Factorial

• Write a program that takes one positive integer input

• The program has a method named factorial that takes one integer parameter and returns an integer.
  
  • the integer returned is the factorial  \( 4! = 4 \times 3 \times 2 \times 1 \)

• The main method calls the factorial method and prints the result.

• Then change so that instead of factorial of the input, draw a random integer in 1..n, N times and print the factorial.
Methods and Scope

• Like "if" and "for" loops, methods have scope
• But, unlike those, a method is NOT within the scope of the caller

```java
class Scoper {
    public static void main(String[] args) {
        int anIntA = 5;
        int anIntB = 42;
        System.out.println(anIntA + " " + anIntB);
        changeIt();
        System.out.println(anIntA + " " + anIntB);
    }

    public static void changeIt() {
        int anIntA = 200;
        System.out.println(anIntA + " " + anIntB);
    }
}
```
Methods and Parameters

• Usually, you can change parameters, but those changes do not survive the end of the method

```java
public class ParamCh {
    public static void main(String[] args) {
        int anIntA = 5;
        int anIntB = 42;
        System.out.println(anIntA + " " + anIntB);
        changeIt(anIntB);
        System.out.println(anIntA + " " + anIntB);
    }

    public static void changeIt(int incomming) {
        System.out.println(anIntA + " " + incomming);
        int anIntA = 200;
        incomming = 45000;
        System.out.println(anIntA + " " + incomming);
    }
}
```
Averaging User Input

• Program:
  • get from user N, the number of integers to average
  • get N integers from user
  • compute average

• Methods:
  • get N integers
  • average of N integers
Arrays, Methods and Pointers

• When you make an array "x = new int[6]" what is stored in x is a pointer to the place where info is stored, not the place.
• When x is passed to a function, you pass the pointer.
• Any changes to the array live on after the method
• But, if you change the array pointer ...
• Choices:
  • initialize array in main and pass in empty
  • create and return array in method
Chalkboard

Lucas numbers and the Golden mean

• Program to get a single integer, N, from user (or command line)
• Method to calculate and return the first N Lucas numbers
  
  \[
  L_n := \begin{cases} 
  2 & \text{if } n = 0; \\
  1 & \text{if } n = 1; \\
  L_{n-1} + L_{n-2} & \text{if } n > 1. 
  \end{cases}
  \]

• In main, print \( L[n]/L[n-1] \) (as a double) for \( n=1 \ldots n=(N-1) \).
Overloading

why write only one!

• Can write several methods with the same name
  • must have
    • same return type
    • same modifiers
    • different parameters
• Java considers methods different if they have different signatures

• WHY would I ever want to do this???
  • Squaring
In summary

- Primitive datatypes are passed by value (copy)
- Arrays are passed by reference (alias)
  - So contents changes survive
    - if you do not change the pointer