File Edit View Help

File Edit View Help

Recursion

File Edit View Help

Recursion

- Solve a problem by solving smaller versions of the same problem
  - Divide and Conquer Algorithms
  - Backtracking
- Recursive Method – a method that calls itself
  - "Loop-like"
  - Call stack
- Recursive Methods Required
  - Halting Condition
  - Recursive Call

Example

```
public static void countDown(int i)
{
     if(i < 0 )//Halting Condition
          return;
     System.out.println(i);
     countDown(i-1);//Recursive Call
}
```

File Edit View Help

Recursion Count Down

```java
public static void countDown(int i)
{
    if(i < 0 )//Halting Condition
        return;
    System.out.println(i);
    countDown(i-1);//Recursive Call
}
```

Call Stack in Memory

File Edit View Help

Recursion Count Down

5

```
public static void countDown(int i)
{
    if(i < 0 )//Halting Condition
        return;
    System.out.println(i);
    countDown(i-1);//Recursive Call
}
```

Call Stack in Memory

Recursion Count Down

5

```
public static void countDown(int i)
{
    if(i < 0 )//Halting Condition
        return;
    System.out.println(i);
    countDown(i-1);//Recursive Call
}
```

Call Stack in Memory

Recursion Count Down

5

```
public static void countDown(int i)
{
    if(i < 0 )//Halting Condition
        return;
    System.out.println(i);
    countDown(i-1);//Recursive Call
}
```

Call Stack in Memory

Recursion Count Down

```
public static void countDown(int i)
{
    if(i < 0 )//Halting Condition
        return;
    System.out.println(i);
    countDown(i-1);//Recursive Call
}
```

5

Console
5

Call Stack in Memory

Recursion Count Down

```
public static void countDown(int i)
{
    if(i < 0 )//Halting Condition
        return;
    System.out.println(i);
    countDown(i-1);//Recursive Call
}
```

5

```
Console
5
```

Call Stack in Memory

Recursion Count Down

```
public static void countDown(int i)
{
    if(i < 0 )//Halting Condition
        return;
    System.out.println(i);
    countDown(i-1);//Recursive Call
}
```

4

Console
5

Call Stack in Memory

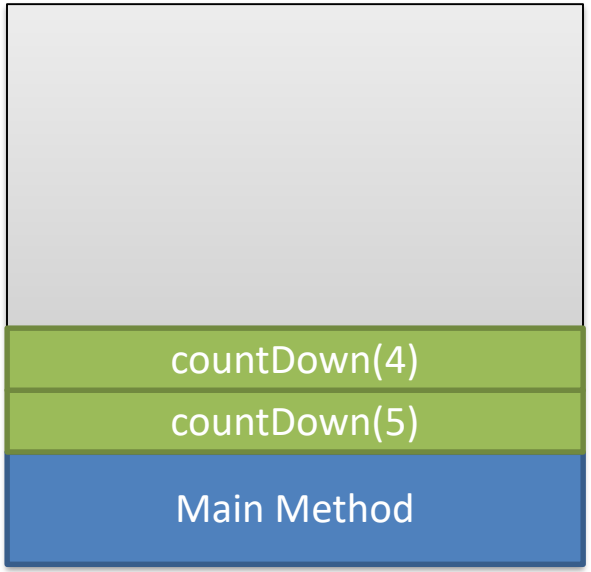
Recursion Count Down

```
public static void countDown(int i)
{
    if(i < 0 )//Halting Condition
        return;
    System.out.println(i);
    countDown(i-1);//Recursive Call
}
```

4

Console
5

Call Stack in Memory



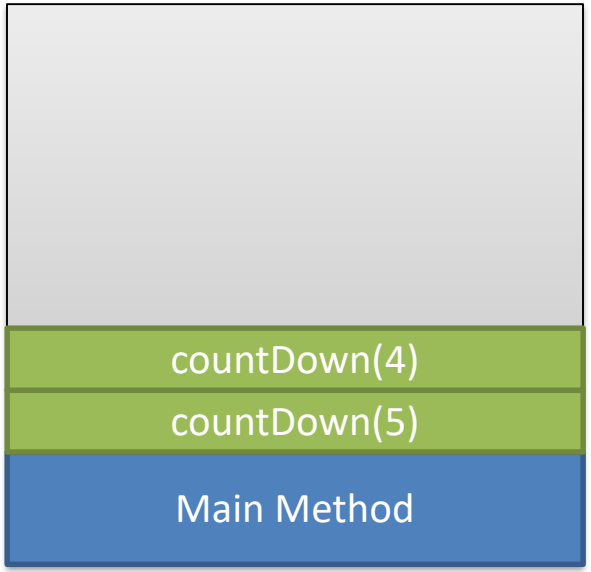
Recursion Count Down

```
public static void countDown(int i)
{
    if(i < 0 )//Halting Condition
        return;
    System.out.println(i);
    countDown(i-1);//Recursive Call
}
```

4

Console
5

Call Stack in Memory



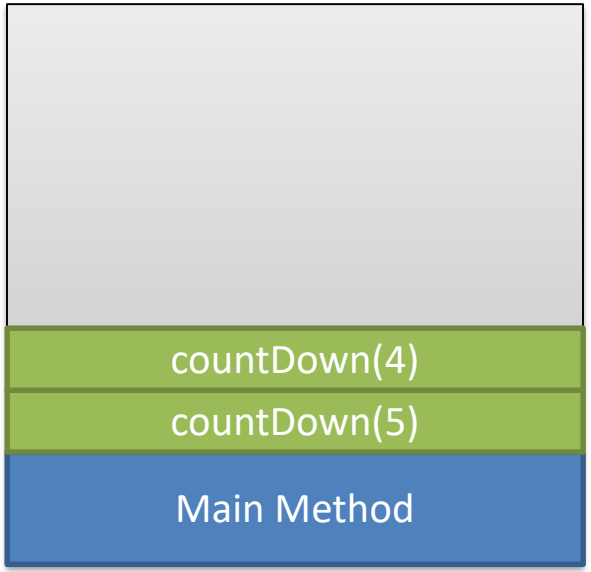
Recursion Count Down

```
public static void countDown(int i)
{
    if(i < 0 )//Halting Condition
        return;
    System.out.println(i);
    countDown(i-1);//Recursive Call
}
```

4

```
Console
5
4
```

Call Stack in Memory



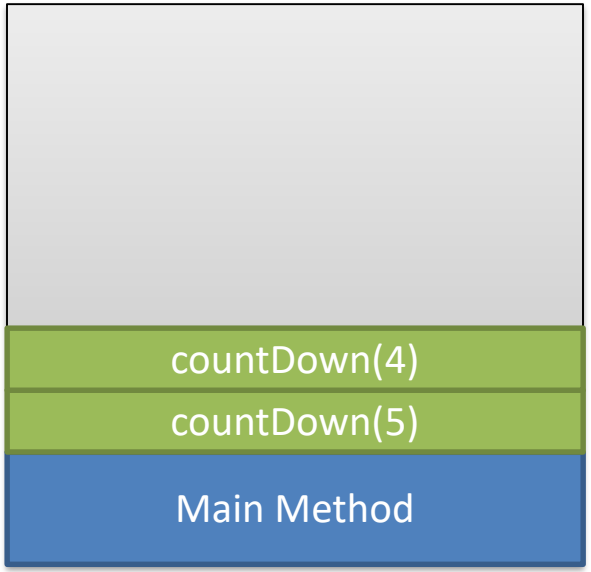
Recursion Count Down

```
public static void countDown(int i)
{
    if(i < 0 )//Halting Condition
        return;
    System.out.println(i);
    countDown(i-1);//Recursive Call
}
```

4

```
Console
5
4
```

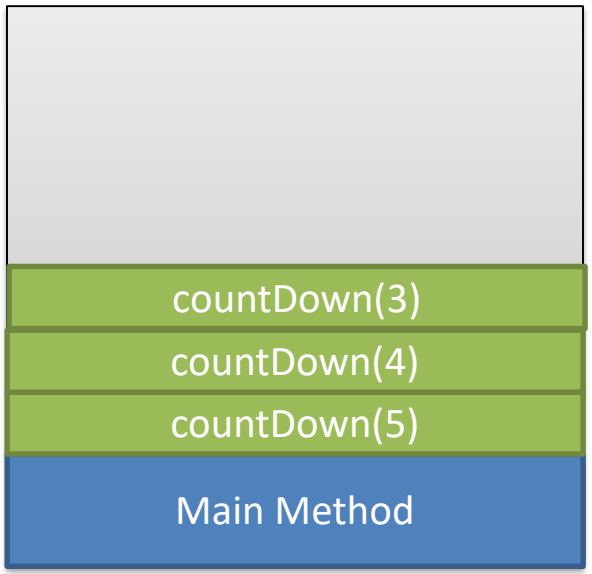
Call Stack in Memory



Recursion Count Down

```
3
public static void countDown(int i)
{
    if(i < 0 )//Halting Condition
        return;
    System.out.println(i);
    countDown(i-1);//Recursive Call
}
```

```
Console
5
4
```



File Edit View Help

A Few Recursive Steps Later...

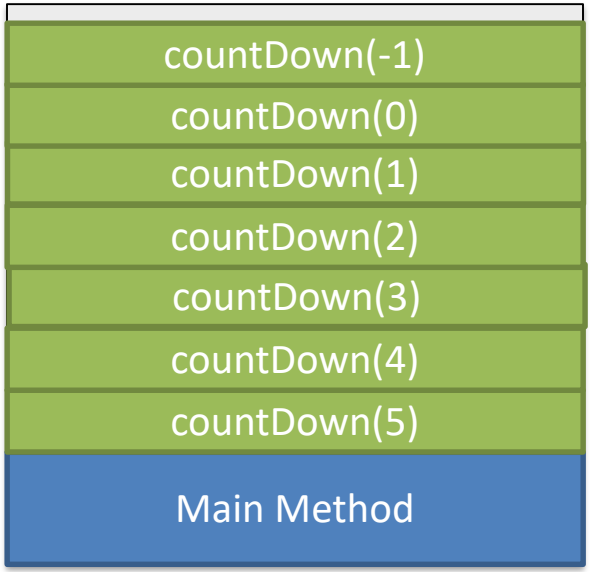
Recursion Count Down

-1

```
public static void countDown(int i)
{
    if(i < 0 )//Halting Condition
        return;
    System.out.println(i);
    countDown(i-1);//Recursive Call
}
```

```
Console
5
4
3
2
1
0
```

Call Stack in Memory



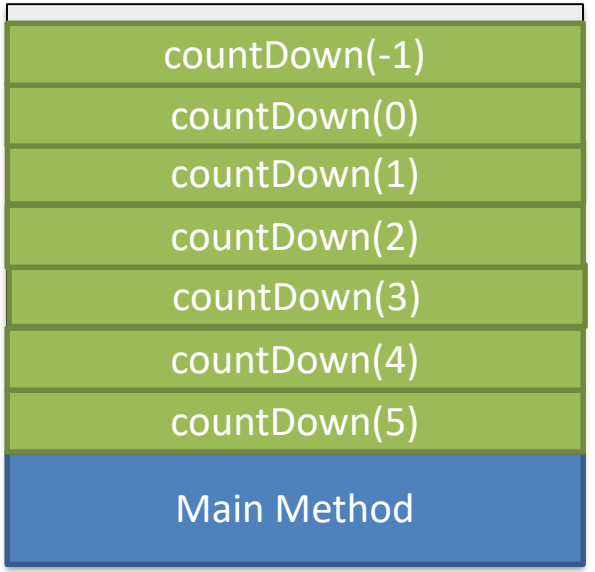
Recursion Count Down

-1

```
public static void countDown(int i)
{
    if(i < 0 )//Halting Condition
        return;
    System.out.println(i);
    countDown(i-1);//Recursive Call
}
```

```
Console
5
4
3
2
1
0
```

Call Stack in Memory



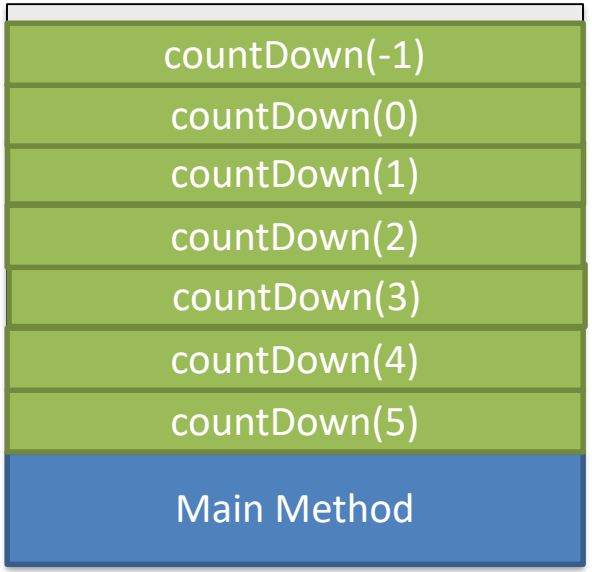
Recursion Count Down

-1

```
public static void countDown(int i)
{
    if(i < 0 )//Halting Condition
        return;
    System.out.println(i);
    countDown(i-1);//Recursive Call
}
```

```
Console
5
4
3
2
1
0
```

Call Stack in Memory



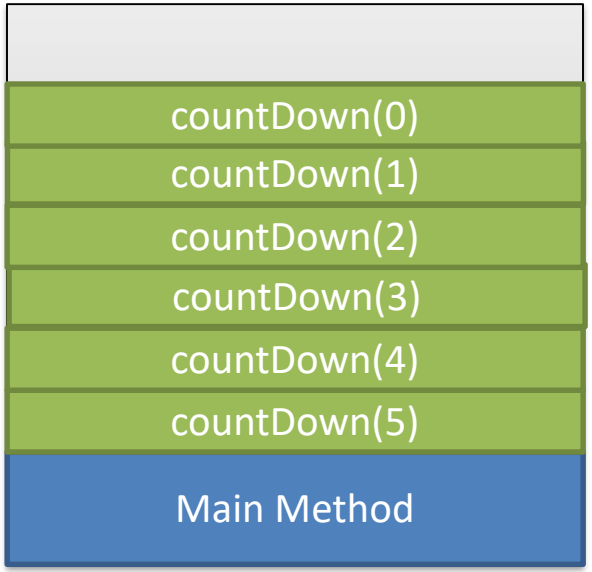
Recursion Count Down

```
public static void countDown(int i)
{
    if(i < 0 )//Halting Condition
        return;
    System.out.println(i);
    countDown(i-1);//Recursive Call
}
```

0

```
Console
5
4
3
2
1
0
```

Call Stack in Memory



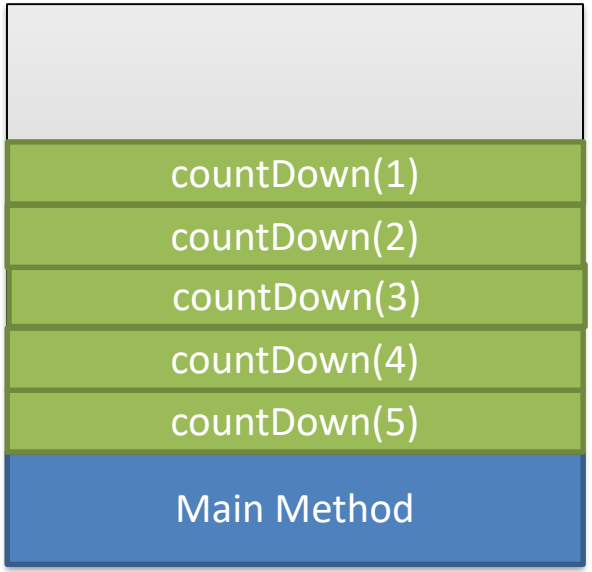
Recursion Count Down

1

```
public static void countDown(int i)
{
    if(i < 0 )//Halting Condition
        return;
    System.out.println(i);
    countDown(i-1);//Recursive Call
}
```

```
Console
5
4
3
2
1
0
```

Call Stack in Memory



Recursion Count Down

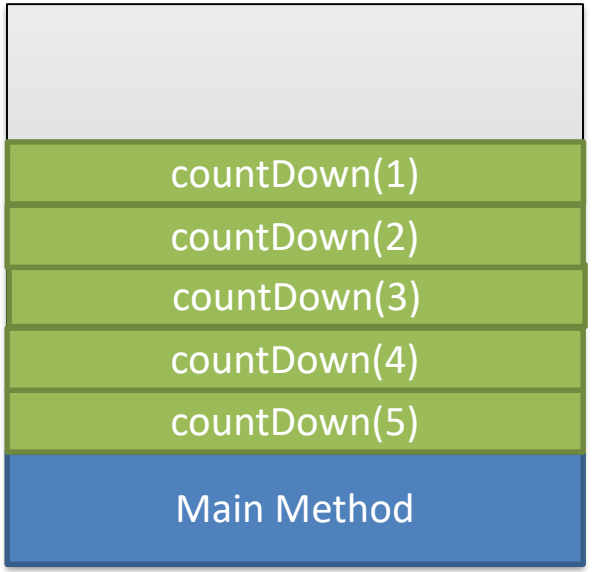
1

```
public static void countDown(int i)
{
    if(i < 0 )//Halting Condition
        return;
    System.out.println(i);
    countDown(i-1);//Recursive Call
}
```

Console

```
5
4
3
2
1
0
```

Call Stack in Memory

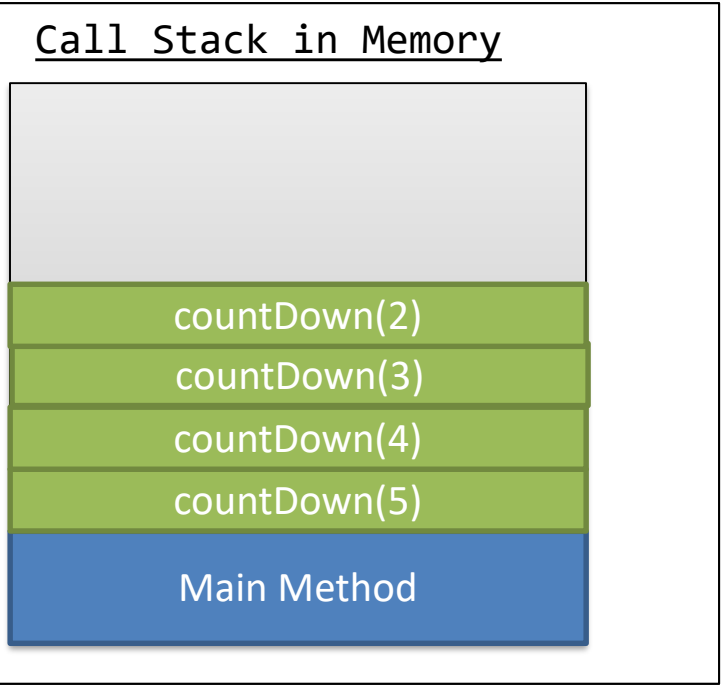


Recursion Count Down

```
public static void countDown(int i)
{
    if(i < 0 )//Halting Condition
        return;
    System.out.println(i);
    countDown(i-1);//Recursive Call
}
```

2

```
Console
5
4
3
2
1
0
```

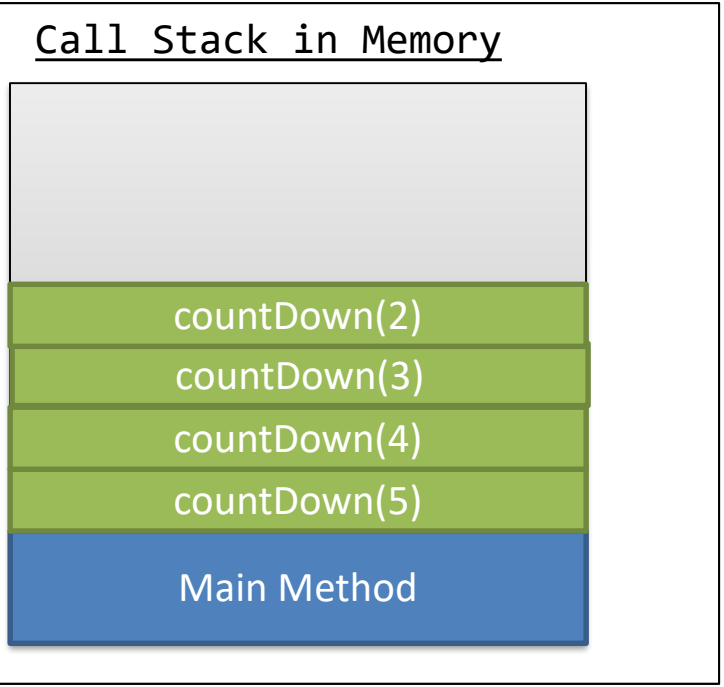


Recursion Count Down

```
public static void countDown(int i)
{
    if(i < 0 )//Halting Condition
        return;
    System.out.println(i);
    countDown(i-1);//Recursive Call
}
```

2

```
Console
5
4
3
2
1
0
```



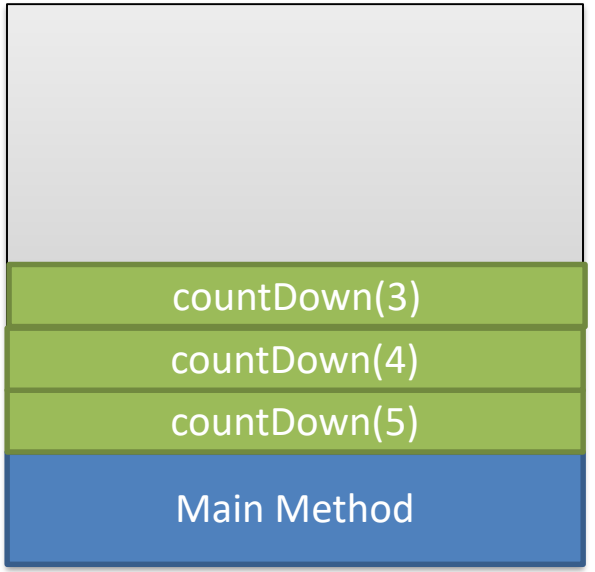
Recursion Count Down

```
public static void countDown(int i)
{
    if(i < 0 )//Halting Condition
        return;
    System.out.println(i);
    countDown(i-1);//Recursive Call
}
```

3

```
Console
5
4
3
2
1
0
```

Call Stack in Memory



File Edit View Help

A Few Returns Later...

Recursion Count Down

```
public static void countDown(int i)
{
    if(i < 0 )//Halting Condition
        return;
    System.out.println(i);
    countDown(i-1);//Recursive Call
}
```

5

```
Console
5
4
3
2
1
0
```

Call Stack in Memory

Recursion Count Down

```
public static void countDown(int i)
{
    if(i < 0 )//Halting Condition
        return;
    System.out.println(i);
    countDown(i-1);//Recursive Call
}
```

5

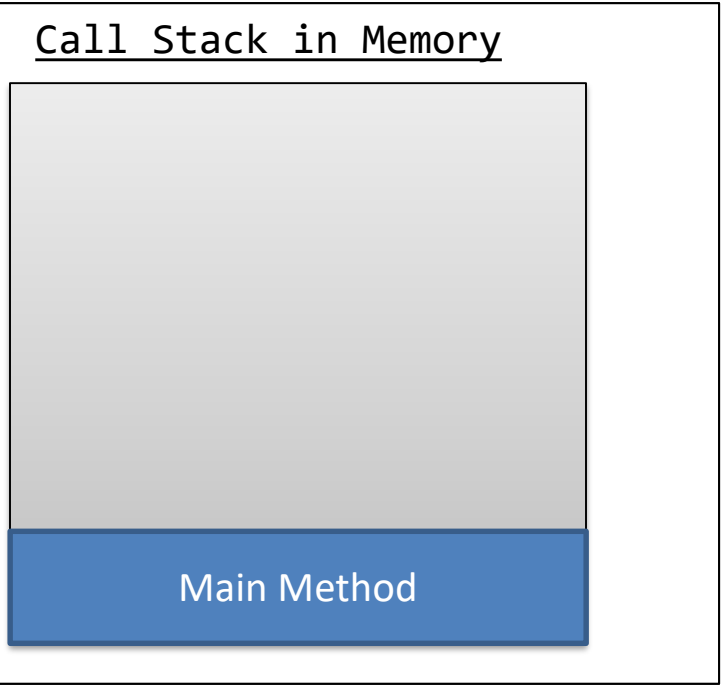
```
Console
5
4
3
2
1
0
```

Call Stack in Memory

Recursion Count Down

```java
public static void countDown(int i)
{
    if(i < 0 )//Halting Condition
        return;
    System.out.println(i);
    countDown(i-1);//Recursive Call
}
```

```
Console
5
4
3
2
1
0
```



File Edit View Help

Recursion
Factorial

$$n! = \begin{cases} 1 & if\ n = 0 \\ (n-1)! \times n & if\ n > 0 \end{cases}$$

File Edit View Help

Recursion Factorial

```java
public static int factorial(int number)
{
    if(number == 0)//Halting stopping condition
        return 1;
    else
        return number*factorial(number-1);//Recursive call
}
```

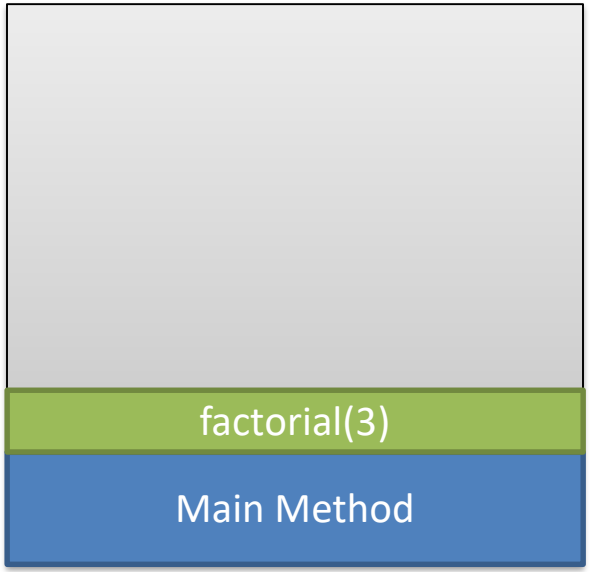
Call Stack in Memory

Recursion Factorial

3

```
public static int factorial(int number)
{
    if(number == 0)//Halting stopping condition
        return 1;
    else
        return number*factorial(number-1);//Recursive call
}
```

Call Stack in Memory

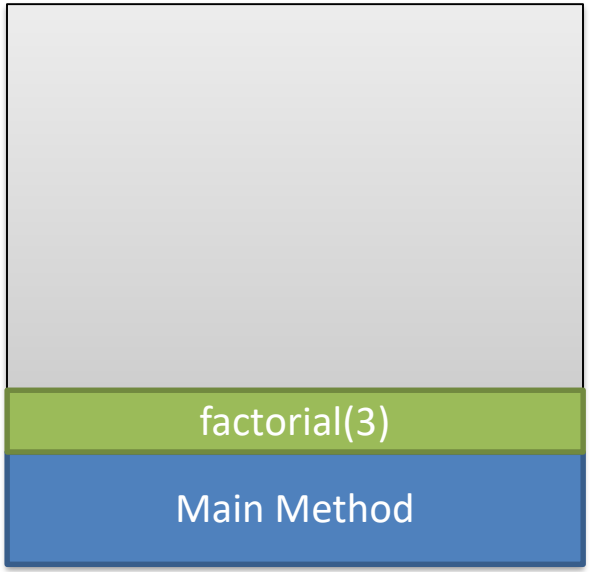


Recursion Factorial

3

```
public static int factorial(int number)
{
    if(number == 0)//Halting stopping condition
        return 1;
    else
        return number*factorial(number-1);//Recursive call
}
```

Call Stack in Memory



Recursion Factorial

3

```
public static int factorial(int number)
{
    if(number == 0)//Halting stopping condition
        return 1;
    else
        return number*factorial(number-1);//Recursive call
}
```

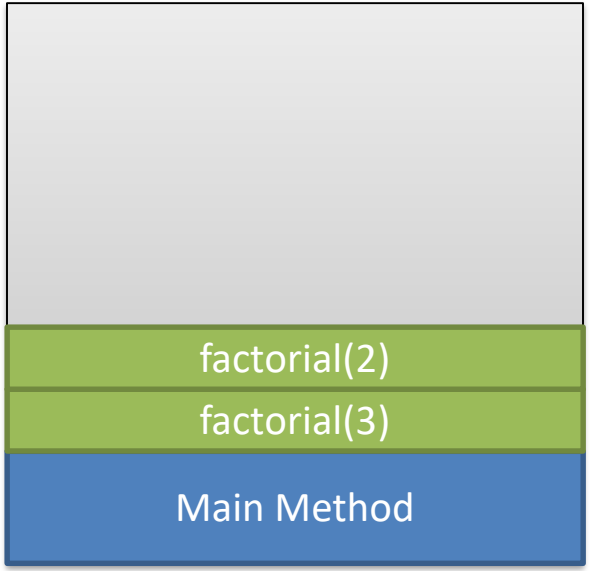
Call Stack in Memory

Recursion Factorial

2

```
public static int factorial(int number)
{
    if(number == 0)//Halting stopping condition
        return 1;
    else
        return number*factorial(number-1);//Recursive call
}
```

Call Stack in Memory

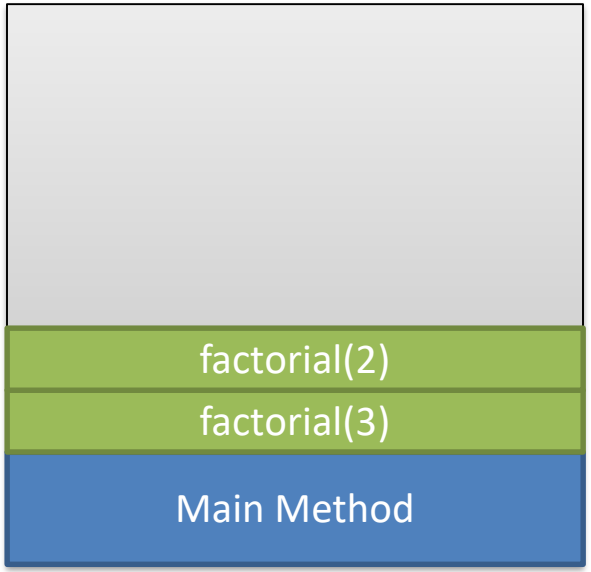


Recursion Factorial

2

```
public static int factorial(int number)
{
    if(number == 0)//Halting stopping condition
        return 1;
    else
        return number*factorial(number-1);//Recursive call
}
```

Call Stack in Memory

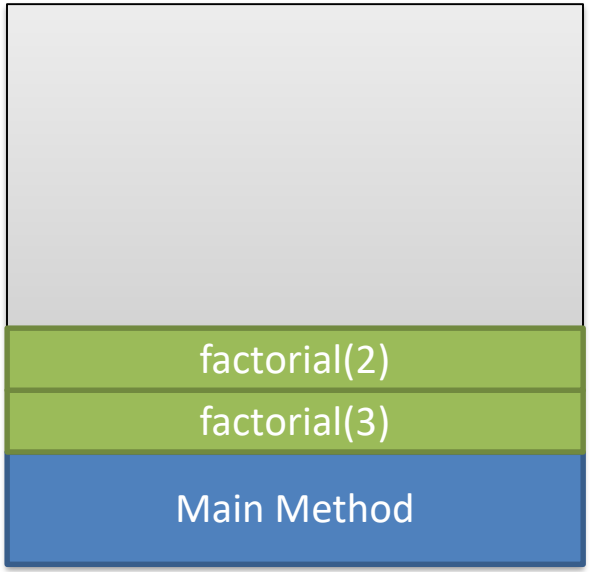


Recursion Factorial

2

```
public static int factorial(int number)
{
    if(number == 0)//Halting stopping condition
        return 1;
    else
        return number*factorial(number-1);//Recursive call
}
```

Call Stack in Memory

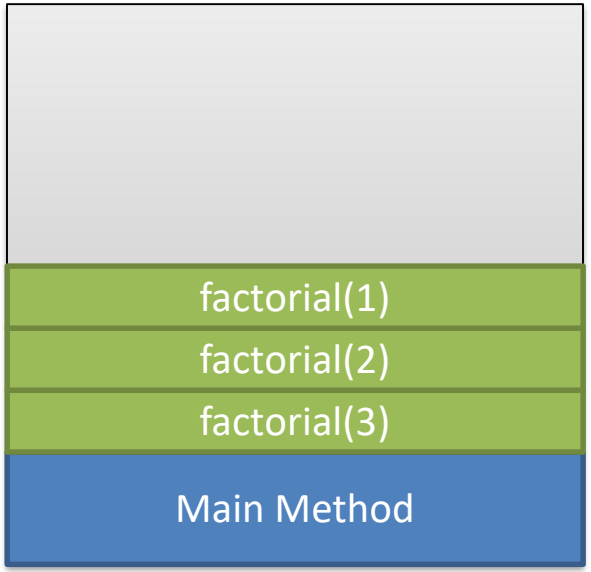


Recursion Factorial

1

```
public static int factorial(int number)
{
    if(number == 0)//Halting stopping condition
        return 1;
    else
        return number*factorial(number-1);//Recursive call
}
```

Call Stack in Memory

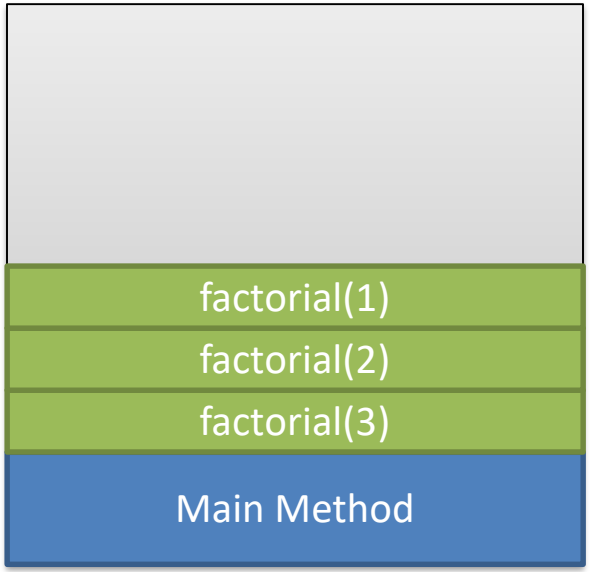


Recursion Factorial

1

```
public static int factorial(int number)
{
    if(number == 0)//Halting stopping condition
        return 1;
    else
        return number*factorial(number-1);//Recursive call
}
```

Call Stack in Memory

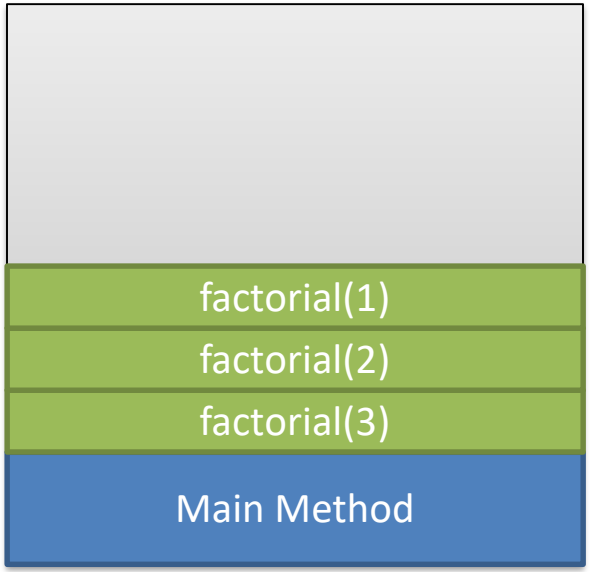


Recursion Factorial

1

```
public static int factorial(int number)
{
    if(number == 0)//Halting stopping condition
        return 1;
    else
        return number*factorial(number-1);//Recursive call
}
```

Call Stack in Memory

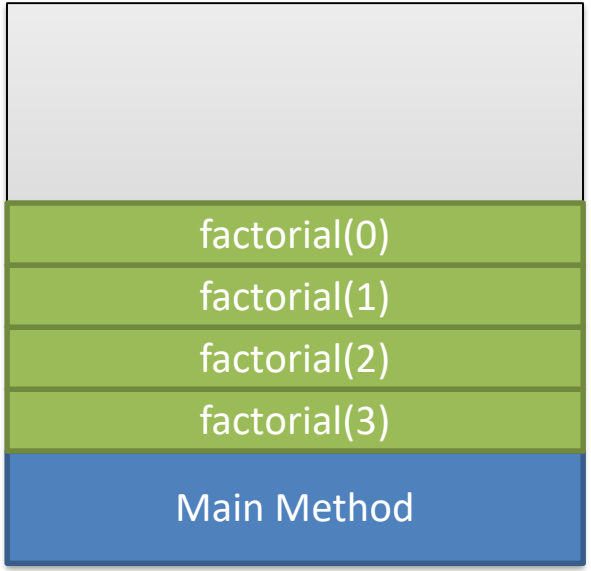


Recursion Factorial

0

```
public static int factorial(int number)
{
    if(number == 0)//Halting stopping condition
        return 1;
    else
        return number*factorial(number-1);//Recursive call
}
```

Call Stack in Memory

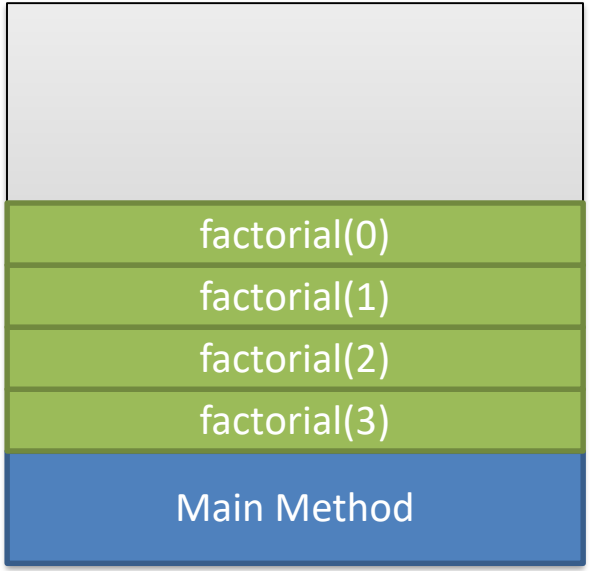


Recursion Factorial

0

```
public static int factorial(int number)
{
    if(number == 0)//Halting stopping condition
        return 1;
    else
        return number*factorial(number-1);//Recursive call
}
```

Call Stack in Memory

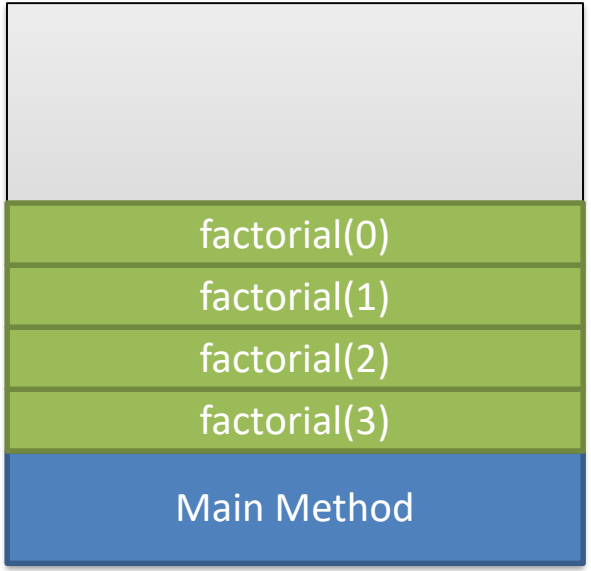


Recursion Factorial

0

```
public static int factorial(int number)
{
    if(number == 0)//Halting stopping condition
        return 1;
    else
        return number*factorial(number-1);//Recursive call
}
```

Call Stack in Memory



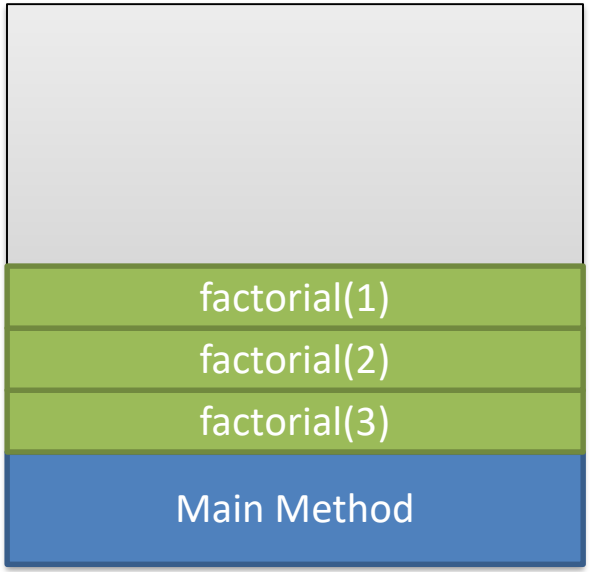
Recursion Factorial

1

```
public static int factorial(int number)
{
    if(number == 0)//Halting stopping condition
        return 1;
    else
        return number*factorial(number-1);//Recursive call
}
```

1

Call Stack in Memory



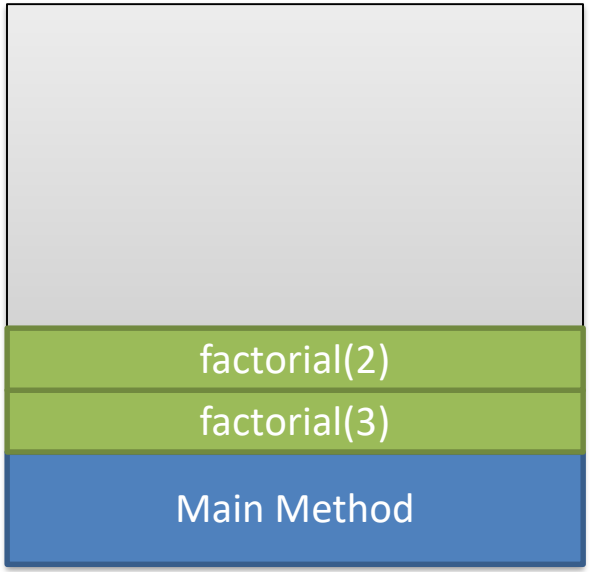
Recursion Factorial

```
public static int factorial(int number)
{
    if(number == 0)//Halting stopping condition
        return 1;
    else
        return number*factorial(number-1);//Recursive call
}
```

2

1

Call Stack in Memory



File Edit View Help

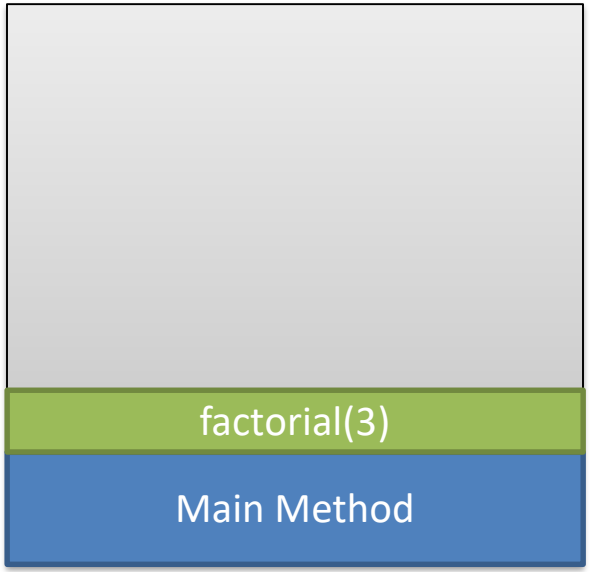
Recursion Factorial

3

```
public static int factorial(int number)
{
    if(number == 0)//Halting stopping condition
        return 1;
    else
        return number*factorial(number-1);//Recursive call
}
```

2

Call Stack in Memory



Recursion Factorial

```
public static int factorial(int number)
{
    if(number == 0)//Halting stopping condition
        return 1;
    else
        return number*factorial(number-1);//Recursive call
}
```

Return Value = 6

Call Stack in Memory

File Edit View Help

Recursion
GCD

$$\gcd(a, 0) = a$$
$$\gcd(a, b) = \gcd(b, a\ MOD\ b)$$

File Edit View Help

Recursion Fibonacci

$$F_n = F_{n-1} + F_{n-2}$$
$$F_1 = 1$$
$$F_2 = 1$$

Fibonacci Number (n)	1	2	3	4	5	6	7	8	…
Value	1	1	2	3	5	8	13	21	…

File Edit View Help

Sierpinski's Carpet

Concept

- Cut area in to 9 equal squares
  - 3 Horizontal
  - 3 Vertical
- Fill in the Center Square
- Repeat this process for the 8 surrounding squares until a limit has been reached
  - Recursive Depth
  - Pixel Limit

Example