**Revision Notes: Functions in Java**

**Table of Contents**

1. [Introduction to Functions](https://companion.scaler.com/meeting/class/858011/notes?_gl=1*2m1h9n*_gcl_gb*R0NMLjE3NTYzOTMzMzIuQ2xvS0NRancyYnJGQmhENUFSSkpBTGpyRC1FZkdFTFBVT1VRX1dCcF8tNmY1bWVrYnByRkNmMVpnREtiS2t5cEZGYThTWi12Mkt4M3pHa1JCWVVxdFB6R3RncDQ5UDFHTHp5NHR4SmJYbmhCOE5Vc2VqaklhQm9Dd1NBLkNOZTZDS2JGdk9ZQkVMeV8wNkVELmlZc2xDT0NrX01rQ0VMeV8wNkVELkJaSUVDTVNhMFprWkVMeV8wNkVELnpnZDFDSkd4N2ZZYUVMeV8wNkVELm1jS05DUHZDajRzWUVLRHo1YVlwLmFqVVJDT1duMXZZYUVLRHo1YVlwLjFrVTVDUEdUcUlVWkVJRHFrOW9xLjNKZGVDSVBFMV9ZYUVJRHFrOW9xLmF6ZTZDT3pIMWZzWUVKbVVqOG9x*_gcl_au*NjA1MTY2NDYxLjE3NTQyNDU4MzUuMTgwNjA3NzQ0Mi4xNzU2MzkzMzI4LjE3NTYzOTMzMzI.*FPAU*NjE4OTk4MTc5LjE3NTQyNDU3MzE.*_ga*MTEwOTQzNzc1OC4xNjkxODM5NDAy*_ga_53S71ZZG1X*czE3NTc4Mzc0ODAkbzM5MSRnMSR0MTc1Nzg0NDg4NiRqMTgkbDAkaDUyMjU4MjIyNw..#introduction-to-functions)
2. [Function Definitions](https://companion.scaler.com/meeting/class/858011/notes?_gl=1*2m1h9n*_gcl_gb*R0NMLjE3NTYzOTMzMzIuQ2xvS0NRancyYnJGQmhENUFSSkpBTGpyRC1FZkdFTFBVT1VRX1dCcF8tNmY1bWVrYnByRkNmMVpnREtiS2t5cEZGYThTWi12Mkt4M3pHa1JCWVVxdFB6R3RncDQ5UDFHTHp5NHR4SmJYbmhCOE5Vc2VqaklhQm9Dd1NBLkNOZTZDS2JGdk9ZQkVMeV8wNkVELmlZc2xDT0NrX01rQ0VMeV8wNkVELkJaSUVDTVNhMFprWkVMeV8wNkVELnpnZDFDSkd4N2ZZYUVMeV8wNkVELm1jS05DUHZDajRzWUVLRHo1YVlwLmFqVVJDT1duMXZZYUVLRHo1YVlwLjFrVTVDUEdUcUlVWkVJRHFrOW9xLjNKZGVDSVBFMV9ZYUVJRHFrOW9xLmF6ZTZDT3pIMWZzWUVKbVVqOG9x*_gcl_au*NjA1MTY2NDYxLjE3NTQyNDU4MzUuMTgwNjA3NzQ0Mi4xNzU2MzkzMzI4LjE3NTYzOTMzMzI.*FPAU*NjE4OTk4MTc5LjE3NTQyNDU3MzE.*_ga*MTEwOTQzNzc1OC4xNjkxODM5NDAy*_ga_53S71ZZG1X*czE3NTc4Mzc0ODAkbzM5MSRnMSR0MTc1Nzg0NDg4NiRqMTgkbDAkaDUyMjU4MjIyNw..#function-definitions)
3. [Function Return Types](https://companion.scaler.com/meeting/class/858011/notes?_gl=1*2m1h9n*_gcl_gb*R0NMLjE3NTYzOTMzMzIuQ2xvS0NRancyYnJGQmhENUFSSkpBTGpyRC1FZkdFTFBVT1VRX1dCcF8tNmY1bWVrYnByRkNmMVpnREtiS2t5cEZGYThTWi12Mkt4M3pHa1JCWVVxdFB6R3RncDQ5UDFHTHp5NHR4SmJYbmhCOE5Vc2VqaklhQm9Dd1NBLkNOZTZDS2JGdk9ZQkVMeV8wNkVELmlZc2xDT0NrX01rQ0VMeV8wNkVELkJaSUVDTVNhMFprWkVMeV8wNkVELnpnZDFDSkd4N2ZZYUVMeV8wNkVELm1jS05DUHZDajRzWUVLRHo1YVlwLmFqVVJDT1duMXZZYUVLRHo1YVlwLjFrVTVDUEdUcUlVWkVJRHFrOW9xLjNKZGVDSVBFMV9ZYUVJRHFrOW9xLmF6ZTZDT3pIMWZzWUVKbVVqOG9x*_gcl_au*NjA1MTY2NDYxLjE3NTQyNDU4MzUuMTgwNjA3NzQ0Mi4xNzU2MzkzMzI4LjE3NTYzOTMzMzI.*FPAU*NjE4OTk4MTc5LjE3NTQyNDU3MzE.*_ga*MTEwOTQzNzc1OC4xNjkxODM5NDAy*_ga_53S71ZZG1X*czE3NTc4Mzc0ODAkbzM5MSRnMSR0MTc1Nzg0NDg4NiRqMTgkbDAkaDUyMjU4MjIyNw..#function-return-types)
4. [Calling Functions](https://companion.scaler.com/meeting/class/858011/notes?_gl=1*2m1h9n*_gcl_gb*R0NMLjE3NTYzOTMzMzIuQ2xvS0NRancyYnJGQmhENUFSSkpBTGpyRC1FZkdFTFBVT1VRX1dCcF8tNmY1bWVrYnByRkNmMVpnREtiS2t5cEZGYThTWi12Mkt4M3pHa1JCWVVxdFB6R3RncDQ5UDFHTHp5NHR4SmJYbmhCOE5Vc2VqaklhQm9Dd1NBLkNOZTZDS2JGdk9ZQkVMeV8wNkVELmlZc2xDT0NrX01rQ0VMeV8wNkVELkJaSUVDTVNhMFprWkVMeV8wNkVELnpnZDFDSkd4N2ZZYUVMeV8wNkVELm1jS05DUHZDajRzWUVLRHo1YVlwLmFqVVJDT1duMXZZYUVLRHo1YVlwLjFrVTVDUEdUcUlVWkVJRHFrOW9xLjNKZGVDSVBFMV9ZYUVJRHFrOW9xLmF6ZTZDT3pIMWZzWUVKbVVqOG9x*_gcl_au*NjA1MTY2NDYxLjE3NTQyNDU4MzUuMTgwNjA3NzQ0Mi4xNzU2MzkzMzI4LjE3NTYzOTMzMzI.*FPAU*NjE4OTk4MTc5LjE3NTQyNDU3MzE.*_ga*MTEwOTQzNzc1OC4xNjkxODM5NDAy*_ga_53S71ZZG1X*czE3NTc4Mzc0ODAkbzM5MSRnMSR0MTc1Nzg0NDg4NiRqMTgkbDAkaDUyMjU4MjIyNw..#calling-functions)
5. [Practical Examples](https://companion.scaler.com/meeting/class/858011/notes?_gl=1*2m1h9n*_gcl_gb*R0NMLjE3NTYzOTMzMzIuQ2xvS0NRancyYnJGQmhENUFSSkpBTGpyRC1FZkdFTFBVT1VRX1dCcF8tNmY1bWVrYnByRkNmMVpnREtiS2t5cEZGYThTWi12Mkt4M3pHa1JCWVVxdFB6R3RncDQ5UDFHTHp5NHR4SmJYbmhCOE5Vc2VqaklhQm9Dd1NBLkNOZTZDS2JGdk9ZQkVMeV8wNkVELmlZc2xDT0NrX01rQ0VMeV8wNkVELkJaSUVDTVNhMFprWkVMeV8wNkVELnpnZDFDSkd4N2ZZYUVMeV8wNkVELm1jS05DUHZDajRzWUVLRHo1YVlwLmFqVVJDT1duMXZZYUVLRHo1YVlwLjFrVTVDUEdUcUlVWkVJRHFrOW9xLjNKZGVDSVBFMV9ZYUVJRHFrOW9xLmF6ZTZDT3pIMWZzWUVKbVVqOG9x*_gcl_au*NjA1MTY2NDYxLjE3NTQyNDU4MzUuMTgwNjA3NzQ0Mi4xNzU2MzkzMzI4LjE3NTYzOTMzMzI.*FPAU*NjE4OTk4MTc5LjE3NTQyNDU3MzE.*_ga*MTEwOTQzNzc1OC4xNjkxODM5NDAy*_ga_53S71ZZG1X*czE3NTc4Mzc0ODAkbzM5MSRnMSR0MTc1Nzg0NDg4NiRqMTgkbDAkaDUyMjU4MjIyNw..#practical-examples)
   * [Count Factors of a Number](https://companion.scaler.com/meeting/class/858011/notes?_gl=1*2m1h9n*_gcl_gb*R0NMLjE3NTYzOTMzMzIuQ2xvS0NRancyYnJGQmhENUFSSkpBTGpyRC1FZkdFTFBVT1VRX1dCcF8tNmY1bWVrYnByRkNmMVpnREtiS2t5cEZGYThTWi12Mkt4M3pHa1JCWVVxdFB6R3RncDQ5UDFHTHp5NHR4SmJYbmhCOE5Vc2VqaklhQm9Dd1NBLkNOZTZDS2JGdk9ZQkVMeV8wNkVELmlZc2xDT0NrX01rQ0VMeV8wNkVELkJaSUVDTVNhMFprWkVMeV8wNkVELnpnZDFDSkd4N2ZZYUVMeV8wNkVELm1jS05DUHZDajRzWUVLRHo1YVlwLmFqVVJDT1duMXZZYUVLRHo1YVlwLjFrVTVDUEdUcUlVWkVJRHFrOW9xLjNKZGVDSVBFMV9ZYUVJRHFrOW9xLmF6ZTZDT3pIMWZzWUVKbVVqOG9x*_gcl_au*NjA1MTY2NDYxLjE3NTQyNDU4MzUuMTgwNjA3NzQ0Mi4xNzU2MzkzMzI4LjE3NTYzOTMzMzI.*FPAU*NjE4OTk4MTc5LjE3NTQyNDU3MzE.*_ga*MTEwOTQzNzc1OC4xNjkxODM5NDAy*_ga_53S71ZZG1X*czE3NTc4Mzc0ODAkbzM5MSRnMSR0MTc1Nzg0NDg4NiRqMTgkbDAkaDUyMjU4MjIyNw..#count-factors-of-a-number)
   * [Sum of Digits](https://companion.scaler.com/meeting/class/858011/notes?_gl=1*2m1h9n*_gcl_gb*R0NMLjE3NTYzOTMzMzIuQ2xvS0NRancyYnJGQmhENUFSSkpBTGpyRC1FZkdFTFBVT1VRX1dCcF8tNmY1bWVrYnByRkNmMVpnREtiS2t5cEZGYThTWi12Mkt4M3pHa1JCWVVxdFB6R3RncDQ5UDFHTHp5NHR4SmJYbmhCOE5Vc2VqaklhQm9Dd1NBLkNOZTZDS2JGdk9ZQkVMeV8wNkVELmlZc2xDT0NrX01rQ0VMeV8wNkVELkJaSUVDTVNhMFprWkVMeV8wNkVELnpnZDFDSkd4N2ZZYUVMeV8wNkVELm1jS05DUHZDajRzWUVLRHo1YVlwLmFqVVJDT1duMXZZYUVLRHo1YVlwLjFrVTVDUEdUcUlVWkVJRHFrOW9xLjNKZGVDSVBFMV9ZYUVJRHFrOW9xLmF6ZTZDT3pIMWZzWUVKbVVqOG9x*_gcl_au*NjA1MTY2NDYxLjE3NTQyNDU4MzUuMTgwNjA3NzQ0Mi4xNzU2MzkzMzI4LjE3NTYzOTMzMzI.*FPAU*NjE4OTk4MTc5LjE3NTQyNDU3MzE.*_ga*MTEwOTQzNzc1OC4xNjkxODM5NDAy*_ga_53S71ZZG1X*czE3NTc4Mzc0ODAkbzM5MSRnMSR0MTc1Nzg0NDg4NiRqMTgkbDAkaDUyMjU4MjIyNw..#sum-of-digits)
   * [Check Even or Odd](https://companion.scaler.com/meeting/class/858011/notes?_gl=1*2m1h9n*_gcl_gb*R0NMLjE3NTYzOTMzMzIuQ2xvS0NRancyYnJGQmhENUFSSkpBTGpyRC1FZkdFTFBVT1VRX1dCcF8tNmY1bWVrYnByRkNmMVpnREtiS2t5cEZGYThTWi12Mkt4M3pHa1JCWVVxdFB6R3RncDQ5UDFHTHp5NHR4SmJYbmhCOE5Vc2VqaklhQm9Dd1NBLkNOZTZDS2JGdk9ZQkVMeV8wNkVELmlZc2xDT0NrX01rQ0VMeV8wNkVELkJaSUVDTVNhMFprWkVMeV8wNkVELnpnZDFDSkd4N2ZZYUVMeV8wNkVELm1jS05DUHZDajRzWUVLRHo1YVlwLmFqVVJDT1duMXZZYUVLRHo1YVlwLjFrVTVDUEdUcUlVWkVJRHFrOW9xLjNKZGVDSVBFMV9ZYUVJRHFrOW9xLmF6ZTZDT3pIMWZzWUVKbVVqOG9x*_gcl_au*NjA1MTY2NDYxLjE3NTQyNDU4MzUuMTgwNjA3NzQ0Mi4xNzU2MzkzMzI4LjE3NTYzOTMzMzI.*FPAU*NjE4OTk4MTc5LjE3NTQyNDU3MzE.*_ga*MTEwOTQzNzc1OC4xNjkxODM5NDAy*_ga_53S71ZZG1X*czE3NTc4Mzc0ODAkbzM5MSRnMSR0MTc1Nzg0NDg4NiRqMTgkbDAkaDUyMjU4MjIyNw..#check-even-or-odd)
   * [Magnitude of a Number](https://companion.scaler.com/meeting/class/858011/notes?_gl=1*2m1h9n*_gcl_gb*R0NMLjE3NTYzOTMzMzIuQ2xvS0NRancyYnJGQmhENUFSSkpBTGpyRC1FZkdFTFBVT1VRX1dCcF8tNmY1bWVrYnByRkNmMVpnREtiS2t5cEZGYThTWi12Mkt4M3pHa1JCWVVxdFB6R3RncDQ5UDFHTHp5NHR4SmJYbmhCOE5Vc2VqaklhQm9Dd1NBLkNOZTZDS2JGdk9ZQkVMeV8wNkVELmlZc2xDT0NrX01rQ0VMeV8wNkVELkJaSUVDTVNhMFprWkVMeV8wNkVELnpnZDFDSkd4N2ZZYUVMeV8wNkVELm1jS05DUHZDajRzWUVLRHo1YVlwLmFqVVJDT1duMXZZYUVLRHo1YVlwLjFrVTVDUEdUcUlVWkVJRHFrOW9xLjNKZGVDSVBFMV9ZYUVJRHFrOW9xLmF6ZTZDT3pIMWZzWUVKbVVqOG9x*_gcl_au*NjA1MTY2NDYxLjE3NTQyNDU4MzUuMTgwNjA3NzQ0Mi4xNzU2MzkzMzI4LjE3NTYzOTMzMzI.*FPAU*NjE4OTk4MTc5LjE3NTQyNDU3MzE.*_ga*MTEwOTQzNzc1OC4xNjkxODM5NDAy*_ga_53S71ZZG1X*czE3NTc4Mzc0ODAkbzM5MSRnMSR0MTc1Nzg0NDg4NiRqMTgkbDAkaDUyMjU4MjIyNw..#magnitude-of-a-number)
6. [Best Practices](https://companion.scaler.com/meeting/class/858011/notes?_gl=1*2m1h9n*_gcl_gb*R0NMLjE3NTYzOTMzMzIuQ2xvS0NRancyYnJGQmhENUFSSkpBTGpyRC1FZkdFTFBVT1VRX1dCcF8tNmY1bWVrYnByRkNmMVpnREtiS2t5cEZGYThTWi12Mkt4M3pHa1JCWVVxdFB6R3RncDQ5UDFHTHp5NHR4SmJYbmhCOE5Vc2VqaklhQm9Dd1NBLkNOZTZDS2JGdk9ZQkVMeV8wNkVELmlZc2xDT0NrX01rQ0VMeV8wNkVELkJaSUVDTVNhMFprWkVMeV8wNkVELnpnZDFDSkd4N2ZZYUVMeV8wNkVELm1jS05DUHZDajRzWUVLRHo1YVlwLmFqVVJDT1duMXZZYUVLRHo1YVlwLjFrVTVDUEdUcUlVWkVJRHFrOW9xLjNKZGVDSVBFMV9ZYUVJRHFrOW9xLmF6ZTZDT3pIMWZzWUVKbVVqOG9x*_gcl_au*NjA1MTY2NDYxLjE3NTQyNDU4MzUuMTgwNjA3NzQ0Mi4xNzU2MzkzMzI4LjE3NTYzOTMzMzI.*FPAU*NjE4OTk4MTc5LjE3NTQyNDU3MzE.*_ga*MTEwOTQzNzc1OC4xNjkxODM5NDAy*_ga_53S71ZZG1X*czE3NTc4Mzc0ODAkbzM5MSRnMSR0MTc1Nzg0NDg4NiRqMTgkbDAkaDUyMjU4MjIyNw..#best-practices)

**Introduction to Functions**

Functions are fundamental building blocks in Java that help in breaking down complex problems into smaller, manageable pieces. Each function performs a specific task, which makes the code more modular and reusable.

**Function Definitions**

**Basic Syntax**

A function in Java is defined with a return type, a name, and any number of parameters. Here is a basic template:

public static <ReturnType> functionName(<Parameters>) {

// function body

}

**Example**

public static int addNumbers(int a, int b) {

return a + b;

}

**Function Return Types**

The return type of a function is the type of value that the function returns. Here are some common return types:

* int for integer values.
* double for fractional numbers.
* String for sequences of characters.
* boolean for true/false values.
* void if no value is returned.

**Example**

public static boolean isEven(int num) {

return num % 2 == 0;

}

**Calling Functions**

Functions are called by their name and passing the required parameters. The result can be stored in a variable or used directly.

**Example**

public static void main(String[] args) {

int result = addNumbers(5, 7);

System.out.println(result); // Output: 12

}

**Practical Examples**

**Count Factors of a Number**

To check whether a number is prime, we sometimes need to count its factors. Here’s how:

**countFactors Function**

public static int countFactors(int number) {

int count = 0;

for (int i = 1; i <= number; i++) {

if (number % i == 0) {

count++;

}

}

return count;

}

**Sum of Digits**

To find the sum of the digits of a number:

**sumOfDigits Function**

public static int sumOfDigits(int num) {

int sum = 0;

while (num > 0) {

sum += num % 10;

num /= 10;

}

return sum;

}

**Check Even or Odd**

Multiple ways to check if a number is even or odd:

**isEven Function**

public static boolean isEven(int num) {

return num % 2 == 0;

}

**checkEven Function**

public static String checkEven(int num) {

if (num % 2 == 0) {

return "Even";

} else {

return "Odd";

}

}

**checkEvenAndPrint Function**

public static void checkEvenAndPrint(int num) {

if (num % 2 == 0) {

System.out.println(num + " is even");

} else {

System.out.println(num + " is odd");

}

}

**Magnitude of a Number**

Determine if a number is small, medium, or large:

**checkMagnitude Function**

public static String checkMagnitude(int num) {

if (num < 10) {

return "Small";

} else if (num <= 20) {

return "Medium";

} else {

return "Large";

}

}

**Best Practices**

1. **Naming Conventions:** Use meaningful names for functions to indicate their purpose.
2. **Single Responsibility:** Each function should perform a single task or logic.
3. **Modularity:** Break down complex tasks into smaller functions.
4. **Return Types:** Use appropriate return types and handle all possible return paths.
5. **Documentation:** Comment and document functions for easier understanding and maintenance.

By following these principles, you can write efficient, readable, and maintainable code in Java.

These notes cover the essential aspects of functions as discussed in your live class. Make sure to practice writing and calling functions to solidify your understanding.

**Analogies Discussed**

Using functions is like delegating tasks to different team members in a project. Each member (function) has a clear role and responsibility, making the project (program) more organized and manageable.

References:

* [transcript.txt](https://companion.scaler.com/meeting/class/858011/8)

<https://github.com/KingsGambitLab/Lecture_Notes/blob/non-dsa/Academy%20DSA%20Typed%20Notes/Java%20Refresher/Refresher%20Functions.md>

FUNCTION:

A block of code designed to perform a specific task

PARAMETERS:

A variable used in a function to refer to one of the pieces of data provided as input to the function.

RETURN TYPE:

The data type of the value a function returns.

VOID:

A function return type meaning the function doesn’t return value.

BOOLEAN:

A datatype with two possible values: true or false.

PRIME NUMBER:

A number greater than 1 that has no positive divisors other than 1 and itself.

COUNT FACTORS:

A function to calculate the number of factors of a given number.

CHECK PRIME:

A function to check if a given number is prime.

REDUNDANCY:

Unnecessary repetition in code.

READABILITY:

The ease with which a human reader can comprehend the purpose and flow of code.

MAINTAINABILITY:

The ease with which code can be modified or extended.

MODULUS OPERATOR:

An operator that returns the remainder when one integer is divided by another.

BALCK BOX TECHNIQUE:

A programming practice where a function or process is used without any knowledge of the internal working.

SUM OF DIGITS:

A function to compute the sum of all digits in a given number.

Functions:

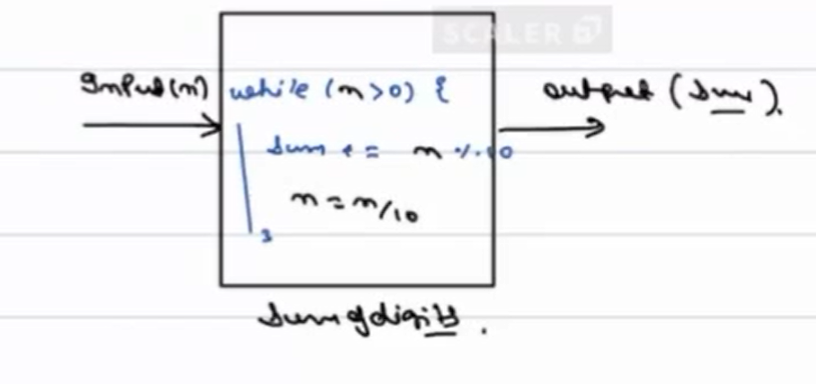
Without Function Issue:

Redundancy

Readability

Maintainability

A function for outer user is more like black box.



Syntax:

Return\_Value\_Data\_Type Function\_Name(InputType1 Input1, InputType2 Input2,..So On){

//Code/Logic

:

:

Return Value;

}

public class Main {

  public static int int2Sum(int a,int b){

    return a+b;

  }

  public static void main(String[] args) {

     System.out.print(int2Sum(10,15));

  }

}

If we not returning anything then return type is void.

With return type void we can write return; <- there is no meaning , if we not written it code still work.

Whatever written after return statement is not reachable as that code never be execute so we get unreachable statement exception.