash)
  - ‘\uxxxx’ (char hexadecimal xxxx in Unicode set)

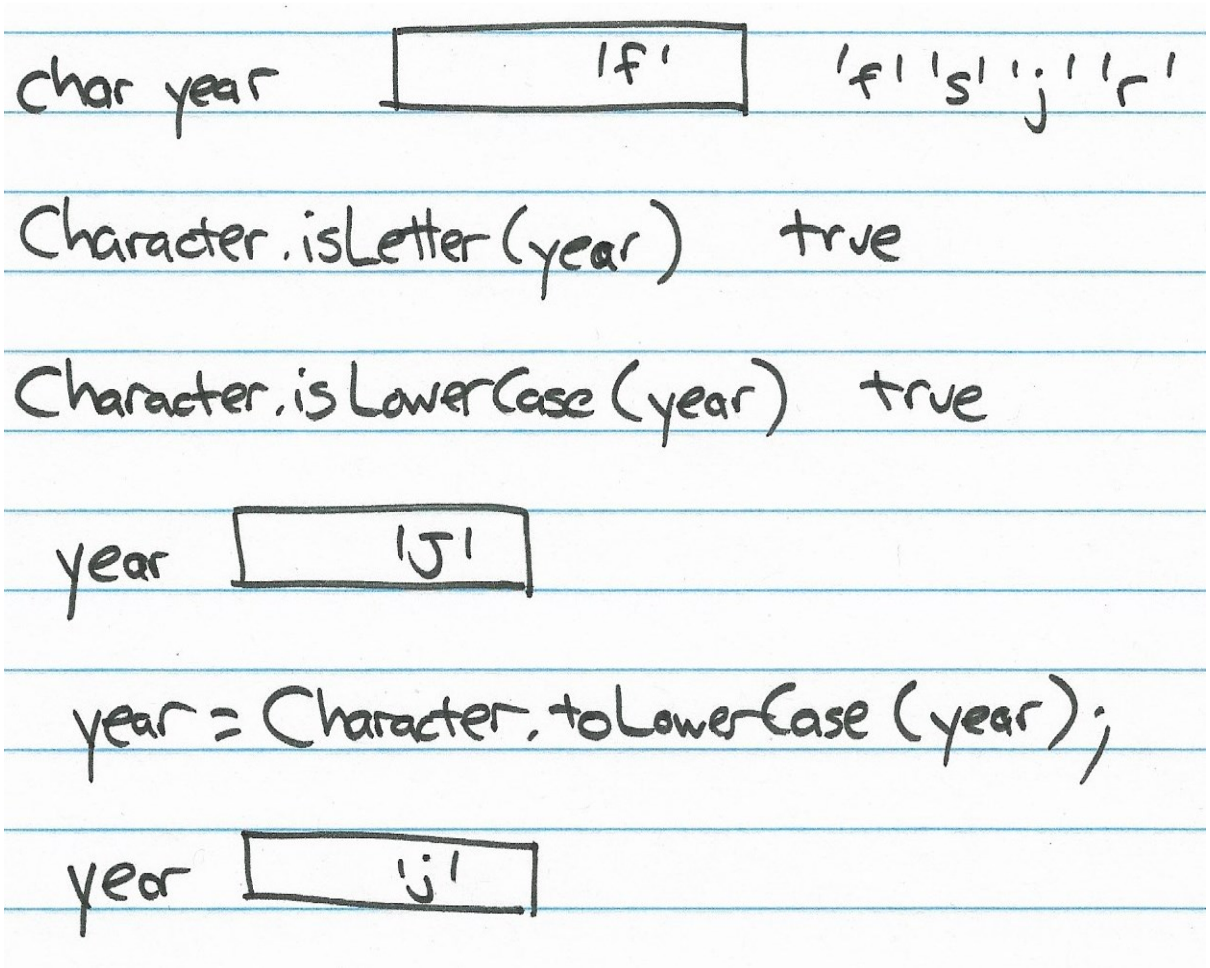
Set of char Operations

- Treated as (upcast to) String for printing and concatenation
  - System.out.println(‘A’) -> prints A
  - “Hello” + ‘!’ -> “Hello!”
- Treated as integer for arithmetic operations
  - ‘a’ + 0 -> 97
  - ‘z’ – ‘a’ -> 25
- Many more operations implemented by methods in class Character

Useful Character Methods

- Character.isDigit(char value)
- Character.isLetter(char value)
- Character.isLetterOrDigit(char value)
- Character.isLowerCase(char value)
- Character.isUpperCase(char value)
- Character.isWhiteSpace(char value)
- Character.toLowerCase(char value)
- Character.toUpperCase(char value)

Useful Character Methods



char year 'f' 'f' 's' 'j' 'r'

Character.isLetter(year) true

Character.isLowerCase(year) true

year 'J'

year = Character.toLowerCase(year);

year 'j'

Primitive Type: boolean

- Set of two elements { true, false }
- Set of operations
  - Logical: && (and), || (or), ^ (xor), and ! (not)
  - Testing in various Java statements (e.g., if)
- Created by comparison operators
  - x < y, x <= y, x == y, x != y, x > y, x >= y
  - And result of logical operators (above)
  - Note: == and != also work with reference types, but only compare references (addresses) not the values

Video 2
Reference Types and Strings

Reference Types

- Unlike primitive types:
  - Are extensible: can be created by the programmer
  - Variable declaration creates space for reference to object, not the object itself
- Defined with a class declaration
- Set of values: created with the ***new*** operator
- Set of operations: defined and implemented by the methods in the class

Declarations and Reference Types

The declaration of a variable of a reference type allocates space only for the reference, not for the object to which it refers

If the variable is a field of an object, and you say...

```
Wheel w;
```

... `w` will contain the "null pointer" (pointer to nothing). Then, if later you say...

```
w.getRadius();
```

... this will result in a “null pointer exception”

If the variable is a local variable in a method, and you say...

```
Wheel w;
```

... the Java compiler will refuse to compile it and make you have an actual reference...

```
Wheel w = new Wheel (17.5);
```

```
Wheel w = null;
```

... will also compile, but can easily lead to a “null pointer exception”

Important Reference Type: String

- Java class String is built in
  - No need to import
  - Language supports String literals (“hello”)
- Because String is a class…
  - Instances (e.g., literals) are objects
  - Can create with new operator

```
String greeting = new String (“Hello”);
```

  - String variables hold references to objects

Local Variable Type Inference

```
Wheel w = new Wheel (15.75);
String school = new String ("Purdue");
```

repeat the class name `Wheel` and `String`

Java allows a shortened form using `var`

```
var w = new Wheel (15.75);
var school = new String ("Purdue");
```

These are equivalent to the declarations above

Operations on Strings

- Concatenation (+) built-in to Java
- Lots of operations defined as methods in the String class
- Strings are immutable: no operation on a String object changes the value of that object

Comparing Strings

- == does not work in the way you might expect
- Strings are objects
- s1 == s2
- == between objects only compares the references (addresses) of the objects
- Two different String objects with the exact same characters will compare == false (since their objects are stored in different locations)
- Use String equals method: s1.equals(s2)

Formatting Strings

- Template (“format”) string includes regular characters and “escape” sequences:
  - %s: string
  - %d: integer (byte, short, int, long)
  - %f: float, including double

```
System.out.printf(“%s! %d or %f”, “Hi”, 42, 3.14159);
prints Hi! 42 or 3.14159
```

- Width specification allowed
  - %10s: pad string on left to make it 10 characters
  - %-10s: pad string on right to make it 10 characters
  - %12d: pad integer on left to make it 12 characters
  - %10.3f: format number with 3 decimals; total width 10
  - %.2f: format with 2 decimals; whatever width needed
- Full details in Javadocs Formatter

The final Keyword

- `final` is a modifier used in variable declarations
- Prevents the variable from being changed
- Example 1

```
final int SIZE = 100;
SIZE = 50;  // not allowed by Java
```

- Example 2

```
Math.PI = 3.1; // also not allowed
```

Wrapper Classes and Useful Methods

- Byte
- Short
- Integer
- Long
- Float
- Double
- Character
- Boolean

String to Numeric Value

- Convert (parse) a String to a numeric value
- Available for all numeric wrapper classes, but two most useful ones are…
  - Integer.parseInt("4000")
  - Double.parseDouble(" 66.23457")
- Watch for NumberFormatException