

# **CSE 1322**

## **Module 4 – Part 1**

Recursion: The Function Stack



# Function Stack – What is it?

- Also known as Call Stack.
- It is a special region in memory that we use to temporarily store the information of active functions (methods) calls during the execution of the program.

# Function Stack – Stack

- We are going to cover this later in the semester.
- A stack is a Last-In, First-Out (LIFO) data structure.
- This means that the last item added, is the first one removed.
- Think of a stack of pancakes, most likely you start eating from the pancake at the top.

# Function Stack – Stack

- We are going to cover this later in the semester.
- A stack is a Last-In, First-Out (LIFO) data structure.
- This means that the last item added, is the first one removed.
- Think of a stack of pancakes, most likely you start eating from the pancake at the top.
- Meaning that the one of the top pancake was placed last in the plate, while the bottom pancake was the first one placed in the plate.

# Function Stack

- The Function Stack behaves the same.
- The first function added into the stack will usually always be the first item in the stack.
- While the main method is still active, our program is still active.
- Whenever we do a method call, we add another function into the stack.
- When the method call is resolved, we delete it from the stack.

# Function Stack

- The running function will always be at the “top” of the stack.
- Any functions below the top stack, is “waiting” for the top stack to resolve.

# Function Stack

```
public class Main {  
    public static void method2(){  
        System.out.println("Method 2 was called");  
    }  
  
    public static void method1(){  
        System.out.println("Method 1 was called");  
        method2();  
        System.out.println("Method 1 still alive");  
    }  
  
    public static void main(String[] args) {  
        method2();  
        method1();  
        System.out.println("Finished!");  
    }  
}
```

*Week-8/Function-Stack/Main.java*

# Function Stack

```
public class Main {  
    public static void method2(){  
        System.out.println("Method 2 was called");  
    }  
  
    public static void method1(){  
        System.out.println("Method 1 was called");  
        method2();  
        System.out.println("Method 1 still alive");  
    }  
  
    public static void main(String[] args) {  
        method2();  
        method1();  
        System.out.println("Finished!");  
    }  
}
```

main()  
1. Call method2()  
2. Call method1()  
3. Print "Finished!"

*Week-8/Function-Stack/Main.java*





# Function Stack

```
public class Main {  
    public static void method2(){  
        System.out.println("Method 2 was called");  
    }  
  
    public static void method1(){  
        System.out.println("Method 1 was called");  
        method2();  
        System.out.println("Method 1 still alive");  
    }  
  
    public static void main(String[] args) {  
        method2();  
        method1();  
        System.out.println("Finished!");  
    }  
}
```

method2()
1. Print "Method 2 was called"
main()
1. Call method2()
2. Call method1()
3. Print "Finished!"

Week-8/Function-Stack/Main.java

# Function Stack

```
public class Main {  
    public static void method2(){  
        System.out.println("Method 2 was called");  
    }  
  
    public static void method1(){  
        System.out.println("Method 1 was called");  
        method2();  
        System.out.println("Method 1 still alive");  
    }  
  
    public static void main(String[] args) {  
        method2();  
        method1();  
        System.out.println("Finished!");  
    }  
}
```

method2()
1. Print "Method 2 was called"
main()
1. Call method2()
2. Call method1()
3. Print "Finished!"

Week-8/Function-Stack/Main.java

# Function Stack

```
public class Main {  
    public static void method2(){  
        System.out.println("Method 2 was called");  
    }  
  
    public static void method1(){  
        System.out.println("Method 1 was called");  
        method2();  
        System.out.println("Method 1 still alive");  
    }  
  
    public static void main(String[] args) {  
        method2();  
        method1();  
        System.out.println("Finished!");  
    }  
}
```

Method 2 was called

method2()

1. Print "Method 2 was called"

main()

1. Call method2()
2. Call method1()
3. Print "Finished!"

Week-8/Function-Stack/Main.java



# Function Stack

```
public class Main {  
    public static void method2(){  
        System.out.println("Method 2 was called");  
    }  
  
    public static void method1(){  
        System.out.println("Method 1 was called");  
        method2();  
        System.out.println("Method 1 still alive");  
    }  
  
    public static void main(String[] args) {  
        method2();  
        method1();  
        System.out.println("Finished!");  
    }  
}
```

Method 2 was called

main()  
1. Call method1()  
2. Print "Finished!"

Week-8/Function-Stack/Main.java

# Function Stack

```
public class Main {  
    public static void method2(){  
        System.out.println("Method 2 was called");  
    }  
  
    public static void method1(){  
        System.out.println("Method 1 was called");  
        method2();  
        System.out.println("Method 1 still alive");  
    }  
  
    public static void main(String[] args) {  
        method2();  
        method1();  
        System.out.println("Finished!");  
    }  
}
```

Method 2 was called

method1()

1. Print "Method 1 was called"
2. Call method2()
3. Print "Method 1 still alive"

main()

1. Call method1()
2. Print "Finished!"

Week-8/Function-Stack/Main.java



# Function Stack

```
public class Main {  
    public static void method2(){  
        System.out.println("Method 2 was called");  
    }  
  
    public static void method1(){  
        System.out.println("Method 1 was called");  
        method2();  
        System.out.println("Method 1 still alive");  
    }  
  
    public static void main(String[] args) {  
        method2();  
        method1();  
        System.out.println("Finished!");  
    }  
}
```

Method 2 was called

method1()

1. Print "Method 1 was called"
2. Call method2()
3. Print "Method 1 still alive"

main()

1. Call method1()
2. Print "Finished!"

Week-8/Function-Stack/Main.java



# Function Stack

```
public class Main {  
    public static void method2(){  
        System.out.println("Method 2 was called");  
    }  
  
    public static void method1(){  
        System.out.println("Method 1 was called");  
        method2();  
        System.out.println("Method 1 still alive");  
    }  
  
    public static void main(String[] args) {  
        method2();  
        method1();  
        System.out.println("Finished!");  
    }  
}
```

Method 2 was called  
Method 1 was called

method1() 1. Print "Method 1 was called" 2. Call method2() 3. Print "Method 1 still alive"
main() 1. Call method1() 2. Print "Finished!"

Week-8/Function-Stack/Main.java



# Function Stack

```
public class Main {  
    public static void method2(){  
        System.out.println("Method 2 was called");  
    }  
  
    public static void method1(){  
        System.out.println("Method 1 was called");  
        method2();  
        System.out.println("Method 1 still alive");  
    }  
  
    public static void main(String[] args) {  
        method2();  
        method1();  
        System.out.println("Finished!");  
    }  
}
```

Method 2 was called  
Method 1 was called

method2() 1. Print "Method 2 was called"
method1() 1. Call method2() 2. Print "Method 1 still alive"
main() 1. Call method1() 2. Print "Finished!"

Week-8/Function-Stack/Main.java



# Function Stack

```
public class Main {  
    public static void method2(){  
        System.out.println("Method 2 was called");  
    }  
  
    public static void method1(){  
        System.out.println("Method 1 was called");  
        method2();  
        System.out.println("Method 1 still alive");  
    }  
  
    public static void main(String[] args) {  
        method2();  
        method1();  
        System.out.println("Finished!");  
    }  
}
```

Method 2 was called  
Method 1 was called

method2() 1. Print "Method 2 was called"
method1() 1. Call method2() 2. Print "Method 1 still alive"
main() 1. Call method1() 2. Print "Finished!"

Week-8/Function-Stack/Main.java

# Function Stack

```
public class Main {  
    public static void method2(){  
        System.out.println("Method 2 was called");  
    }  
  
    public static void method1(){  
        System.out.println("Method 1 was called");  
        method2();  
        System.out.println("Method 1 still alive");  
    }  
  
    public static void main(String[] args) {  
        method2();  
        method1();  
        System.out.println("Finished!");  
    }  
}
```

Method 2 was called  
Method 1 was called  
Method 2 was called

method2() 1. Print "Method 2 was called"
method1() 1. Call method2() 2. Print "Method 1 still alive"
main() 1. Call method1() 2. Print "Finished!"

Week-8/Function-Stack/Main.java

# Function Stack

```
public class Main {  
    public static void method2(){  
        System.out.println("Method 2 was called");  
    }  
  
    public static void method1(){  
        System.out.println("Method 1 was called");  
        method2();  
        System.out.println("Method 1 still alive");  
    }  
  
    public static void main(String[] args) {  
        method2();  
        method1();  
        System.out.println("Finished!");  
    }  
}
```

Method 2 was called  
Method 1 was called  
Method 2 was called  
Method 1 still alive

method1() 1. Print "Method 1 still alive"
main() 1. Call method1() 2. Print "Finished!"

Week-8/Function-Stack/Main.java



**KENNESAW STATE**  
UNIVERSITY

# Function Stack

```
public class Main {  
    public static void method2(){  
        System.out.println("Method 2 was called");  
    }  
  
    public static void method1(){  
        System.out.println("Method 1 was called");  
        method2();  
        System.out.println("Method 1 still alive");  
    }  
  
    public static void main(String[] args) {  
        method2();  
        method1();  
        System.out.println("Finished!");  
    }  
}
```

```
Method 2 was called  
Method 1 was called  
Method 2 was called  
Method 1 still alive  
Finished!
```

```
main()  
1. Print "Finished!"
```

*Week-8/Function-Stack/Main.java*

# Function Stack

```
public class Main {  
    public static void method2(){  
        System.out.println("Method 2 was called");  
    }  
  
    public static void method1(){  
        System.out.println("Method 1 was called");  
        method2();  
        System.out.println("Method 1 still alive");  
    }  
  
    public static void main(String[] args) {  
        method2();  
        method1();  
        System.out.println("Finished!");  
    }  
}
```

```
Method 2 was called  
Method 1 was called  
Method 2 was called  
Method 1 still alive  
Finished!
```

**Program terminated**

*Week-8/Function-Stack/Main.java*



# Repetition Structures

- Loops/Iterative:
  - Also called Explicit Repetition.
  - State is maintained by value.
  - Examples:
    - For Loop
    - Foreach Loop
    - While Loop
    - Do While Loop

# Repetition Structures

- Recursion or Implicit Repetition:
  - A function calling itself.
  - State is maintained in the call stack.

# Recursion

- Recursion is a programming concept that involves a function calling itself.
- Recursion is used to reduce complex problems into smaller subproblems.
- Code-wise, recursion solutions looks simpler and shorter than its iterative counterpart.



# Recursion

- A recursive method consists of two essential components:
- Base Case – We define the stopping point of the recursion.
- Recursive Call – We define the part where the function calls itself.

# In-Class Example 1

- Create a Recursive Method that prints a counter.
- It should take in one integer value, which is the starting number for the counter.
- The counter should count down to 0.

# Function Stack

```
public class Counter {  
    public static void counter(int number){  
        // Base case: Since we count from number to  
        // 0, we should stop at -1  
        if(number < 0){  
            return;  
        }  
        System.out.println("Counter: " + number);  
  
        // Recursive Call: Since we count from number  
        // to 0, each call should decrease number by 1  
        counter(number - 1);  
    }  
  
    public static void main(String[] args) {  
        counter(5);  
    }  
}
```

Week-8/Function-Stack/Counter.java



# Function Stack

```
public class Counter {  
    public static void counter(int number){  
        // Base case: Since we count from number to  
        // 0, we should stop at -1  
        if(number < 0){  
            return;  
        }  
        System.out.println("Counter: " + number);  
  
        // Recursive Call: Since we count from number  
        // to 0, each call should decrease number by 1  
        counter(number - 1);  
    }  
  
    public static void main(String[] args) {  
        counter(5);  
    }  
}
```

```
main()  
1. Call counter(5)
```

Week-8/Function-Stack/Counter.java



# Function Stack

```
public class Counter {  
    public static void counter(int number){  
        // Base case: Since we count from number to  
        // 0, we should stop at -1  
        if(number < 0){  
            return;  
        }  
        System.out.println("Counter: " + number);  
  
        // Recursive Call: Since we count from number  
        // to 0, each call should decrease number by 1  
        counter(number - 1);  
    }  
  
    public static void main(String[] args) {  
        counter(5);  
    }  
}
```

```
main()  
1. Call counter(5)
```

Week-8/Function-Stack/Counter.java



# Function Stack

```
public class Counter {  
    public static void counter(int number){// counter(5)  
        // Base case: Since we count from number to  
        // 0, we should stop at -1  
        if(number < 0){  
            return;  
        }  
        System.out.println("Counter: " + number);  
  
        // Recursive Call: Since we count from number  
        // to 0, each call should decrease number by 1  
        counter(number - 1);  
    }  
  
    public static void main(String[] args) {  
        counter(5);  
    }  
}
```

counter(5) 1. Is 5 < 0? 2. Print "Counter: 5" 3. Call counter(4)
main() 1. Call counter(5)

Week-8/Function-Stack/Counter.java

# Function Stack

```
public class Counter {  
    public static void counter(int number){// counter(5)  
        // Base case: Since we count from number to  
        // 0, we should stop at -1  
        if(number < 0){  
            return;  
        }  
        System.out.println("Counter: " + number);  
  
        // Recursive Call: Since we count from number  
        // to 0, each call should decrease number by 1  
        counter(number - 1);  
    }  
  
    public static void main(String[] args) {  
        counter(5);  
    }  
}
```

counter(5) 1. Is 5 < 0? 2. Print "Counter: 5" 3. Call counter(4)
main() 1. Call counter(5)

Week-8/Function-Stack/Counter.java

# Function Stack

```
public class Counter {  
    public static void counter(int number){// counter(5)  
        // Base case: Since we count from number to  
        // 0, we should stop at -1  
        if(number < 0){  
            return;  
        }  
        System.out.println("Counter: " + number);  
  
        // Recursive Call: Since we count from number  
        // to 0, each call should decrease number by 1  
        counter(number - 1);  
    }  
  
    public static void main(String[] args) {  
        counter(5);  
    }  
}
```

Counter: 5

counter(5)  
1. Print "Counter: 5"  
2. Call counter(4)

main()  
1. Call counter(5)

Week-8/Function-Stack/Counter.java





# Function Stack

```
public class Counter {  
    public static void counter(int number){// counter(5)  
        // Base case: Since we count from number to  
        // 0, we should stop at -1  
        if(number < 0){  
            return;  
        }  
        System.out.println("Counter: " + number);  
  
        // Recursive Call: Since we count from number  
        // to 0, each call should decrease number by 1  
        counter(number - 1);  
    }  
  
    public static void main(String[] args) {  
        counter(5);  
    }  
}
```

Counter: 5

counter(5) 1. Call counter(4)
main() 1. Call counter(5)

Week-8/Function-Stack/Counter.java

# Function Stack

```
public class Counter {  
    public static void counter(int number){// counter(4)  
        // Base case: Since we count from number to  
        // 0, we should stop at -1  
        if(number < 0){  
            return;  
        }  
        System.out.println("Counter: " + number);  
  
        // Recursive Call: Since we count from number  
        // to 0, each call should decrease number by 1  
        counter(number - 1);  
    }  
  
    public static void main(String[] args) {  
        counter(5);  
    }  
}
```

Counter: 5

counter(4)
1. Is 4 < 0?
2. Print "Counter: 4"
3. Call counter(3)
counter(5)
1. Call counter(4)
main()
1. Call counter(5)

Week-8/Function-Stack/Counter.java



# Function Stack

```
public class Counter {  
    public static void counter(int number){// counter(4)  
        // Base case: Since we count from number to  
        // 0, we should stop at -1  
        if(number < 0){  
            return;  
        }  
        System.out.println("Counter: " + number);  
  
        // Recursive Call: Since we count from number  
        // to 0, each call should decrease number by 1  
        counter(number - 1);  
    }  
  
    public static void main(String[] args) {  
        counter(5);  
    }  
}
```

Counter: 5

counter(4)
1. Is 4 < 0?
2. Print "Counter: 4"
3. Call counter(3)
counter(5)
1. Call counter(4)
main()
1. Call counter(5)

Week-8/Function-Stack/Counter.java



# Function Stack

```
public class Counter {  
    public static void counter(int number){// counter(4)  
        // Base case: Since we count from number to  
        // 0, we should stop at -1  
        if(number < 0){  
            return;  
        }  
        System.out.println("Counter: " + number);  
  
        // Recursive Call: Since we count from number  
        // to 0, each call should decrease number by 1  
        counter(number - 1);  
    }  
  
    public static void main(String[] args) {  
        counter(5);  
    }  
}
```

Counter: 5

Counter: 4

counter(4) 1. Print "Counter: 4" 2. Call counter(3)
counter(5) 1. Call counter(4)
main() 1. Call counter(5)

Week-8/Function-Stack/Counter.java



# Function Stack

```
public class Counter {  
    public static void counter(int number){// counter(4)  
        // Base case: Since we count from number to  
        // 0, we should stop at -1  
        if(number < 0){  
            return;  
        }  
        System.out.println("Counter: " + number);  
  
        // Recursive Call: Since we count from number  
        // to 0, each call should decrease number by 1  
        counter(number - 1);  
    }  
  
    public static void main(String[] args) {  
        counter(5);  
    }  
}
```

Counter: 5

Counter: 4

counter(4) 1. Call counter(3)
counter(5) 1. Call counter(4)
main() 1. Call counter(5)

Week-8/Function-Stack/Counter.java



# Function Stack

```
public class Counter {  
    public static void counter(int number){// counter(3)  
        // Base case: Since we count from number to  
        // 0, we should stop at -1  
        if(number < 0){  
            return;  
        }  
        System.out.println("Counter: " + number);  
  
        // Recursive Call: Since we count from number  
        // to 0, each call should decrease number by 1  
        counter(number - 1);  
    }  
  
    public static void main(String[] args) {  
        counter(5);  
    }  
}
```

Counter: 5

Counter: 4

Counter: 3

counter(3) 1. Call counter(2)
counter(4) 1. Call counter(3)
counter(5) 1. Call counter(4)
main() 1. Call counter(5)

Week-8/Function-Stack/Counter.java

# Function Stack

```
public class Counter {  
    public static void counter(int number){  
        // Base case: Since we count from number to  
        // 0, we should stop at -1  
        if(number < 0){  
            return;  
        }  
        System.out.println("Counter: " + number);  
  
        // Recursive Call: Since we count from number  
        // to 0, each call should decrease number by 1  
        counter(number - 1);  
    }  
  
    public static void main(String[] args) {  
        counter(5);  
    }  
}
```

Counter: 5  
Counter: 4  
Counter: 3  
Counter: 2

counter(2)
1. Call counter(1)
counter(3)
1. Call counter(2)
counter(4)
1. Call counter(3)
counter(5)
1. Call counter(4)
main()
1. Call counter(5)

Week-8/Function-Stack/Counter.java

# Function Stack

```
public class Counter {  
    public static void counter(int number){// counter(1)  
        // Base case: Since we count from number to  
        // 0, we should stop at -1  
        if(number < 0){  
            return;  
        }  
        System.out.println("Counter: " + number);  
  
        // Recursive Call: Since we count from number  
        // to 0, each call should decrease number by 1  
        counter(number - 1);  
    }  
  
    public static void main(String[] args) {  
        counter(5);  
    }  
}
```

Counter: 5  
Counter: 4  
Counter: 3  
Counter: 2  
Counter: 1

counter(1)
1. Call counter(0)
counter(2)
1. Call counter(1)
counter(3)
1. Call counter(2)
counter(4)
1. Call counter(3)
counter(5)
1. Call counter(4)
main()
1. Call counter(5)

Week-8/Function-Stack/Counter.java



# Function Stack

```
public class Counter {  
    public static void counter(int number){// counter(0)  
        // Base case: Since we count from number to  
        // 0, we should stop at -1  
        if(number < 0){  
            return;  
        }  
        System.out.println("Counter: " + number);  
  
        // Recursive Call: Since we count from number  
        // to 0, each call should decrease number by 1  
        counter(number - 1);  
    }  
  
    public static void main(String[] args) {  
        counter(5);  
    }  
}
```

Counter: 5  
Counter: 4  
Counter: 3  
Counter: 2  
Counter: 1

counter(0) 1. Is 0 < 0? 2. Print "Counter: 0" 3. Call counter(-1)
counter(1) 1. Call counter(0)
counter(2) 1. Call counter(1)
counter(3) 1. Call counter(2)
counter(4) 1. Call counter(3)
counter(5) 1. Call counter(4)
main() 1. Call counter(5)

Week-8/Function-Stack/Counter.java

# Function Stack

```
public class Counter {  
    public static void counter(int number){// counter(0)  
        // Base case: Since we count from number to  
        // 0, we should stop at -1  
        if(number < 0){  
            return;  
        }  
        System.out.println("Counter: " + number);  
  
        // Recursive Call: Since we count from number  
        // to 0, each call should decrease number by 1  
        counter(number - 1);  
    }  
  
    public static void main(String[] args) {  
        counter(5);  
    }  
}
```

Counter: 5  
Counter: 4  
Counter: 3  
Counter: 2  
Counter: 1

counter(0) 1. Is 0 < 0? 2. Print "Counter: 0" 3. Call counter(-1)
counter(1) 1. Call counter(0)
counter(2) 1. Call counter(1)
counter(3) 1. Call counter(2)
counter(4) 1. Call counter(3)
counter(5) 1. Call counter(4)
main() 1. Call counter(5)

Week-8/Function-Stack/Counter.java

# Function Stack

```
public class Counter {  
    public static void counter(int number){// counter(0)  
        // Base case: Since we count from number to  
        // 0, we should stop at -1  
        if(number < 0){  
            return;  
        }  
        System.out.println("Counter: " + number);  
  
        // Recursive Call: Since we count from number  
        // to 0, each call should decrease number by 1  
        counter(number - 1);  
    }  
  
    public static void main(String[] args) {  
        counter(5);  
    }  
}
```

Counter: 5  
Counter: 4  
Counter: 3  
Counter: 2  
Counter: 1  
Counter: 0

counter(0)  
1. Print "Counter: 0"  
2. Call counter(-1)

counter(1)  
1. Call counter(0)

counter(2)  
1. Call counter(1)

counter(3)  
1. Call counter(2)

counter(4)  
1. Call counter(3)

counter(5)  
1. Call counter(4)

main()  
1. Call counter(5)

Week-8/Function-Stack/Counter.java

# Function Stack

```
public class Counter {  
    public static void counter(int number){// counter(0)  
        // Base case: Since we count from number to  
        // 0, we should stop at -1  
        if(number < 0){  
            return;  
        }  
        System.out.println("Counter: " + number);  
  
        // Recursive Call: Since we count from number  
        // to 0, each call should decrease number by 1  
        counter(number - 1);  
    }  
  
    public static void main(String[] args) {  
        counter(5);  
    }  
}
```

Counter: 5  
Counter: 4  
Counter: 3  
Counter: 2  
Counter: 1  
Counter: 0

counter(0) 1. Call counter(-1)
counter(1) 1. Call counter(0)
counter(2) 1. Call counter(1)
counter(3) 1. Call counter(2)
counter(4) 1. Call counter(3)
counter(5) 1. Call counter(4)
main() 1. Call counter(5)

Week-8/Function-Stack/Counter.java

# Function Stack

```
public class Counter {  
    public static void counter(int number){// counter(-1)  
        // Base case: Since we count from number to  
        // 0, we should stop at -1  
        if(number < 0){  
            return;  
        }  
        System.out.println("Counter: " + number);  
  
        // Recursive Call: Since we count from number  
        // to 0, each call should decrease number by 1  
        counter(number - 1);  
    }  
  
    public static void main(String[] args) {  
        counter(5);  
    }  
}
```

Counter: 5  
Counter: 4  
Counter: 3  
Counter: 2  
Counter: 1  
Counter: 0

counter(-1)  
1. Is -1 < 0?  
2. Print "Counter: -1"  
3. Call counter(-2)

counter(0)  
1. Call counter(-1)

counter(1)  
1. Call counter(0)

counter(2)  
1. Call counter(1)

counter(3)  
1. Call counter(2)

counter(4)  
1. Call counter(3)

counter(5)  
1. Call counter(4)

main()  
1. Call counter(5)

Week-8/Function-Stack/Counter.java



# Function Stack

```
public class Counter {  
    public static void counter(int number){// counter(-1)  
        // Base case: Since we count from number to  
        // 0, we should stop at -1  
        if(number < 0){  
            return;  
        }  
        System.out.println("Counter: " + number);  
  
        // Recursive Call: Since we count from number  
        // to 0, each call should decrease number by 1  
        counter(number - 1);  
    }  
  
    public static void main(String[] args) {  
        counter(5);  
    }  
}
```

Counter: 5  
Counter: 4  
Counter: 3  
Counter: 2  
Counter: 1  
Counter: 0

counter(-1)  
1. Is -1 < 0?  
2. Print "Counter: -1"  
3. Call counter(-2)

counter(0)  
1. Call counter(-1)

counter(1)  
1. Call counter(0)

counter(2)  
1. Call counter(1)

counter(3)  
1. Call counter(2)

counter(4)  
1. Call counter(3)

counter(5)  
1. Call counter(4)

main()  
1. Call counter(5)

Week-8/Function-Stack/Counter.java



# Function Stack

```
public class Counter {  
    public static void counter(int number){// counter(-1)  
        // Base case: Since we count from number to  
        // 0, we should stop at -1  
        if(number < 0){  
            return;  
        }  
        System.out.println("Counter: " + number);  
  
        // Recursive Call: Since we count from number  
        // to 0, each call should decrease number by 1  
        counter(number - 1);  
    }  
  
    public static void main(String[] args) {  
        counter(5);  
    }  
}
```

Counter: 5  
Counter: 4  
Counter: 3  
Counter: 2  
Counter: 1  
Counter: 0

counter(-1)  
1. Is -1 < 0?  
**2. Print "Counter: -1"**  
**3. Call counter(-2)**

counter(0)  
1. Call counter(-1)

counter(1)  
1. Call counter(0)

counter(2)  
1. Call counter(1)

counter(3)  
1. Call counter(2)

counter(4)  
1. Call counter(3)

counter(5)  
1. Call counter(4)

main()  
1. Call counter(5)

Week-8/Function-Stack/Counter.java

# Function Stack

```
public class Counter {  
    public static void counter(int number){// counter(-1)  
        // Base case: Since we count from number to  
        // 0, we should stop at -1  
        if(number < 0){  
            return;  
        }  
        System.out.println("Counter: " + number);  
  
        // Recursive Call: Since we count from number  
        // to 0, each call should decrease number by 1  
        counter(number - 1);  
    }  
  
    public static void main(String[] args) {  
        counter(5);  
    }  
}
```

Counter: 5  
Counter: 4  
Counter: 3  
Counter: 2  
Counter: 1  
Counter: 0

counter(-1)

1. Is -1 < 0?
2. return

counter(0)

1. Call counter(-1)

counter(1)

1. Call counter(0)

counter(2)

1. Call counter(1)

counter(3)

1. Call counter(2)

counter(4)

1. Call counter(3)

counter(5)

1. Call counter(4)

main()

1. Call counter(5)

Week-8/Function-Stack/Counter.java





# Function Stack

```
public class Counter {  
    public static void counter(int number){// counter(0)  
        // Base case: Since we count from number to  
        // 0, we should stop at -1  
        if(number < 0){  
            return;  
        }  
        System.out.println("Counter: " + number);  
  
        // Recursive Call: Since we count from number  
        // to 0, each call should decrease number by 1  
        counter(number - 1);  
    }  
  
    public static void main(String[] args) {  
        counter(5);  
    }  
}
```

Counter: 5  
Counter: 4  
Counter: 3  
Counter: 2  
Counter: 1  
Counter: 0

counter(0) 1. Call counter(-1)
counter(1) 1. Call counter(0)
counter(2) 1. Call counter(1)
counter(3) 1. Call counter(2)
counter(4) 1. Call counter(3)
counter(5) 1. Call counter(4)
main() 1. Call counter(5)

Week-8/Function-Stack/Counter.java



**KENNESAW STATE**  
UNIVERSITY

# Function Stack

```
public class Counter {  
    public static void counter(int number){// counter(1)  
        // Base case: Since we count from number to  
        // 0, we should stop at -1  
        if(number < 0){  
            return;  
        }  
        System.out.println("Counter: " + number);  
  
        // Recursive Call: Since we count from number  
        // to 0, each call should decrease number by 1  
        counter(number - 1);  
    }  
  
    public static void main(String[] args) {  
        counter(5);  
    }  
}
```

Counter: 5  
Counter: 4  
Counter: 3  
Counter: 2  
Counter: 1  
Counter: 0

counter(1) 1. Call counter(0)
counter(2) 1. Call counter(1)
counter(3) 1. Call counter(2)
counter(4) 1. Call counter(3)
counter(5) 1. Call counter(4)
main() 1. Call counter(5)

Week-8/Function-Stack/Counter.java



**KENNESAW STATE**  
UNIVERSITY

# Function Stack

```
public class Counter {  
    public static void counter(int number){  
        // Base case: Since we count from number to  
        // 0, we should stop at -1  
        if(number < 0){  
            return;  
        }  
        System.out.println("Counter: " + number);  
  
        // Recursive Call: Since we count from number  
        // to 0, each call should decrease number by 1  
        counter(number - 1);  
    }  
  
    public static void main(String[] args) {  
        counter(5);  
    }  
}
```

Counter: 5  
Counter: 4  
Counter: 3  
Counter: 2  
Counter: 1  
Counter: 0

counter(2)
1. Call counter(1)
counter(3)
1. Call counter(2)
counter(4)
1. Call counter(3)
counter(5)
1. Call counter(4)
main()
1. Call counter(5)

Week-8/Function-Stack/Counter.java

# Function Stack

```
public class Counter {  
    public static void counter(int number){// counter(3)  
        // Base case: Since we count from number to  
        // 0, we should stop at -1  
        if(number < 0){  
            return;  
        }  
        System.out.println("Counter: " + number);  
  
        // Recursive Call: Since we count from number  
        // to 0, each call should decrease number by 1  
        counter(number - 1);  
    }  
  
    public static void main(String[] args) {  
        counter(5);  
    }  
}
```

Counter: 5  
Counter: 4  
Counter: 3  
Counter: 2  
Counter: 1  
Counter: 0

counter(3)
1. Call counter(2)
counter(4)
1. Call counter(3)
counter(5)
1. Call counter(4)
main()
1. Call counter(5)

Week-8/Function-Stack/Counter.java

# Function Stack

```
public class Counter {  
    public static void counter(int number){// counter(4)  
        // Base case: Since we count from number to  
        // 0, we should stop at -1  
        if(number < 0){  
            return;  
        }  
        System.out.println("Counter: " + number);  
  
        // Recursive Call: Since we count from number  
        // to 0, each call should decrease number by 1  
        counter(number - 1);  
    }  
  
    public static void main(String[] args) {  
        counter(5);  
    }  
}
```

Counter: 5  
Counter: 4  
Counter: 3  
Counter: 2  
Counter: 1  
Counter: 0

counter(4) 1. Call counter(3)
counter(5) 1. Call counter(4)
main() 1. Call counter(5)

Week-8/Function-Stack/Counter.java

# Function Stack

```
public class Counter {  
    public static void counter(int number){// counter(5)  
        // Base case: Since we count from number to  
        // 0, we should stop at -1  
        if(number < 0){  
            return;  
        }  
        System.out.println("Counter: " + number);  
  
        // Recursive Call: Since we count from number  
        // to 0, each call should decrease number by 1  
        counter(number - 1);  
    }  
  
    public static void main(String[] args) {  
        counter(5);  
    }  
}
```

Counter: 5  
Counter: 4  
Counter: 3  
Counter: 2  
Counter: 1  
Counter: 0

counter(5) 1. Call counter(4)
main() 1. Call counter(5)

Week-8/Function-Stack/Counter.java

# Function Stack

```
public class Counter {  
    public static void counter(int number){  
        // Base case: Since we count from number to  
        // 0, we should stop at -1  
        if(number < 0){  
            return;  
        }  
        System.out.println("Counter: " + number);  
  
        // Recursive Call: Since we count from number  
        // to 0, each call should decrease number by 1  
        counter(number - 1);  
    }  
  
    public static void main(String[] args) {  
        counter(5);  
    }  
}
```

```
Counter: 5  
Counter: 4  
Counter: 3  
Counter: 2  
Counter: 1  
Counter: 0
```

```
main()  
1. Call counter(5)
```

Week-8/Function-Stack/Counter.java



# Function Stack

```
public class Counter {  
    public static void counter(int number){  
        // Base case: Since we count from number to  
        // 0, we should stop at -1  
        if(number < 0){  
            return;  
        }  
        System.out.println("Counter: " + number);  
  
        // Recursive Call: Since we count from number  
        // to 0, each call should decrease number by 1  
        counter(number - 1);  
    }  
  
    public static void main(String[] args) {  
        counter(5);  
    }  
}
```

```
Counter: 5  
Counter: 4  
Counter: 3  
Counter: 2  
Counter: 1  
Counter: 0
```

```
main()
```

Week-8/Function-Stack/Counter.java





# Function Stack

```
public class Counter {  
    public static void counter(int number){  
        // Base case: Since we count from number to  
        // 0, we should stop at -1  
        if(number < 0){  
            return;  
        }  
        System.out.println("Counter: " + number);  
  
        // Recursive Call: Since we count from number  
        // to 0, each call should decrease number by 1  
        counter(number - 1);  
    }  
  
    public static void main(String[] args) {  
        counter(5);  
    }  
}
```

```
Counter: 5  
Counter: 4  
Counter: 3  
Counter: 2  
Counter: 1  
Counter: 0
```

**Program terminated**

Week-8/Function-Stack/Counter.java



## In-Class Example 2

- Let's try the opposite, the method should now count up, starting from 0 until we reach the input number.