

A decorative background featuring a network diagram with nodes and connecting lines. The nodes are represented by circles of varying sizes and colors, including blue, grey, and white. Some nodes are highlighted with blue outlines. The lines are thin and grey, creating a complex web-like structure. The diagram is positioned in the corners of the slide, with a larger concentration on the left side and a smaller one on the bottom right.

# **Kotlin - Why, How and When**



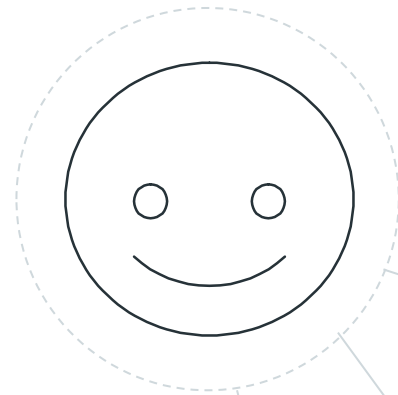
# Hello!

## I am Sadman Samee

I'm here to talk about **kotlin**

You can find me at:

[fb.me/sameesadman](https://fb.me/sameesadman)



A decorative network diagram in the top-left corner, consisting of various sized circles (nodes) connected by thin lines (edges). Some nodes are solid grey, while others are hollow with a grey outline. The network is dense and irregular.

# Why move to Kotlin?

Less codes, less crash, nicer codebase.

A decorative network diagram at the top of the slide, featuring a central node with a blue double quote icon, surrounded by a dashed circle. This central node is connected to several other nodes, which are further connected to a larger, more complex network of nodes and lines extending across the top of the slide. The nodes are represented by small circles, some solid and some dashed, connected by thin lines.

“

*Kotlin is very **concise**.  
Drastically reduce the amount  
of boilerplate code.*

A decorative graphic at the top of the slide featuring a network of interconnected nodes and lines. The nodes are represented by circles of varying sizes, some with concentric rings, and are connected by thin lines. The overall style is light and modern, with a focus on the central quote.

“

*Saves a lot more time to get  
more sleep.*

# In Java we write...

```
final List<Integer> numbers = Arrays.asList(1, 2, 3);

final Map<Integer, String> map = new HashMap<Integer, String>();
map.put(1, "One");
map.put(2, "Two");
map.put(3, "Three");
```

*// Java 9*

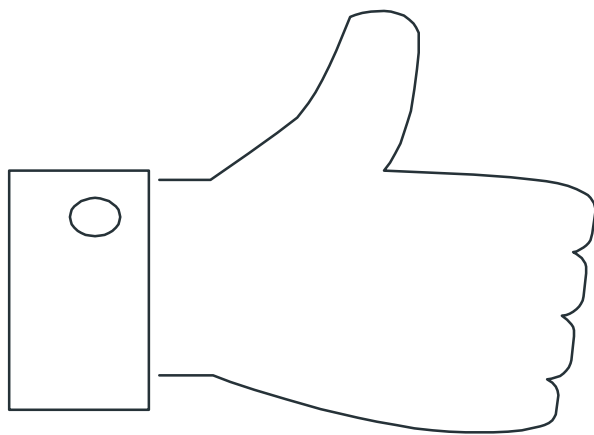
```
final List<Integer> numbers = List.of(1, 2, 3);

final Map<Integer, String> map = Map.of(1, "One",
                                         2, "Two",
                                         3, "Three");
```

# And In Kotlin...

```
val numbers = listOf(1, 2, 3)
```

```
val map = mapOf(1 to "One",  
                2 to "Two",  
                3 to "Three")
```



A decorative network diagram at the top of the slide, featuring a complex web of interconnected nodes and lines. A central node is highlighted with a dashed circle and a blue double quote symbol.

“

*Get more with less code =  
Decreased dev time and cost.*



A decorative network diagram at the top of the slide, featuring a series of interconnected nodes and lines. A central node is highlighted with a dashed circle and a solid circle, containing a large blue double quote symbol.

“

*That's why many big companies  
are moving to **Kotlin***

# Safe

Avoid  
NullPointerException.  
The Billion Dollar  
mistake



# The Pyramid of Doom...

```
if(pyramid != null){
```

```
    if(sheep != null){
```

```
        if(you != null){
```

```
            }
```

```
        }
```

```
    }
```



... will be replaced with this!

```
val name = pyramid?.sheep?.you ?: "idiot"
```



# Less Verbose code

Kotlin requires less numbers of code compared to Java.

## Java

```
private String s1 = "my string 1";  
private final String s2 = "my string 2";  
private static final String s3 = "my string 3";
```

## Kotlin

```
var s1 = "my string 1"  
val s2 = "my string 2"  
val s3 = ""my string 3""
```



When should you move to  
Kotlin?

**ANYTIME!**



Because  
Kotlin costs  
nothing to adapt







**One-click** Java to Kotlin converter.

3 seconds task and 90-99% code  
coverage!





Both languages can be used on the same  
project.

Convenient



Kotlin can use existing **Java libraries**.

Availability



# Thanks!

## Any questions?

You can find me at:

[fb.me/sameesadman](https://fb.me/sameesadman)

sadman.tonmoy@gmail.com