Hello World

কটলিন কিঃ

কটলিন একটি ওপেন সোর্স, স্ট্যাটিকালি-টাইপড, মাল্টি প্যারাডাইম প্রোগ্রামিং ল্যাঙ্গুয়েজ, যা অবজেক্ট-ওরিয়েন্টেড প্রোগ্রামিং ও ফাংশনাল প্রোগ্রামিং উভয়ই সাপোর্ট করে। কটলিন প্রোগ্রামিং ল্যাঙ্গুয়েজের সিনট্যাক্স সি, সি++, সি#, জাভা প্রোগ্রামিং ল্যাঙ্গুয়েজের মত সিনট্যাক্স। আর কটলিনে - লাইনের শেষে সেমিকোলন দেয়া অপশনাল, না দিলেও কাজ করবে।

Hello World প্রোগ্রামঃ

```
fun main(args: Array<String>) {
   println("Hello, World!")
}
Output: Hello, World!
```

এখানে main একটি ফাংশন। কটলিন প্রগ্রাম রান করে এক্সিকিউটেবল কটলিন বাইট কোড তৈরি করতে JVM এর সাহায্য নেয়। আর কটলিন প্রগ্রামের এন্ট্রি পয়েন্ট হচ্ছে main ফাংশন। অর্থাৎ কটলিন প্রগ্রাম রান করে main ফাংশন থেকে।

আর println একটি সিস্টেম ফাংশন। কোনো কিছু প্রিন্ট করার জন্য print ও println ফাংশন ব্যবহার করা হয়। print ফাংশনটি সব কিছু একই লাইনে প্রিন্ট করে, অপরদিকে println ফাংশনটি কোনো কিছু প্রিন্ট করে কার্সর পয়েন্টার নতুন লাইনে চলে যায়।

এখানে println একটি সিস্টেম ফাংশন। তাহলে ফাংশন কি? ফাংশন নিয়ে বিস্তারিত ধারণা পরে দেয়া হবে। এখানে শুধুমাত্র বোঝার সুবিধার সুবিধার্তে ফাংশন নিয়ে সামান্য আলোচনা করা হলো।

Function is a set of instructions to perform specific tasks.
তাহলে বলা যায় কোন স্পেসিফিক বা নিদৃষ্ট কাজ করার জন্য সেট অফ ইনিস্ট্রাকশনকে বোঝায়।

```
fun name(args Arguments) {
   // Code
}
```

ফাংশনকে নেম ব্লক কোডও বলা হয়। ফাংশন আর্গুমেন্ট হিসেবে ইনপুট নিয়ে থাকে এবং সেট অফ ইনিস্ট্রাকশন বা সেট অফ কোডের মাধ্যমে আউটপুট জেনারেট করে।

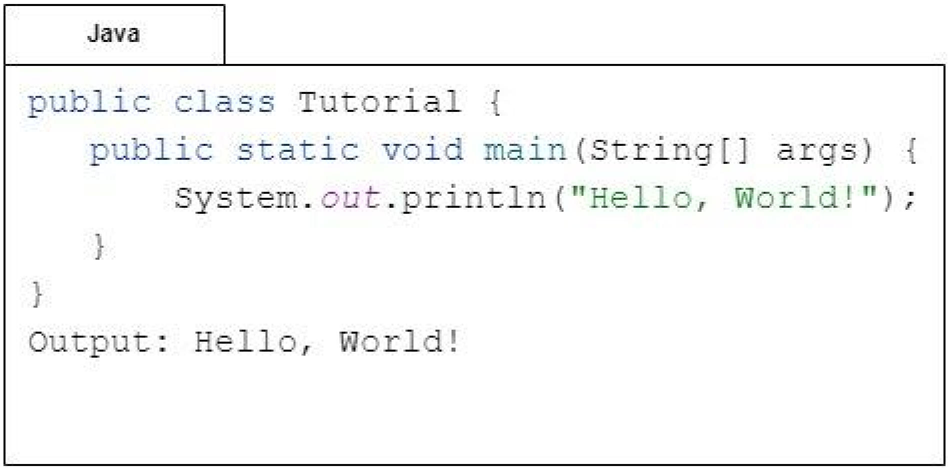
Hello World প্রোগ্রামটি জাভাতেঃ

এখন দেখি Hello World! প্রগ্রামটি জাভাতে কিভাবে লেখা হতো।

```
public class Tutorial {
   public static void main(String[] args) {
       System.out.println("Hello, World!");
   }
}
Output: Hello, World!
```

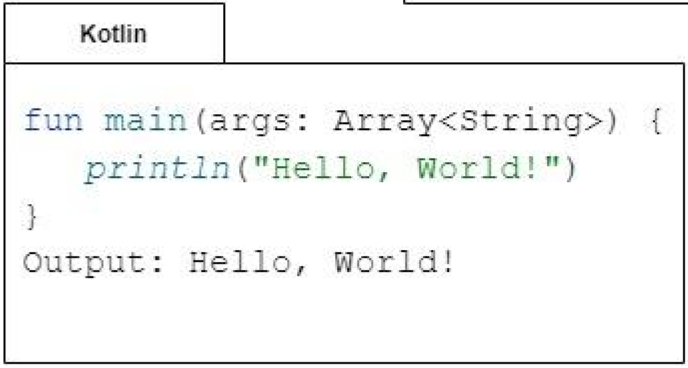
কটলিন ও জাভা কোডের তুলনাঃ

পাবলিক ক্লাস, ক্লাস নেম। ক্লাস নেম এর ভিতরে পাবলিক স্ট্যাটিক main ফাংশন বা main মেথড। যদি কোন জাভা কোডকে কম্পাইল করে এক্সিকিউট করতে হয় তাহলে ক্লাস বা ক্লাস ফাইলের প্রয়োজন হয়। অর্থাৎ জাভা ভার্চুয়াল মেশিন বা JVM শুধুমাত্র ক্লাস ফাইল নিয়ে কাজ করে।



Java

```
public class Tutorial {
   public static void main(String[] args) {
       System.out.println("Hello, World!");
   }
}
Output: Hello, World!
```



Kotlin

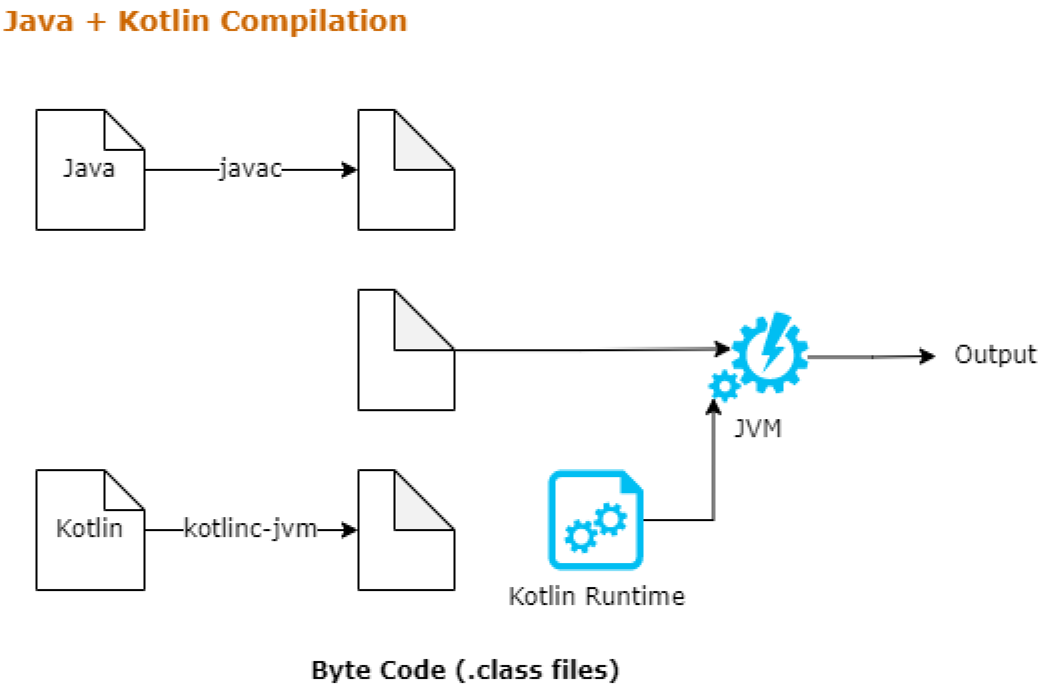
```
fun main(args: Array<String>) {
   println("Hello, World!")
}
Output: Hello, World!
```

অপরপক্ষে কটলিনও জাভা ভার্চুয়াল মেশিন বা JVM এ রান করে। অর্থাৎ কটলিন রানটাইমে কোড এক্সিকিউট করার জন্য JVM এর প্রয়োজন হয়। কিন্তু আমরা জানি কটলিনের main ফাংশন কোন ক্লাসের ভিতর থাকে না। বা কটলিনের main মেথড রান করার জন্য কোন ক্লাসের প্রয়োজন হয় না। তাহলে এখন প্রশ্ন আসে কটলিন কোড রান করার জন্য কি রান টাইমে ইন্টার্নালি ক্লাস তৈরি কর?
এর উত্তর হ্যা।

কটলিন কোড যখন এক্সিকিউট হয়, তখন রানটাইমে কটলিন কম্পাইলার ইন্টার্নালি একটি ক্লাস ফাইল তৈরি করে মেমোরিতে লোড করে। সতরাং কটলিন রানটাইমে জাভার মতোই কাজ করে বা জাভার মতোই আচরণ করে।

কটলিন ও জাভা কোডে রান ও কম্পাইলেশনঃ

কটলিন যেহেতু জাভা ভার্চ্যুয়াল মেশিন (JVM) এ রান করে সুতরাং একটা কটলিন প্রজেক্টে কটলিনের সাথে জাভা ফাইলও থাকতে পারে। অর্থাৎ জাভা কোড ও কটলিন কোড একসাথেই থাকতে পারে। এখন দেখবো জাভা কোড ও কটলিন কোড কিভাবে জাভা ভার্চ্যুয়াল মেশিন বা JVM এ কম্পাইল হয়।



জাভা কোড JAVAC কম্পাইলে, কম্পাইল হয়ে বাইট কোড জেনারেট করবে। আর কটলিন কোড KOTLINC-JVM কম্পাইলে, কম্পাইল হয়ে বাইট কোড জেনারেট করবে। এই দু রকমের বাইট কোড জাভা ভার্চ্যুয়াল মেশিন (JVM) এবং তার সাথে KotlinJavaRuntime মিলে আউটপুট তৈরি করে।

Agenda:

- Hello world example
- Compilation
- Bytecode

Referance

- #2.1 Kotlin Hello World: How it works? Part-2
- Kotlin - Hello World Program | Compilation & ByteCode | CheezyCode #3

End - Hello World!

Variable

Variable প্রোগ্রাম

End - Variable

List of content
- Basic Syntax Comments
- Function
- Class
- String interpolation
- Loop
- Variables and Constants
- Data Types: var vs val

#4.2 Kotlin Data Types: VAR vs VAL: Variables and Constants Part-1
https://youtu.be/qAJTqI_aKJU?list=PLIxmoA0rQ-LwgK1JsnMsakYNACYGa1cjR

List of content
- Variables
- Basic data types
- Arithmetic operators

Kotlin for Beginners - Part 4 - VARIABLES
https://youtu.be/kYO7YWVB3jY?list=PLrnPJCHvNZuAIbejjZA1kGfLeA8ZpICB2

List of content final
- Hello World! How does it work?
- Variables
- Basic data types
- Arithmetic operators

Rz Rasel