Introduction to Methods and Interfaces

CS1: Java Programming
Colorado State University

Kris Brown, Wim Bohm and Ben Say

Liang, Introduction to Java Programming, Tenth Edition, (c) 2015 Pearson Education, Inc. All rights reserved.

User Defined Methods - motivation

- We want to write a program that manipulates areas of certain 2D shapes
  - rectangles, squares
  - circles, and spheres
- We do not want to write the expression for these areas every time we need to compute one
  - Similarly, we do not want to write one monster main method to do all the work!
  - We want to divide and conquer: separate logical groups of statements together in one construct

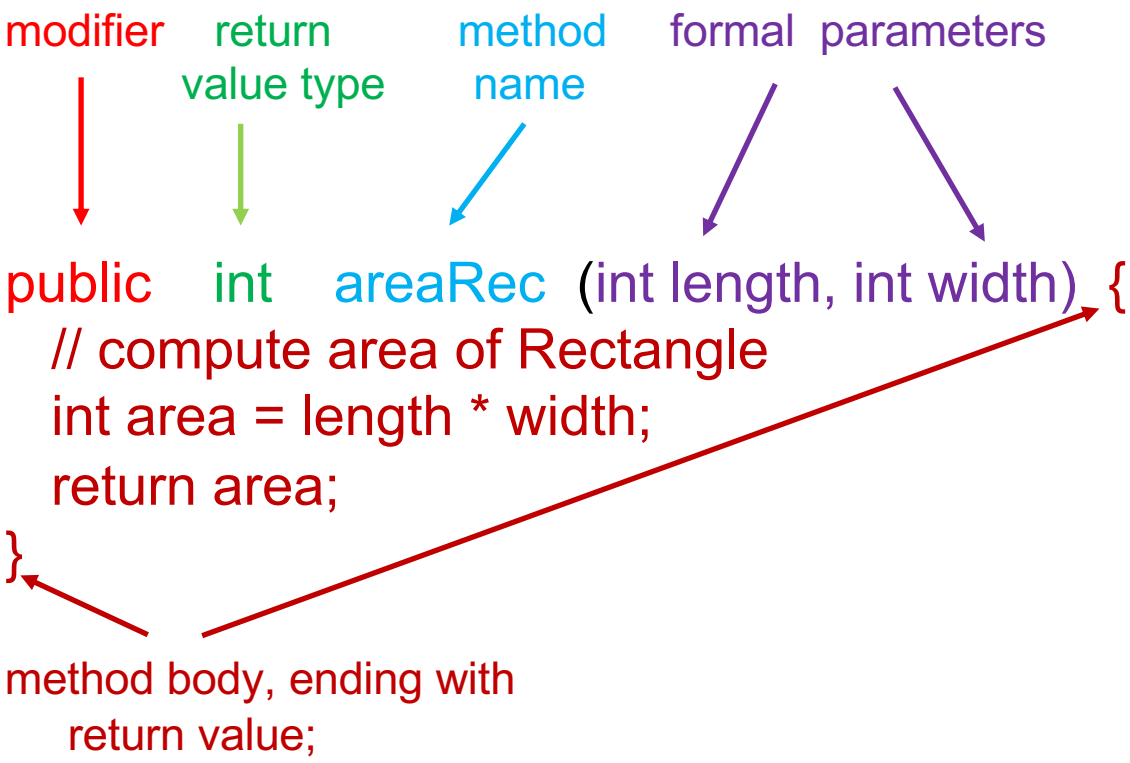
Liang, Introduction to Java Programming, Tenth Edition, (c) 2015 Pearson Education, Inc. All rights reserved.

Methods

- A **method** allows us to group a set of statements together into a logical operation
- There are two aspects to methods:
  - The method **definition**
    - A method is a collection of statements that are grouped together to perform an operation
  - The method **call**
    - Another method can now use the defined method to perform the operation

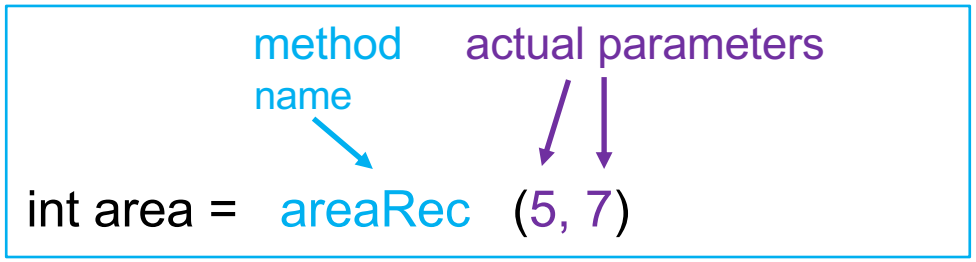
Method definition

A method is a collection of statements that are grouped together to perform an operation. Defining a method:



Calling a Method

A method is a called in another piece of code (main or another method). Calling a method:




```
// definition
public int areaRec(int length, int width){
   // compute area of Rectangle
   int area = length * width;
   return area;
}
```

The *Method signature* is the combination of the method name and the formal parameter list.

Method call: parameter passing

- When a method is called, the values of the actual parameters of the caller are passed (copied) to the formal parameters of the definition.
  - areaRec(5, 7) (in our example)
    passes 5 to length
    and 7 to width

Method return

- A method may return a value.
- The returnValueType is the data type of the value the method returns. If the method does not return a value, the returnValueType is the keyword void.
  - For example, the returnValueType in the main method is void.
- When a method call is finished it returns the returnValue to the caller. In our example code int area = areaRec(5,7)

  areaRec(5, 7) returns 35

Let's go check out the code . . .

Call Stack

In our example code

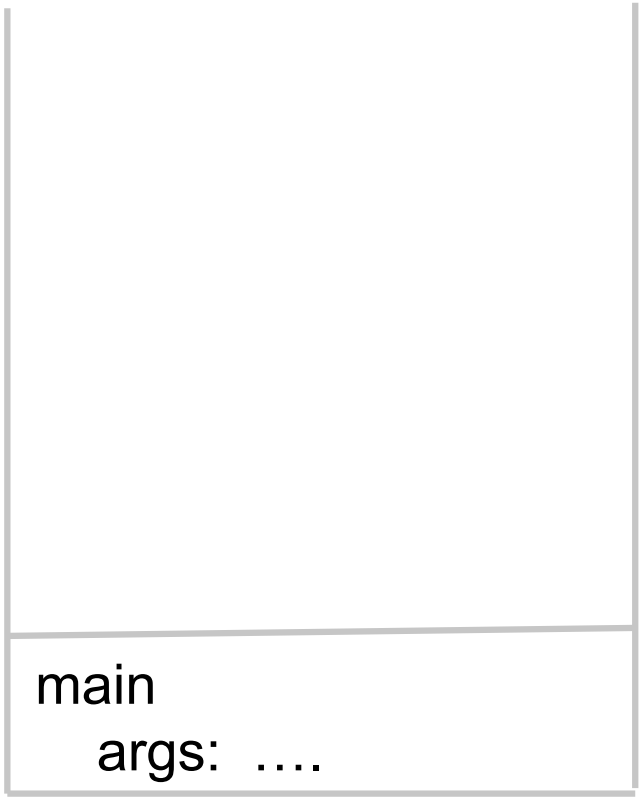
main called doRectangularShapes()

and

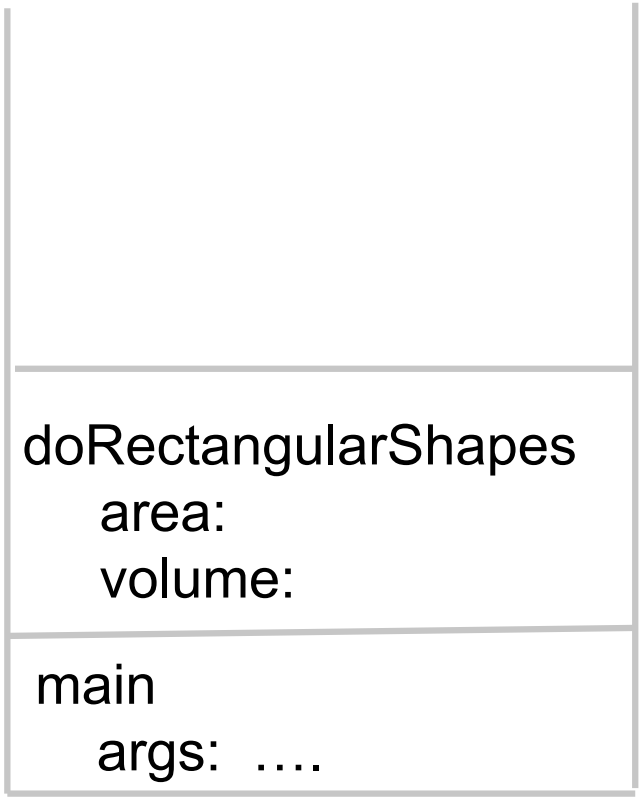
doRectangularShapes called areaRec(9,5)

When our program gets executed, a run time stack allows records called stack-frames to be stacked up and removed, thereby keeping track of the call history.

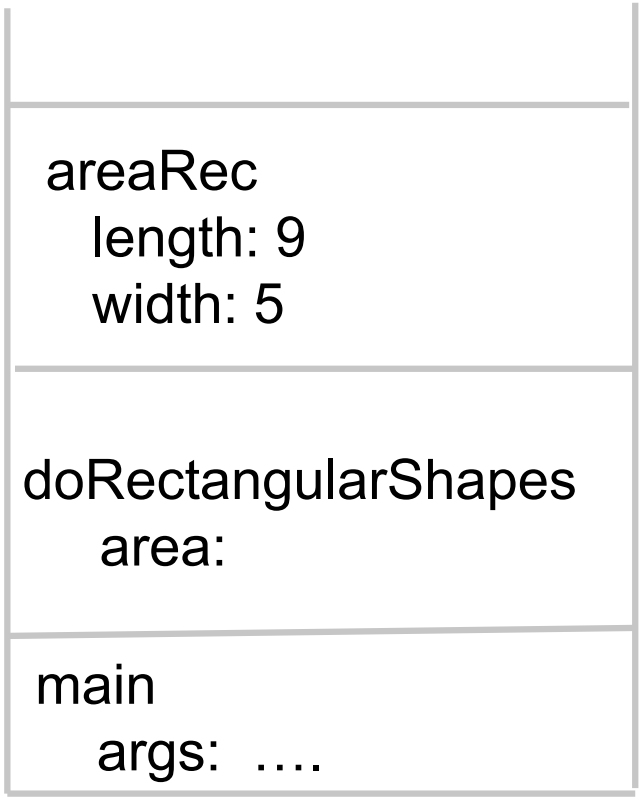
main starts



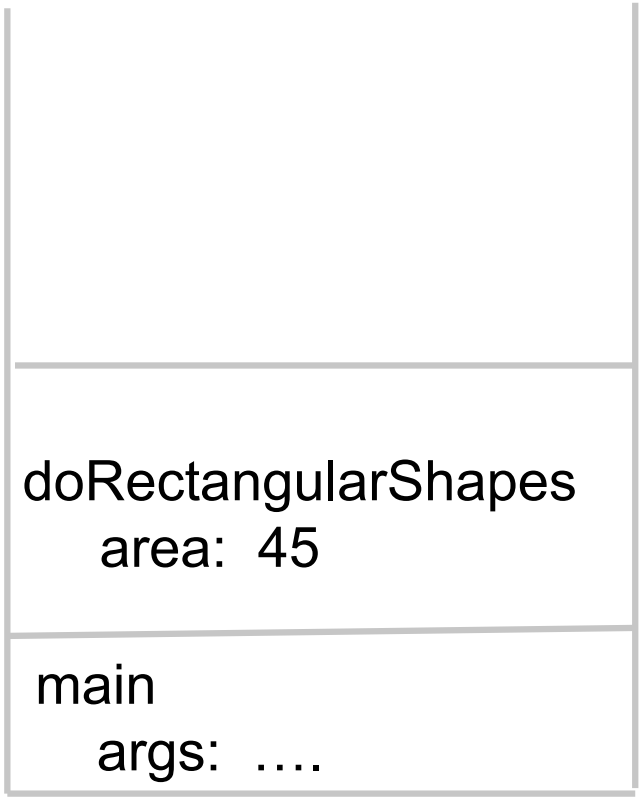
main calls doRectangularShapes()



doRectangularShapes calls areaRec(9,5)

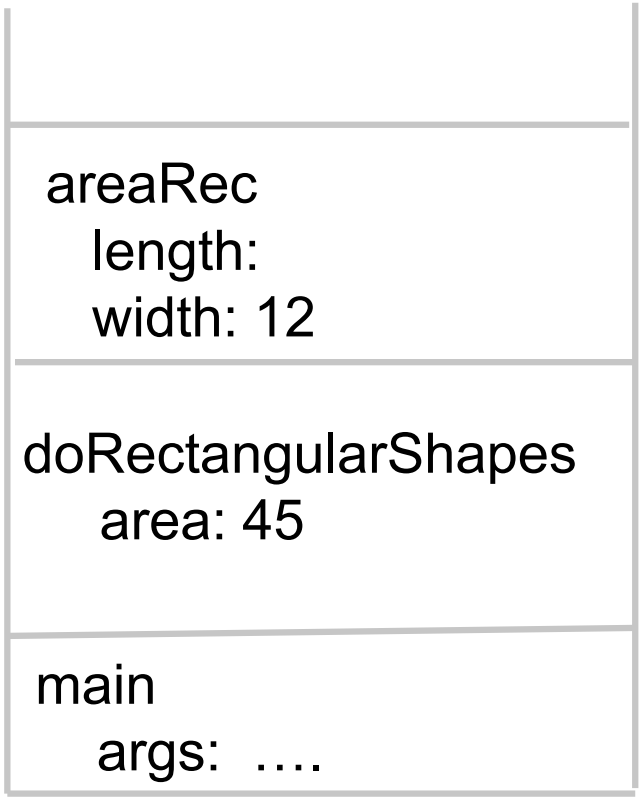


areaRec(9,5) returns 45
doRectangularShapes prints

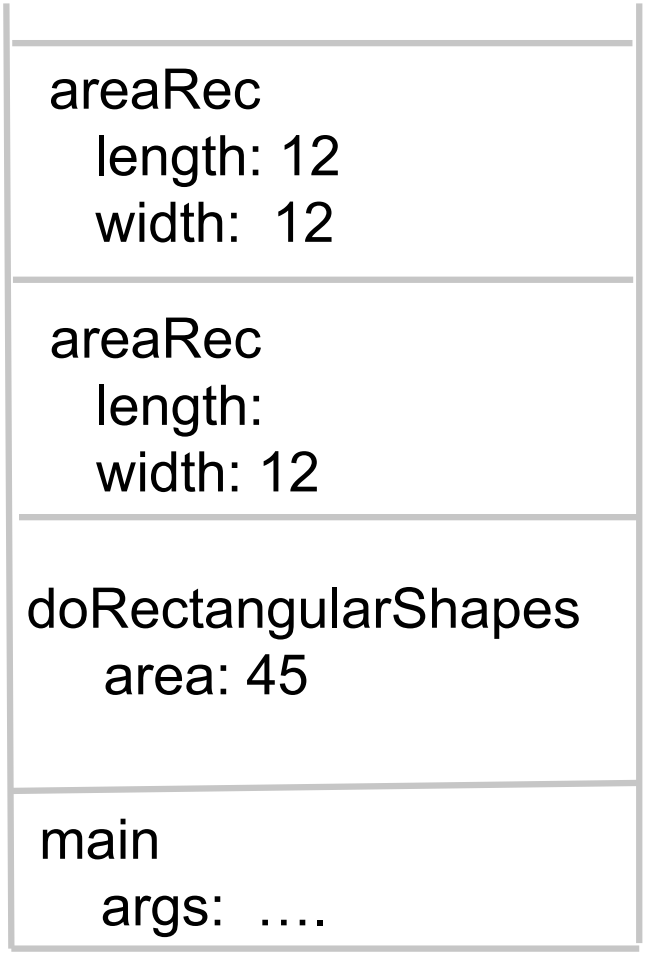



```
output:
   9 by 5 rectangle has area 45
```

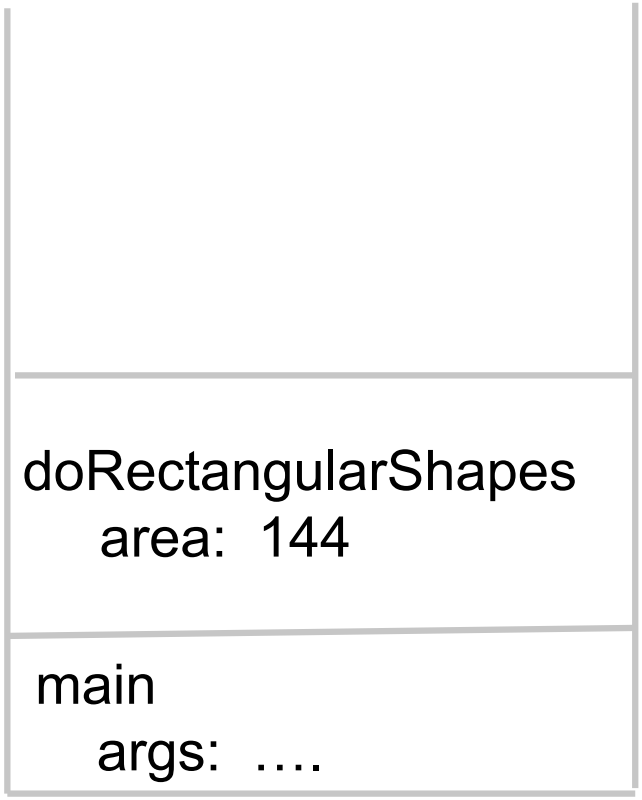
doRectangularShapes calls areaRec(12)



areaRec calls areaRec(12,12)

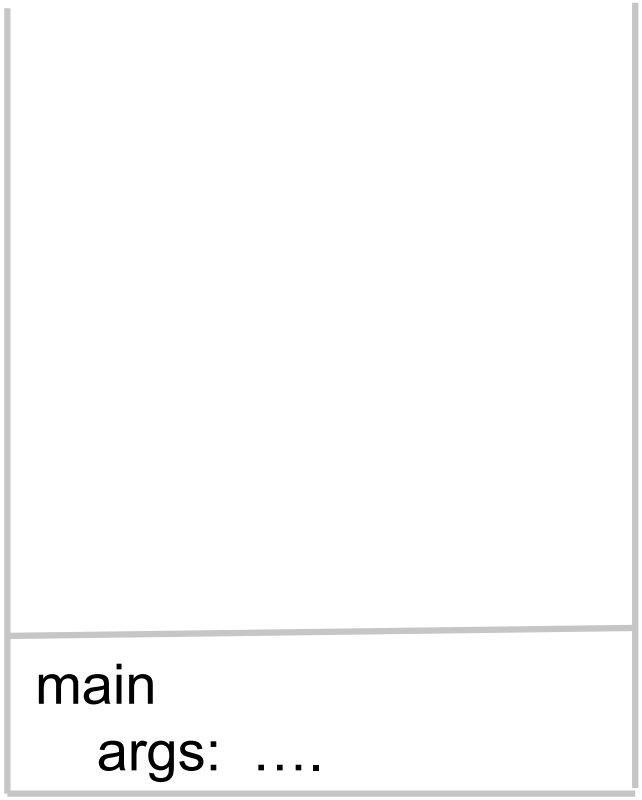


areaRec(12,12) returns 144
areaRec(12) returns 144
doRectangularShapes prints




```
output:
   square with width 12 has area 144
```

doRectangularShapes returns



Your turn!

- Read the program and trace what happens next
- Draw the run time stack with its stack frames for all the call / return events
- Write a program using perimeter using methods