Kotlin – Basics

Tushar B. Kute,
http://tusharkute.com

Kotlin Program Entry Point

- Concept: Every executable Kotlin application needs a starting point.
- Function Name: This entry point is always a function named main.
- Location: It can be a top-level function (not inside a class) or a static member of a class
- Purpose: When you run a Kotlin program, the execution begins inside this main function.
- Syntax Example (Conceptual):

```
fun main() {
    // Your program starts here
}
```

- Note: `fun` is the keyword for declaring a function.

Entry Point with Parameters

- Concept: The main function can also accept command-line arguments.
- Parameter Type: These arguments are passed as an array of strings.
- Use Case: Useful when your program needs input directly from the command line when it's launched.
- Syntax Example (Conceptual):

```
fun main(args: Array<String>) {
    // 'args' will contain command-line arguments
}
```

- Note: Array<String> specifies an array where each element is a String.

print() vs println()

- Purpose: Both functions are used to display output to the console.
- `print()`:
  - Prints its argument to the console.
  - Does NOT add a new line character at the end.
  - Subsequent output will appear on the same line.
- `println()`:
  - Prints its argument to the console.
  - Adds a new line character after the output.
  - Subsequent output will appear on the next line.
- Analogy: Think of println() as pressing "Enter" after typing, while print() does not.

Semicolon (;) in Kotlin

- Rule: Semicolons are optional in Kotlin.
- When they are used:
  - They can be used to separate multiple statements on the same line.
  - This is generally discouraged for readability.
- Best Practice:
  - It is highly recommended to omit semicolons at the end of statements.
  - Kotlin's compiler can infer statement endings based on newlines and syntax.
- Contrast: This is a notable difference from languages like Java or C++, where semicolons are mandatory statement terminators.

Packages in Kotlin

- Packages are used to organize code into logical groups.
- Purpose:
  - Avoid Naming Collisions: Prevents conflicts when different files or libraries use the same class or function names.
  - Improve Readability: Makes it easier to understand the purpose and location of code.
  - Control Visibility: Can affect the visibility of classes and functions (though not covered in detail here).
- Declaration: Declared at the very top of a Kotlin file using the package keyword.
- Importing: To use code from another package, you use the import keyword.
- Default Package: If no package is declared, the code is considered part of the "default package."

Packages in Kotlin

- Syntax Example (Conceptual):

```
package com.example.myproject
// Declares the package

import kotlin.io.println
// Imports a specific function (often implicit)

// Your code goes here
```

Conclusion

- We've covered the fundamental building blocks of a Kotlin program:
  - The main entry point.
  - Outputting text with print() and println().
  - The optional nature of semicolons.
  - The importance of packages for code organization.
- These basic concepts form the foundation for writing any Kotlin application.

Thank you

@mitu_skillologies

@mITuSkillologies

@mitu_group

@mitu-skillologies

@MITUSkillologies

kaggle

@mituskillologies

Web Resources
https://mitu.co.in
http://tusharkute.com

@mituskillologies

contact@mitu.co.in

tushar@tusharkute.com