Name: _____

Java Static Functions

What are Static Functions?

A **static function** (also called a static method) is a function that belongs to the class itself, not to any specific object. You can call a static function without creating an instance of the class. Static functions are useful for utility methods and operations that don't need to access object data.

Simple Analogy: Static Functions are Like Vending Machines

Think of static functions like public utilities that anyone can use:

- **Vending Machine** = You don't need to own the machine to use it
  - You just walk up and use the function (insert money, get snack)
- **Phone Booth** = Anyone can use it without owning it
  - It's always available and works the same for everyone
- **Calculator App** = The Math.sqrt() function works for everyone
  - You don't need to "own" a Math object to use it

Static functions are shared tools - available to everyone without needing a personal copy!

Key Static Function Concepts

- **Belongs to the class:** Not tied to any specific object
- **Called using class name:** ClassName.functionName()
- **Can't access instance variables:** Only works with parameters and local variables
- **Shared by all:** Everyone uses the same function
- **main() is static:** That's why your program can start without creating objects!

Static Function Syntax

Basic Pattern

```
public static returnType functionName(parameters) {
 // code here
 return result; // if returnType is not void
}
```

Parts explained:

- **`public`** - Can be called from anywhere
- **`static`** - Belongs to the class, not objects
- **`returnType`** - What the function gives back (int, double, String, void)
- **`functionName`** - What you call the function
- **`parameters`** - Input values the function needs

Creating and Using Static Functions

Example 1: Simple Static Function (No Return Value)

```
public class Greeter {

 // Static function that prints a greeting
 public static void sayHello(String name) {
  System.out.println("Hello, " + name + "!");
 }

 public static void main(String[] args) {
  // Call the static function directly
  sayHello("Alice");
  sayHello("Bob");
  sayHello("Charlie");
 }
}
```

```
Hello, Alice!
Hello, Bob!
Hello, Charlie!
```

Example 2: Static Function with Return Value

```
public class Calculator {

 // Static function that returns a value
 public static int add(int a, int b) {
  return a + b;
 }

 public static int multiply(int a, int b) {
  return a * b;
 }

 public static void main(String[] args) {
  // Use return values from static functions
  int sum = add(5, 3);
  int product = multiply(4, 7);

  System.out.println("5 + 3 = " + sum);
  System.out.println("4 * 7 = " + product);
 }
}
```

```
5 + 3 = 8
4 * 7 = 28
```

Example 3: Multiple Parameters and Logic

```java
public class MathHelper {

 // Find the larger of two numbers
 public static int findMax(int a, int b) {
  if (a > b) {
   return a;
  } else {
   return b;
  }
 }

 // Check if a number is even
 public static boolean isEven(int number) {
  return number % 2 == 0;
 }

 // Calculate average of three numbers
 public static double average(double a, double b, double c) {
  return (a + b + c) / 3.0;
 }

 public static void main(String[] args) {
  int max = findMax(15, 23);
  System.out.println("Maximum: " + max);

  boolean result = isEven(10);
  System.out.println("Is 10 even? " + result);

  double avg = average(85.5, 92.0, 78.5);
  System.out.println("Average: " + avg);
 }
}
```

```
Maximum: 23
Is 10 even? true
Average: 85.33333333333333
```

Calling Static Functions from Other Classes

Example 4: Using ClassName.functionName()

```java
// MathTools.java
public class MathTools {
 public static int square(int n) {
  return n * n;
 }

 public static int cube(int n) {
  return n * n * n;
 }
}
```

```java
// Main.java
public class Main {
 public static void main(String[] args) {
  // Call static functions from another class
  int result1 = MathTools.square(5);
  int result2 = MathTools.cube(3);

  System.out.println("5 squared = " + result1);
  System.out.println("3 cubed = " + result2);
 }
}
```

```
5 squared = 25
3 cubed = 27
```

⚠ Important Rules for Static Functions

- **Static functions CANNOT access instance variables** (non-static variables)
- **Static functions CAN only call other static functions directly**
- **Static functions work with parameters and local variables only**
- **You don't need to create an object to use static functions**

```java
public class Example {
 int instanceVar = 10; // Non-static variable
 static int staticVar = 20; // Static variable

 public static void staticMethod() {
  // System.out.println(instanceVar); // ERROR! Can't access
  System.out.println(staticVar); // OK! Can access
 }
}
```

Practical Examples

Example 5: Temperature Converter

```java
public class TemperatureConverter {

 public static double celsiusToFahrenheit(double celsius) {
  return (celsius * 9.0 / 5.0) + 32;
 }

 public static double fahrenheitToCelsius(double fahrenheit) {
  return (fahrenheit - 32) * 5.0 / 9.0;
 }

 public static void main(String[] args) {
  double temp1 = celsiusToFahrenheit(25);
```

```
  double temp2 = fahrenheitToCelsius(98.6);

  System.out.println("25°C = " + temp1 + "°F");
  System.out.println("98.6°F = " + temp2 + "°C");
 }
}
```

Example 6: String Utilities

```
public class StringUtils {

 // Count how many times a character appears
 public static int countChar(String text, char target) {
  int count = 0;
  for (int i = 0; i < text.length(); i++) {
   if (text.charAt(i) == target) {
    count++;
   }
  }
  return count;
 }

 // Reverse a string
 public static String reverse(String text) {
  String result = "";
  for (int i = text.length() - 1; i >= 0; i--) {
   result += text.charAt(i);
  }
  return result;
 }

 public static void main(String[] args) {
  String word = "programming";
  int gCount = countChar(word, 'g');
  String reversed = reverse(word);

  System.out.println("Word: " + word);
  System.out.println("Letter 'g' appears: " + gCount + " times");
  System.out.println("Reversed: " + reversed);
 }
}
```

Example 7: Array Helper Functions

```
public class ArrayHelper {

 // Find sum of array elements
 public static int sum(int[] numbers) {
  int total = 0;
  for (int num : numbers) {
```

```
    total += num;
   }
   return total;
  }

  // Find average of array elements
  public static double average(int[] numbers) {
   if (numbers.length == 0) {
    return 0;
   }
   return (double) sum(numbers) / numbers.length;
  }

  // Find maximum value in array
  public static int findMax(int[] numbers) {
   int max = numbers[0];
   for (int num : numbers) {
    if (num > max) {
     max = num;
    }
   }
   return max;
  }

  public static void main(String[] args) {
   int[] scores = {85, 92, 78, 95, 88};

   System.out.println("Sum: " + sum(scores));
   System.out.println("Average: " + average(scores));
   System.out.println("Highest: " + findMax(scores));
  }
}
```

```
Sum: 438
Average: 87.6
Highest: 95
```

Static vs Non-Static Comparison

Static Functions

```
public class Example {
 public static void greet() {
  System.out.println("Hi!");
 }

 public static void main(String[] args) {
  // Call directly
  greet();
 }
}
```

✅ **No object needed**

Non-Static Functions

```
public class Example {
 public void greet() {
  System.out.println("Hi!");
 }

 public static void main(String[] args) {
  // Must create object first
  Example obj = new Example();
  obj.greet();
 }
}
```

❌ **Requires object**

Common Uses for Static Functions

Use Case	Example	Why Static?
Utility functions	Math.sqrt(), Math.pow()	Same result for everyone, no object data needed
Helper methods	Converting units, formatting strings	Pure calculation, no state required
Validation	Checking if email is valid	Just checks input, doesn't store anything
Main method	public static void main(String[] args)	Entry point - must work before objects exist

When to Use Static

- Utility/helper functions
- Math calculations
- Functions that don't need object data
- Conversions and formatting
- Functions called from main()
- Shared functionality across all instances

When NOT to Use Static

- Function needs object variables
- Different behavior per object
- Working with instance state
- Object-specific operations
- When you want polymorphism
- Methods that modify object data

Quick Reference

Static Function Checklist

To create a static function:

1. Add **`public static`** keywords
2. Specify return type (or void)
3. Give it a descriptive name
4. List parameters in parentheses
5. Write the function body in { }
6. Return a value if not void

To call a static function:

- **Same class:** **`functionName(arguments);`**
- **Different class:** **`ClassName.functionName(arguments);`**

Complete Example: Grade Calculator with Multiple Functions

```
import java.util.Scanner;

public class GradeCalculator {

 // Calculate letter grade from percentage
 public static String getLetterGrade(double percentage) {
  if (percentage >= 90) return "A";
  else if (percentage >= 80) return "B";
  else if (percentage >= 70) return "C";
  else if (percentage >= 60) return "D";
  else return "F";
 }

 // Calculate average of test scores
 public static double calculateAverage(double test1, double test2, double test3) {
  return (test1 + test2 + test3) / 3.0;
 }

 // Check if student is passing
 public static boolean isPassing(double average) {
  return average >= 60;
 }

 // Display full grade report
 public static void displayReport(String name, double avg, String letter, boolean passing) {
  System.out.println("\n=== Grade Report ===");
  System.out.println("Student: " + name);
  System.out.println("Average: " + avg);
  System.out.println("Letter Grade: " + letter);
  System.out.println("Status: " + (passing ? "PASSING" : "FAILING"));
 }

 public static void main(String[] args) {
  Scanner input = new Scanner(System.in);

  System.out.print("Enter student name: ");
  String name = input.nextLine();
```

```
        System.out.print("Enter test 1 score: ");
        double test1 = input.nextDouble();

        System.out.print("Enter test 2 score: ");
        double test2 = input.nextDouble();

        System.out.print("Enter test 3 score: ");
        double test3 = input.nextDouble();

        // Use all our static functions
        double average = calculateAverage(test1, test2, test3);
        String letterGrade = getLetterGrade(average);
        boolean passing = isPassing(average);

        displayReport(name, average, letterGrade, passing);

        input.close();
    }
}
```

Your Turn: Write Your Own Definition

What is a static function in Java? How would you explain it to a friend?

Write your definition in your own words:

Complete the following static function to calculate the area of a rectangle:

```
public static _____________ calculateArea(double length, double width) {
 return ____________________________;
}
```

Write a static function called "isPositive" that takes an integer and returns true if it's positive:

```

```

Explain the difference between calling a static function and a non-static function:

Give three examples of when you would use a static function:

1. _____

2. _____

3. _____