Variable types: definitions

- **Instance variable**: a variable that's associated with a particular object
- **Class variable**: a variable that's shared across all objects of a class
  - static keyword
- **Constant**: a variable that has a fixed value (cannot be changed)
  - final keyword

Example

class Cat{
    int numLegs;  // instance variable
    String name;  // instance variable
    String breed;  // instance variable
    static final int MAX_LEGS = 4;  // class variable, and a constant

    public Cat() {
        // constructor; this one's empty
    }
}

Example (cont.)

Cat c1 = new Cat();  // instantiates an object of class Cat
Cat c2 = new Cat();  // instantiates another object of class Cat

// class variable, and a constant

// Note: access a class variable using the class name, not the object name!
System.out.println("Maximum number of legs for "+c1.name+": "+c1.numLegs);
System.out.println("Maximum number of legs for "+c2.name+": "+c2.numLegs);
More about methods

- Recall: methods have three parts
  - return type (if it returns nothing, the type is \texttt{void})
  - name
  - list of input parameters (can be empty)

What happens when we call a method?

Example: Calling the \texttt{Rectangle} constructor in \texttt{FunWithShapes.java}

- Check: Is “\texttt{Rectangle}” a built-in data type in Java (like \texttt{int} and \texttt{String})? (no)
- Check: Is there a \texttt{Rectangle.class} file in the same directory as \texttt{FunWithShapes.java}? (yes, so we'll use this to find the definitions of the variables and methods of \texttt{Rectangle}, as needed)

Example (cont.)

- \texttt{Rectangle r1 = \ldots} is a variable, of type \texttt{Rectangle}, and we'll be assigning it a value right now
- \ldots \texttt{= new Rectangle(...)}: \texttt{new} means call the constructor. Go take a look at the constructor in \texttt{Rectangle.class}
- \texttt{public Rectangle(int w, int h )}: means that the constructor is expecting 2 input parameters, both of type integer. Go back to \texttt{FunWithShapes} and see if that's what we're giving (passing) to the constructor

Example (cont.)

- \ldots \texttt{= new Rectangle(27, 13)}: 27 and 13 are both integers, so the constructor has what it needs to initialize this object
- Back to the \texttt{Rectangle constructor}:
  - substitute 27 for \texttt{w}
  - substitute 13 for \texttt{h}
  - set \texttt{width = w}, and \texttt{w = 27}, so \texttt{width = 27}
  - set \texttt{length = h}, and \texttt{h = 13}, so \texttt{length = 13}
- We're done!
Key point #1

- It doesn't matter what we name the variables in the constructor, nor does it matter if we pass numbers or variables (of whatever name) to the constructor
  - we only care that the parameters match in terms of how many and type
  - example: replace \textit{w} and \textit{h} with \textit{bob} and \textit{nancy}....same results!
  - example: in FunWithShapes, pass in \textit{a} and \textit{b} instead of 13 and 27....same results!
  - did not work if we replace \textit{w} and \textit{h} with \textit{width} and \textit{length}, however

Example

- \texttt{FunWithShapes.java} contains the line
  \begin{verbatim}
  int area1 = r1.calculateArea();
  \end{verbatim}
  - method \texttt{calculateArea} is called in class \texttt{Rectangle}
  - no input parameters needed
  - area is calculated in the method
  - method returns an integer value (area)
  - area's value is sent back to \texttt{FunWithShapes}, and is stored in the variable \texttt{area1}

Key point #2

The same thing goes for the return values for functions: it doesn't matter what we call the return value, as long as the return type of the method matches the type of the variable in the calling program