Method Output
&
Variable Scope

Methods Recap

- User doesn’t need to know how a method works to use it. Just need to know the input type(s) and output type. This is called *“abstraction”*
- A method can have no input or lots of input (**parameters**).
- A method can have no output type (**void**) or ONE output type.

Method Output

- A method can output nothing (**`void`**) or **ONE** thing (int, double, String, etc.)
- Output values are sent back with a **`return`** statement followed by the value to return.
- When a **`return`** statement is executed, the method is exited immediately.

Return Values

```
public static type methodName (parameters)
{
        variables;
        statements;
        return EXPRESSION;  // required
} // end method
```

Call the method the same way as before.

`void` means “no type”

```
public class Square3
{
    public static void printSquare(double x)
    {
        System.out.println(x*x);
    }  // end printSquare method

    public static void main( String [ ] args)
    {
        printSquare(5);
    }  // end main method
}  // end Square3 class
```

Variable Scope

- A **block** of code is often identified with { }
- Variables “live” in the block where they are defined. Where the variable “lives” is referred to as the variable’s **scope**
- Scope starts where variable is declared; ends at the end of the block in which it was declared
- Method parameters and variables are defined inside the method; those variables do not “live” outside the method

```
public class SquareChange {
    public static void square(int x)  {
        System.out.println(“printSquare x = “ + x);
        x = x * x;
        System.out.println(“printSquare x = “ + x);
    }  // end of parameter x scope
    public static void main( String [ ] args)  {
        int x = 5;
        System.out.println(“main x = “ + x);
        printSquare(x);
        System.out.println(“main x = “ + x);
    }  // end of local variable x scope
}  // end SquareChange class
```

Return Values

```
public static ______ methodName (parameters)
{
        variables;
        statements;
        __________________________;
} // end method
```

Call the method the same way as before.

`void` means “no type”