

# Introduction to Programming

*with Java, for Beginners*

## Local vs. Global Variables

## Scope

- *Scope* means the area of code in which an entity is known (or alive)
  - Mainly concerned with **variables** and **methods**
  - Which parts of the program can access them?
- Sometimes scope is **explicitly** designated with a keyword
  - **private**: known only within the class
  - **public**: known outside of (and within) the class
  - Note that Methods have explicit scope
- Other times it is **implicitly** designated by location

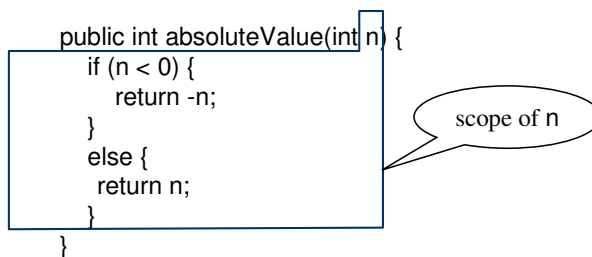
ESE112

1

## Implicit Scope: Method Parameters

- A method parameter is an “input variable”
- **Scope**: the method in which it is defined
- No other method can access (read/write) it

```
public int absoluteValue(int n) {  
    if (n < 0) {  
        return -n;  
    }  
    else {  
        return n;  
    }  
}
```



ESE112

2

## Implicit Scope: Local Variables

- A “local variable” is defined within a method body { }
  - They are inherently private to the method in which they are defined
  - We don't use public/private for local variables
- It may be defined in a block { } within a method body
- **Scope**: point of declaration to end of closest enclosing block

ESE112

3

## Example of Local Variables

```
public int isLarger(int x, int y){
    if (x > y) {
        int larger = x;
    }

    else {
        int larger = y;
    }
    return larger;
}
```

scope of larger

scope of a different larger

Illegal: not declared in current scope

ESE112

4

## Another Example

```
int fibonacci(int limit) {
    int first = 1;
    int second = 1;
    while (first < limit) {
        System.out.print(first + " ");
        int next = first + second;
        first = second;
        second = next;
    }
    System.out.println( );
}
```

second first limit

ESE112

5

## Demonstrating Scope of Local Variable

- By default, *copies* of parameter are sent to a method

```
public static void main(String [] args){
    int x = 0;
    System.out.println("In main: x = " + x);
    foo(x);
    System.out.println("In main: x = " + x);
}
```

Output:  
In main: x = 0  
In foo: x = 0  
In foo: x = 5  
In main: x = 0

```
public static void foo(int x){
    System.out.println("In foo: x = " + x);
    x = 5;
    System.out.println("In foo: x = " + x);
}
```

ESE112

6

## Global Variable

- Is variable declared outside of all methods within class
- Written at beginning of class
- Syntax  
*public static data-type* variableName  
E.g. public static int count;
- Scope
  - Is *accessible (read and modified)* by all methods within the class

## Example

```
//Example use of global variable.  
public class Global{
```

```
    //Global Variable  
    public static final double PI = 3.14; //PI is global, however it is final and hence  
    cannot be changed
```

```
    //Calculate area of a circle  
    public static double area(double r){  
        return r * r * PI;  
    }
```

```
    //Calculate circumference of a circle  
    public static double circum(double r){  
        return 2 * PI * r;  
    }  
}
```

## Accessing Global Variables Outside of Class

- To access global variable outside of the class it is declared in (as long as it is declared public)
  - `Classname.globalVariableName`
- From previous slide
  - `double a = Global.PI * 10;`
- Java has a built in class called Math class.
  - This class declares to Global constants PI & E
  - So you can directly use these instead of creating your own
  - Usage: Math.PI for PI and Math.E for E