

Block/Scope

- **A pair of { }** defines a block with it's own scope

```
public static void main( String args[] )
{
    Scanner kbd = new Scanner( System.in );
    int limit = 55;        // belongs to main
    int speed = 77;        // belongs to main
    if ( speed > limit )    // issue fine
    {
        int fine = 125;
        System.out.println("Enter your name" );
        String name = kbd.next();
        System.out.println( name + ". Your fine is: $" + fine );
        System.out.println("You were " + (limit-speed) + " over limit!");
        // we can reference limit because it is an ancestor block
    }
    System.out.println( "You better slow down " + name ); // ILLEGAL
    // cannot reference name, name is only visible inside the -if- block and
    // any blocks that might be nested inside the -if- block
} //END main
```

Block/Scope

Blocks can be nested to an arbitrary depth. Variable names cannot be re-used in a nested block but can be re-used in unrelated blocks. Analogy: You don't name two kids in the same family with the same name -but- two different families can have kids whose names are identical.

You should distinguish the two identical variables as the **fine** belonging to the 1st -if- block or the **fine** belonging to the 2nd -if- block. Just like two kids named Joe in two different families. The distinction then becomes clear.

```
int limit = 55;
int speed = 77;          // this n belongs to main
if ( speed > limit ) // calculate fine
{
    int overLimit = limit-speed;
    if (overLimit > 30 )
    {
        int fine = 200;
        System.out.println("You owe " + fine );
    }
    else
    {
        int fine = 100; // this variable is not the same as in above block
        System.out.println("You owe " + fine );
    }
}
```

Typical uses of blocks

```
public static void main(String args[])
{
    Scanner kbd = new Scanner( System.in );
    int sum = 0;
    while (sum < 80) // while loop soon to be explained fully
    {
        System.out.print( "Enter a number: " );
        int val = kbd.nextInt();
        sum+= val;    // i.e.  sum = sum + val
    }
    System.out.println( "Sum is: " + sum );
}
```

- Rule of usage: Never declare a variable to have wider scope than needed to accomplish its purpose. Our **val** variable is only needed inside the accumulation loop. Thus we don't declare it at same level as **sum**.