Scope of Local Variables

- A local variable: a variable defined inside a method.
- Scope: the part of the program where the variable can be referenced.
- The scope of a local variable starts from its declaration and continues to the end of the block that contains the variable. A local variable must be declared before it can be used.

Ref: Liang, Introduction to Java Programming
You can declare a local variable with the same name multiple times in different non-nesting blocks in a method, but you cannot declare a local variable twice in nested blocks.

Ref: Liang, Introduction to Java Programming
A variable declared in the initial action part of a `for` loop header has its scope in the entire loop. But a variable declared inside a `for` loop body has its scope limited in the loop body from its declaration and to the end of the block that contains the variable.

Ref: Liang, Introduction to Java Programming
public static void method1() {
  .
  .
  for (int i = 1; i < 10; i++) {
  .
    .
    int j;
  .
  }
}

The scope of i

The scope of j

Ref: Liang, Introduction to Java Programming
It is fine to declare `i` in two non-nesting blocks

```java
public static void method1() {
    int x = 1;
    int y = 1;

    for (int i = 1; i < 10; i++) {
        x += i;
    }

    for (int i = 1; i < 10; i++) {
        y += i;
    }
}
```

It is wrong to declare `i` in two nesting blocks

```java
public static void method2() {
    int i = 1;
    int sum = 0;

    for (int i = 1; i < 10; i++) {
        sum += i;
    }
}
```

Ref: Liang, Introduction to Java Programming
// Fine with no errors

public static void main(String[] args) {
    int x = 1;
    int y = 1;
    // i is declared
    for (int i = 1; i < 10; i++) {
        x += i;
    }
    // i is declared again
    for (int i = 1; i < 10; i++) {
        y += i;
    }
}

Ref: Liang, Introduction to Java Programming
// With errors
public static void main(String[] args){
    int x = 1;
    int y = 1;
    for (int i = 1; i < 10; i++) {
        int x = 0;
        x += i;
    }
}

Ref: Liang, Introduction to Java Programming
Function

- Functions break large computing tasks into smaller ones.
- Appropriate functions hide details of operation from parts of the program that don’t need to know about them.
- Clarifying the logic flows.
Opening Problem

Find the sum of integers from 1 to 10, from 20 to 30, and from 35 to 45, respectively.
int sum = 0;
for (int i = 1; i <= 10; i++)
    sum += i;
System.out.println("Sum from 1 to 10 is " + sum);

sum = 0;
for (int i = 20; i <= 30; i++)
    sum += i;
System.out.println("Sum from 20 to 30 is " + sum);

sum = 0;
for (int i = 35; i <= 45; i++)
    sum += i;
System.out.println("Sum from 35 to 45 is " + sum);
public static int sum(int i1, int i2) {
    int sum = 0;
    for (int i = i1; i <= i2; i++)
        sum += i;
    return sum;
}

discontinued

public static void main(String[] args) {
    System.out.println("Sum from 1 to 10 is " + sum(1, 10));
    System.out.println("Sum from 20 to 30 is " + sum(20, 30));
    System.out.println("Sum from 35 to 45 is " + sum(35, 45));
}
A method is a collection of statements that are grouped together to perform an operation.

**Define a method**

```java
public static int max(int num1, int num2) {
    int result;
    if (num1 > num2)
        result = num1;
    else
        result = num2;
    return result;
}
```

**Invoke a method**

```java
int z = max(x, y);
```

Ref: Liang, Introduction to Java Programming
Function

It may need neither an input or an outcome
The variables defined in the method header are known as **formal parameters**.
When a method is invoked, you pass a value to the parameter. This value is referred to as *actual parameter or argument*.
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`.

```
public static int max(int num1, int num2) {
    int result;
    if (num1 > num2) {
        result = num1;
    } else {
        result = num2;
    }
    return result;
}
```

Ref: Liang, Introduction to Java Programming
public static int max(int num1, int num2) {
    int result;
    if (num1 > num2)
        result = num1;
    else
        result = num2;
    return result;
}

public static void main(String[] args) {
    int i = 5;
    int j = 2;
    int k = max(i, j);
    System.out.println("The maximum between "+i+" and "+j+" is "+k);
}
A return statement is required for a value-returning method. The method shown below in (a) is logically correct, but it has a compilation error because the Java compiler thinks it possible that this method does not return any value.

Public static int sign(int n) {  
  if (n > 0)  
    return 1;  
  else if (n == 0)  
    return 0;  
  else if (n < 0)  
    return -1;  
}

To fix this problem, delete if (n < 0) in (a), so that the compiler will see a return statement to be reached regardless of how the if statement is evaluated.

Ref: Liang, Introduction to Java Programming
public static void nPrintln(String message, int n) {
    for (int i = 0; i < n; i++)
        System.out.println(message);
}

Suppose you invoke the method using
    nPrintln(“Welcome to Java”, 5);
What is the output?

Suppose you invoke the method using
    nPrintln(“Computer Science”, 15);
What is the output?

Can you invoke the method using
    nPrintln(15, “Computer Science”);
This program demonstrates passing values to the methods.
Example

```java
public static int Sum_Two_Numbers(int nNum1, int nNum2)
{
    int nRet;
    nRet = nNum1 + nNum2;
    return nRet;
}

public static void main(String[] args)
{
    System.out.println(Sum_Two_Numbers(1, 10));
}
```
public static void Sum_Two_Numbers(int nNum1, int nNum2) {
    int nRet;
    nRet = nNum1 + nNum2;
    nNum1++;  
    nNum2++;  
}

public static void main(String[] args) {
    int nNum1 = 1, nNum2 = 2;
    Sum_Two_Numbers(nNum1, nNum2);
    System.out.println(nNum1);
    System.out.println(nNum2);
}
Guess It

do{
    Display_Game_Info
    Generate_a_random_key (between 1 and 100)
    do{
        do{
            Ask_a_number
        }while(! The_input_is_valid);  
        if (correct) Display_Congr_and_Break
        else Give_a_Hint
    }while (try is less than or equal to 6);
}while (Ask.want_to_one_more_game == 'y');
Rock-Paper-Scissors

while (Display_Game_Info_and_get_UserInput() != 'q')
{
    Display_Rock_Paper_Scissors
    Generate_a_random_number
    Obtain a user input
    if (!Is_Validate_Input(cInput))
        continue; // Restart the game again.
    Determine_Who_Wins
    Display_Result
}