Java programs are a collection of classes

- Each class is a *Template*, not an *Object*
- Can’t use a class until we create an instance (call “new”)
- Each class contains methods and data
03-1: Java Control Structures

- If:

  ```java
  if (<test>)
    <statement>
  or
  if (<test>)
    <statement1>
  else
    <statement2>
  ```

- A statement is either a single statement (terminated with ;), or a block of statements inside {}
What can we have as the test of an if statement?
- Boolean valued expression
03-3: *If test*

- What can we have as the test of an if statement?
  - boolean variable
  - comparison $x < y$ or $x \neq 0$
  - Combination of boolean expressions, using not (!), and (&&), or (||)
03-4: Boolean Variables

- Hold the value true or false
- Can be used in test (if, while, etc)

```java
boolean b;
boolean c;
b = true;
c = b || false;
b = (3 < 10) && (5 > 11);
c = !b && c;
```
What is (likely) wrong with the following code?

```c
if (x != 0)
    z = a / x;
    y = b / x;
```
What is (likely) wrong with the following code?

```java
if (x != 0) {
    z = a / x;
    y = b / x;
}
```

Moral: Always use {} in if statements, even if they are not necessary.
while loops

while(test)
{
    <loop body>
}

- Evaluate the test
- If the test is true, execute the body of the loop
- Repeat
- Loop body *may be executed* 0 times
03-8: do-while loops

do 
{ 
    <loop body>
} while (<test>);

- Execute the body of the loop
- If the test is true, repeat
- Loop body is *always* executed at least once
03-9: while vs. do-while

- What would happen if:
  - Found a while loop in a piece of code
  - Changed to a do-while (leaving body of loop and test the same)
- How would the execution be different?
03-10: while vs. do-while

- What would happen if:
  - Found a while loop in a piece of code
  - Changed to a do-while (leaving body of loop and test the same)

- How would the execution be different?
  - If the while loop were to execute 0 times, do-while will execute (at least!) one time
  - If the while loop were to execute 1 or more times, should to the same thing ...
    - ... except if the test had side effects (stay tuned for more on this in coming weeks)
for loops

for (<init>; <test>; <inc>)
{
    <body>
}

- Equivalent to:
<init>
while(<test>)
{
    <body>
    <inc>
}
for (number = 1; number < 10; number++)
{
    System.out.print("Number is " + number);
}

• Equivalent to:

number = 1;
while(number < 10)
{
    System.out.print("Number is " + number);
    number++;
}
Create a calculator class that has methods that allow you to:
- add 2 numbers
- multiply 2 numbers (without using the * operator!)
- calculate $x^n$ (power function)
03-14: Calculator Example II

- Add to previous calculator example:
  - Two instance variables, firstOperand and secondOperand
  - New versions of add, multiply, power that take as inputs the instance variables, and return appropriate values
You can have \( > 1 \) method with the same name
- As long as the rest of the method signature, number and types of parameters, are different
- Constructors can also be overloaded
public class Calculator {
    int firstOperand;
    int secondOperand;

    public Calculator() {
        this.firstOperand = 0;
        this.secondOperand = 0;
    }

    public Calculator(int first, int second) {
        this.firstOperand = first;
        this.secondOperand = second;
    }
}
public int power()
{
    return power(this.firstOperand, this.secondOperand);
}

public int power(int x, int y)
{
    int result;
    for (result = 1; y > 0; y--)
    {
        result = multiply(result, x);
    }
    return result;
}
Overloading Methods

- Note that the version of power without parameters called the version of power with parameters
- Why is that a good idea?
Overloading Methods

- Note that the version of power without parameters called the version of power with parameters.

- Why is that a good idea?
  - Both versions of power do the same thing.
  - If you change one, don’t need to change the other.
    - Big problem in industrial code – more than one code path that does the same thing, fix a bug in one, might not fix the same bug in the other.
File name needs to be `<ClassName>.java`

Class Calculator needs to be in file Calculator.java
- No spaces!
Static methods are very different from non-static methods.

Can be somewhat confusing to have both static and non-static methods in the same class.

We encourage a “Driver” class which contains a single static method main.

Could place the static main in one of the other classes in the project – code would compile and run just fine, though it is a little confusing.