Panel 1

Last time:

methods: perfect to solve small, well-defined tasks

double, int, boolean, char, void

```
public static ReturnType name (input list)
{
    stuff;
}
```

Methods only execute when called!

1

Panel 2

Ex:

```
public class Test
{
    public static double twice (double x)
    {
        return x+x;
    }

    public static void main (String args[])
    {
        double y = twice (2.0);
        System.out.println(twice(twice(2.0)));
    }
}
```

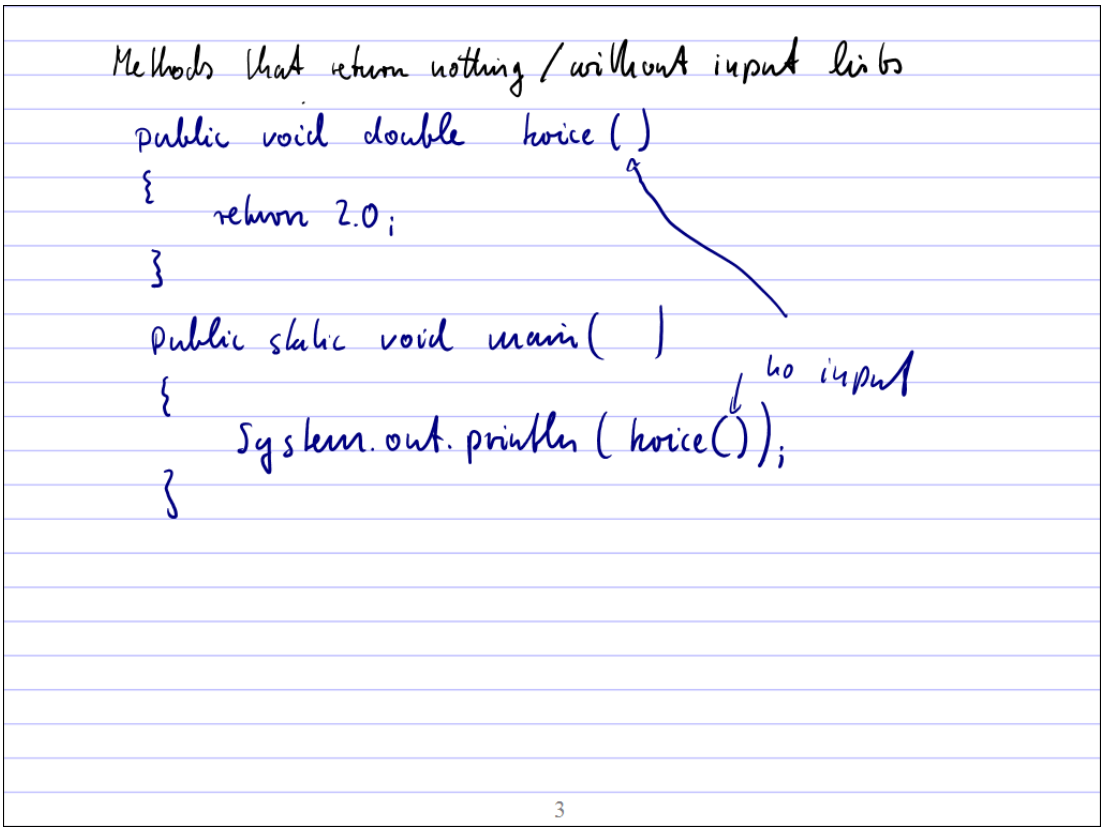
method 1

method 2

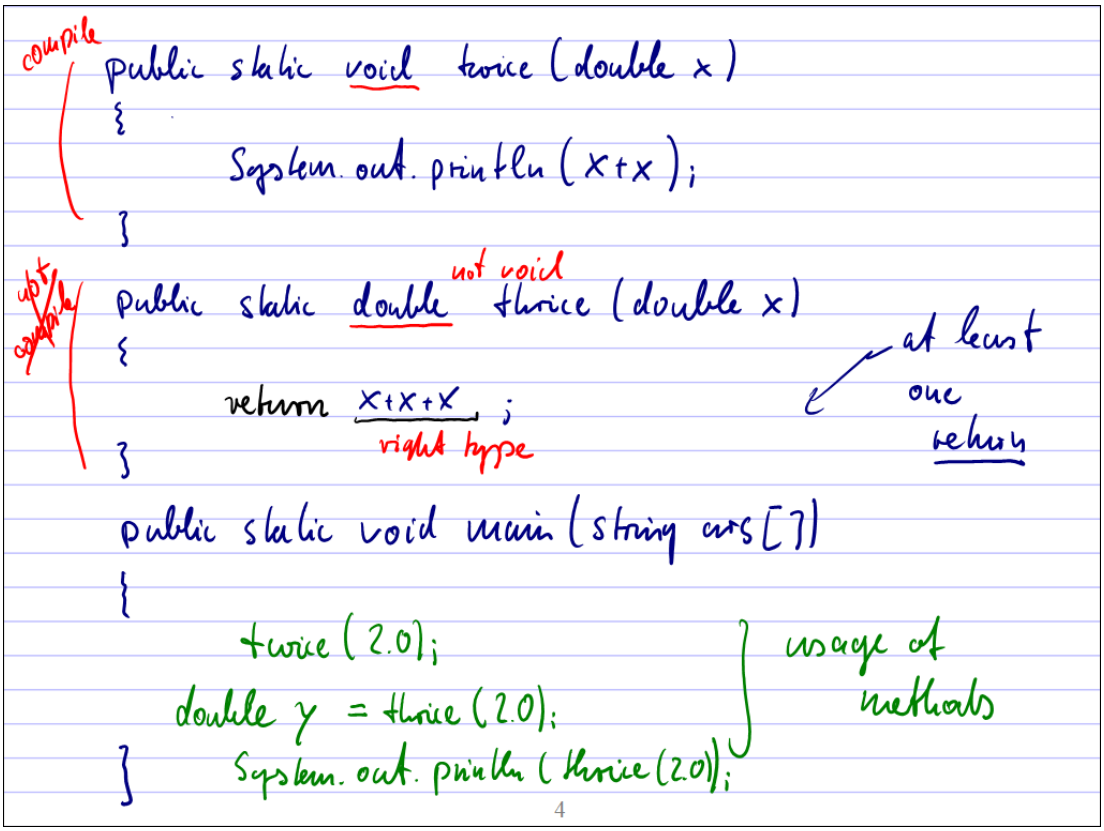
Execute program => automatically calls main method!

2

Panel 3



Panel 4



Panel 5

Methods that return void do not need the return keyword and can be called without handling their output.

Methods that return a type must use the return keyword and must handle the output somehow.

5

Panel 6

Ex: Write a program with a menu to convert to/from degree F/C.

Want:

title

a) From F to C
b) From C to F
c) Quit

=> 4 methods

1.) double convertFtoC(double f)
2.) double convertCtoF(double c)
3.) void showMenu()
4.) char getChoice()

main

Outline

4 tasks:

1) convert F to C
2) convert C to F
3) show menu
4) get user choice / a, b, or c

6

Panel 7

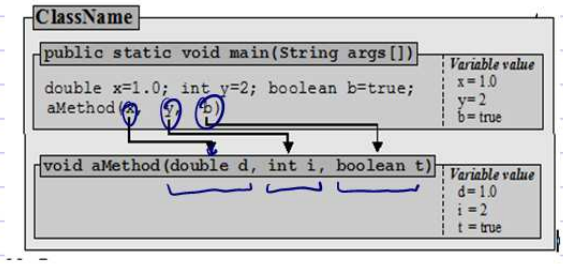
```
double convertFtoC(double f) {...return ...  }
double convertCtoF(double c) {...return ...  }
void showMenu()              {..System.out.println(....) }
char getChoice()             {...Console.readChar(), return }

public static void main(String args[])
{
        showMenu();
    char choice = getChoice();
        if (choice == 'a')
        {
                double x = Console.readDouble();
            double y=convertFtoC(x);
        }
}
```

7

Panel 8

Parameter passing



Parameters are matched in order

Types must be compatible

double x = aMethod(....)

Value of calling variable is copied into the variable of called method

8

Panel 9

Example:

```
public class ValueParameter
{
   public static void aMethod(int input)
   {
      System.out.println("Value of input at start of aMethod: " + input);
      input += 10;
      System.out.println("Value of input at end of aMethod: " + input);
   }

   public static void main(String args[])
   {
      int number = 10;
      System.out.println("Value before calling method: " +  number);
      aMethod(number);
      System.out.println("Value after calling the method: " + number);
   }
}
```

10 ✓
10 ✓
20 ✓
10 ✓

if a method needs to change a value so that the change impacts calling method, it must use return!

9

Panel 10

A method without a return statement, i.e. a method that has void return type, can not modify value of a variable in any other method.

A method with return type non-void can return exactly one value!

10

Panel 11

Fields and Local Variables

In Java, there are two types of variables:

- *Variables that are defined inside a class but not inside a method. These variables are called fields. The type of a field, currently, needs to be prefaced with the keyword* `static`[9]*. All methods of a class have direct access to field variables.*
- *Variables that are declared inside a method when needed. These variables are called local variables. Input variables declared in the method header are local variables of that method.*

```
public class FieldsAndVars
{
   static double aField = 10.0;

   public static void main(String args[])
   {
      double int aLocalVar = 20;
      for (int i = 0; i < 10; i++)
         System.out.println("i = " + i);
   }
}
```

one field

local vars

a Local Var

i

11

Panel 12

```
1   public class TrickyScope
2   {  static int x = 1;
3
4      public static void method1()
5      {  int x = 10;
6         x += 10;
7      }
8      public static void method2(int y)
9      {  x += 10;
10        y += 20;
11        z += 30;
12     }
13     public static void main(String args[])
14     {  int z = 3;
15        method1();
16        method2(z);
17        x += 1;
18        y += 2;
19        z += 3;
20        System.out.println(x);
21        System.out.println(y);
22        System.out.println(z);
23     }
24  }
```

one field: x

x valid

z

y is local here

3

shows:

z = ~~3~~ 6

x = ~~1~~ ~~11~~ 12

12

6

12

Panel 13

Example 2.09: Compound interest computation using fields

Write a program to compute the interest and balances if $10,000.00 are deposited for 10 years in an account at an interest rate of 6.5%, compounded once per year. The program should display the interest for each year as well as the yearly amount including interest.

use main method and another "compute Interest" method, as well as fields and variables.

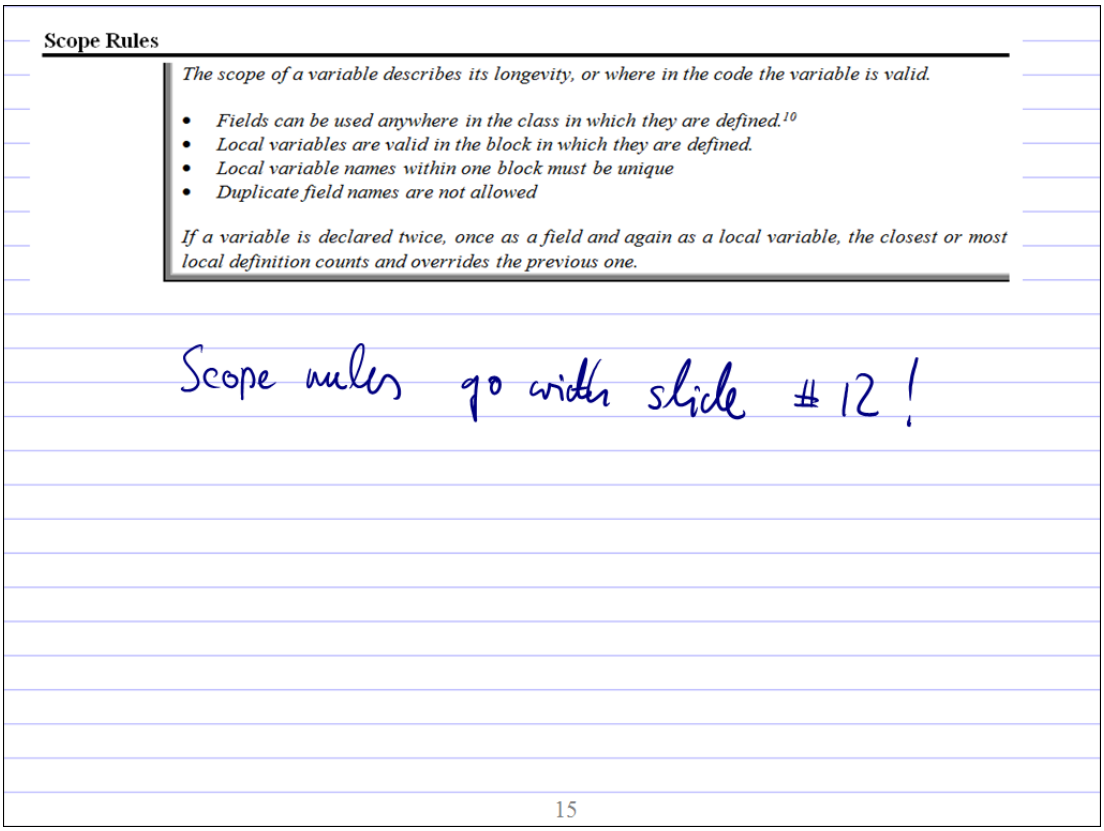
13

Panel 14

~~Strings~~ next time

14

Panel 15


Scope Rules
The scope of a variable describes its longevity, or where in the code the variable is valid.
Fields can be used anywhere in the class in which they are defined.10
Local variables are valid in the block in which they are defined.
Local variable names within one block must be unique
Duplicate field names are not allowed
If a variable is declared twice, once as a field and again as a local variable, the closest or most local definition counts and overrides the previous one.
Scope rules go with slide #12!
15