KOTLIN!

Less code, Less bugs, Read easier

STUDY HARD!

KOTLIN FROM JETBRAIN

- Since 2011
- Open Sourced 2012
- Version 1.0 in 2016
- Official for Android Apps 2017

Document https://kotlinlang.org/docs/reference/

BASICS

https://play.kotlinlang.org/

Print to Console

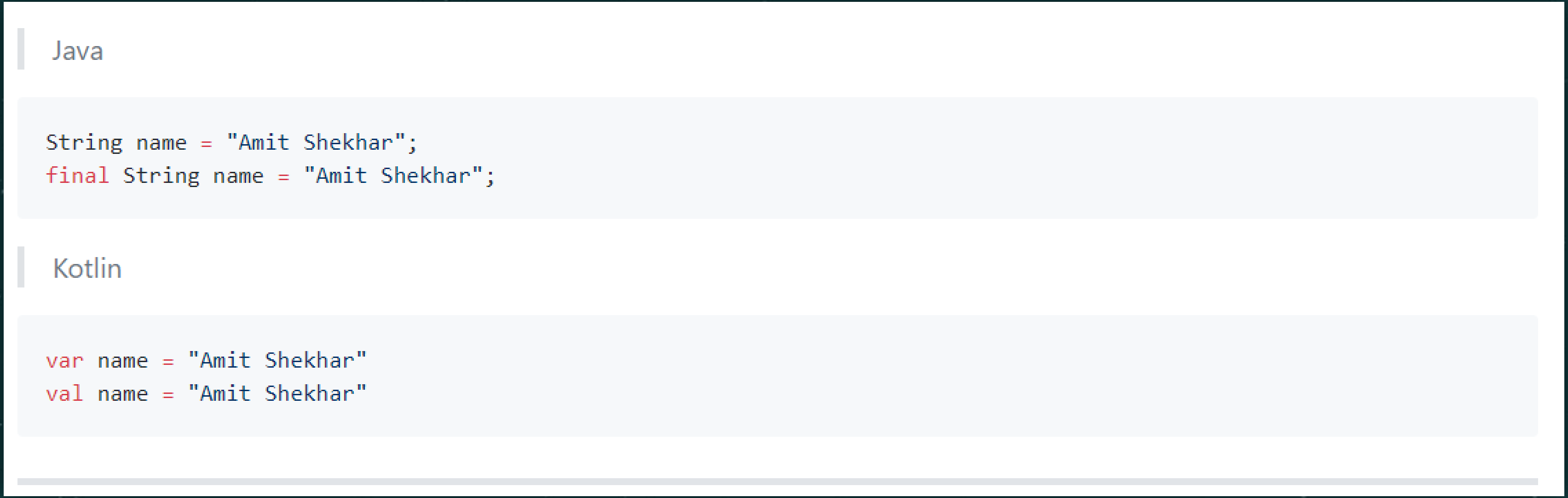
Java

```
System.out.print("Amit Shekhar");
System.out.println("Amit Shekhar");
```

Kotlin

```
print("Amit Shekhar")
println("Amit Shekhar")
```

Constants and Variables



Java

```java
String name = "Amit Shekhar";
final String name = "Amit Shekhar";
```

Kotlin

```kotlin
var name = "Amit Shekhar"
val name = "Amit Shekhar"
```

IMMUTABLE VS MUTABLE

Variables

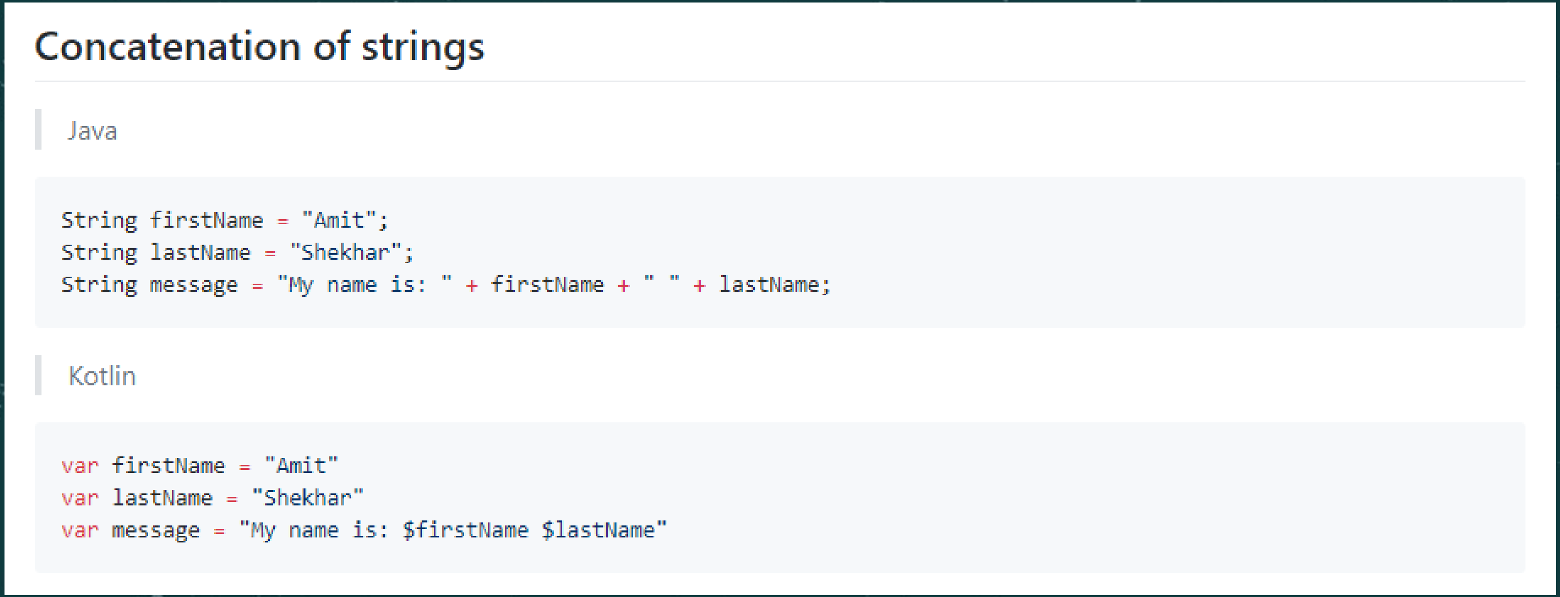
Java

```java
int w;
int z = 2;
z = 3;
w = 1;
```

Kotlin

```kotlin
var w: Int
var z = 2
z = 3
w = 1
```

Concatenation of strings



Concatenation of strings

Java

```
String firstName = "Amit";
String lastName = "Shekhar";
String message = "My name is: " + firstName + " " + lastName;
```

Kotlin

```
var firstName = "Amit"
var lastName = "Shekhar"
var message = "My name is: $firstName $lastName"
```

Assigning the null value

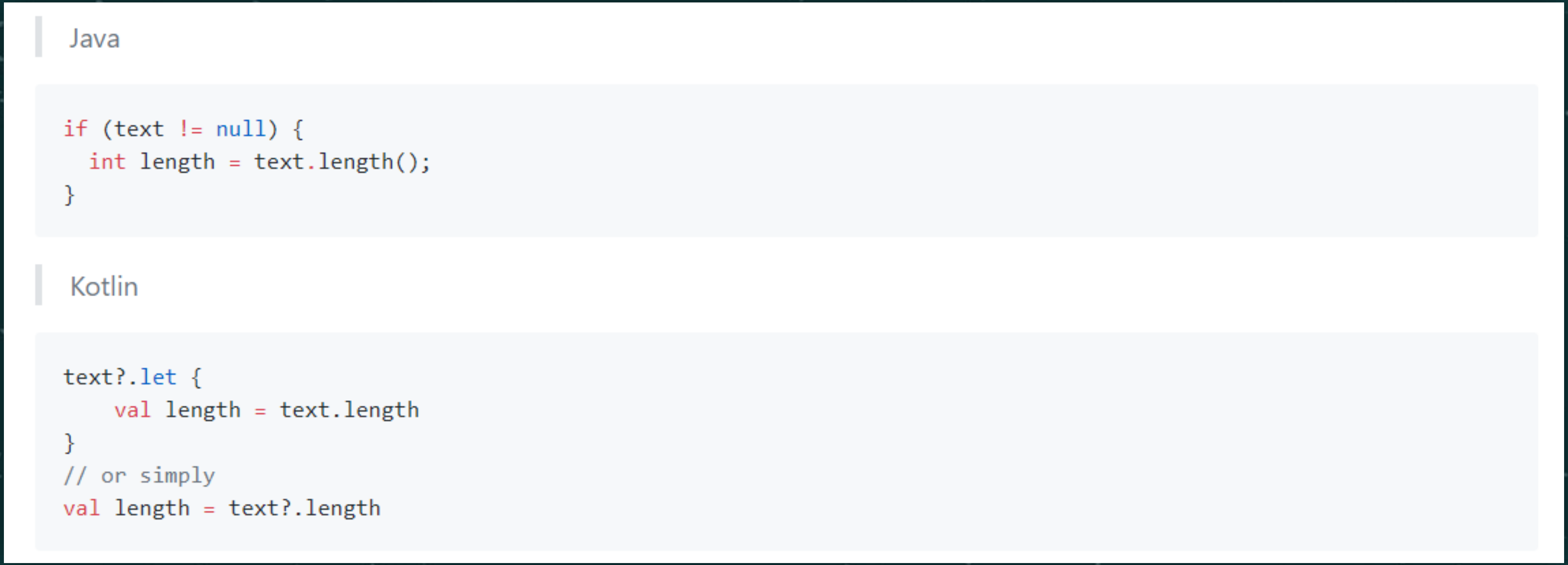
Java

```java
final String name = null;

String lastName;
lastName = null
```

Kotlin

```kotlin
val name: String? = null

var lastName: String?
lastName = null

var firstName: String
firstName = null // Compilation error!!
```

Verify if value is null



Java

```java
if (text != null) {
  int length = text.length();
}
```

Kotlin

```kotlin
text?.let {
    val length = text.length
}
// or simply
val length = text?.length
```

Multiple conditions

Java

```
if (score >= 0 && score <= 300) { }
```

Kotlin

```
if (score in 0..300) { }
```

Smart Cast

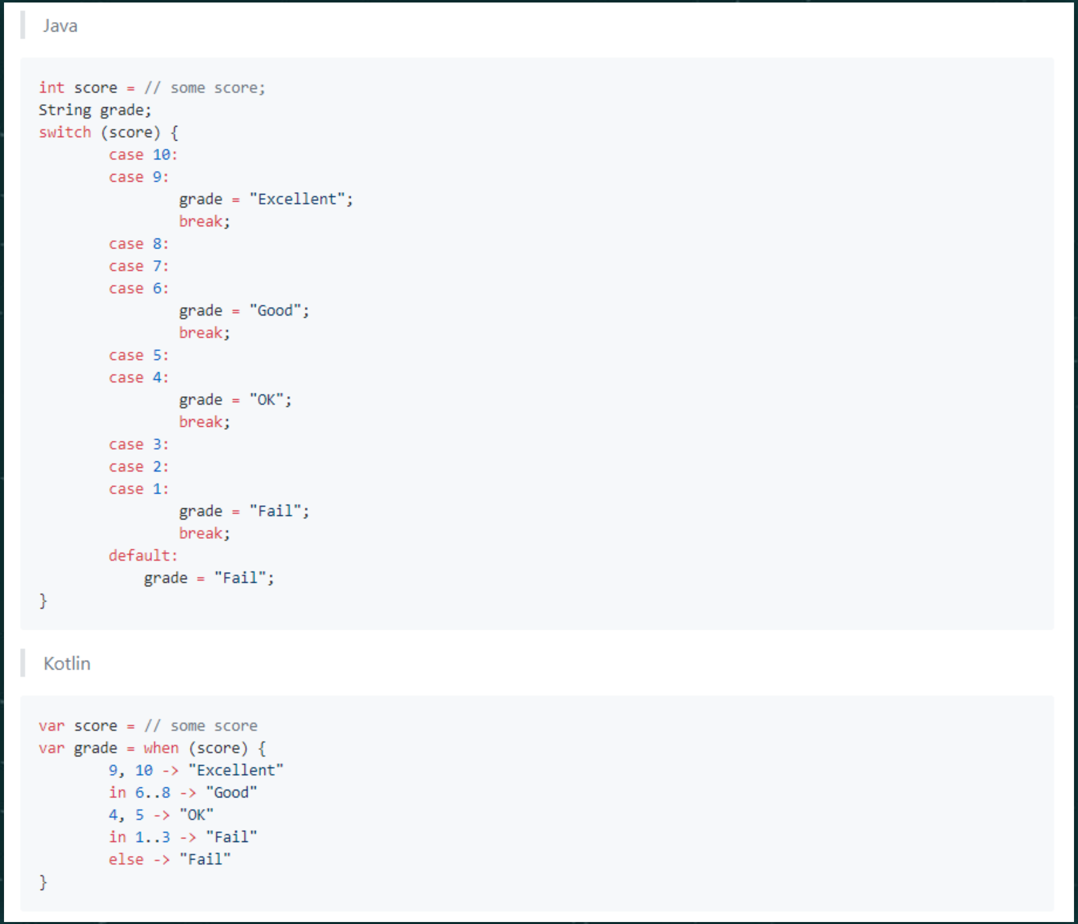
Java

```java
if(a instanceof String){
  final String result = ((String) a).substring(1);
}
```

Kotlin

```kotlin
if (a is String) {
  val result = a.substring(1)
}
```

Multiple Conditions (Switch case)



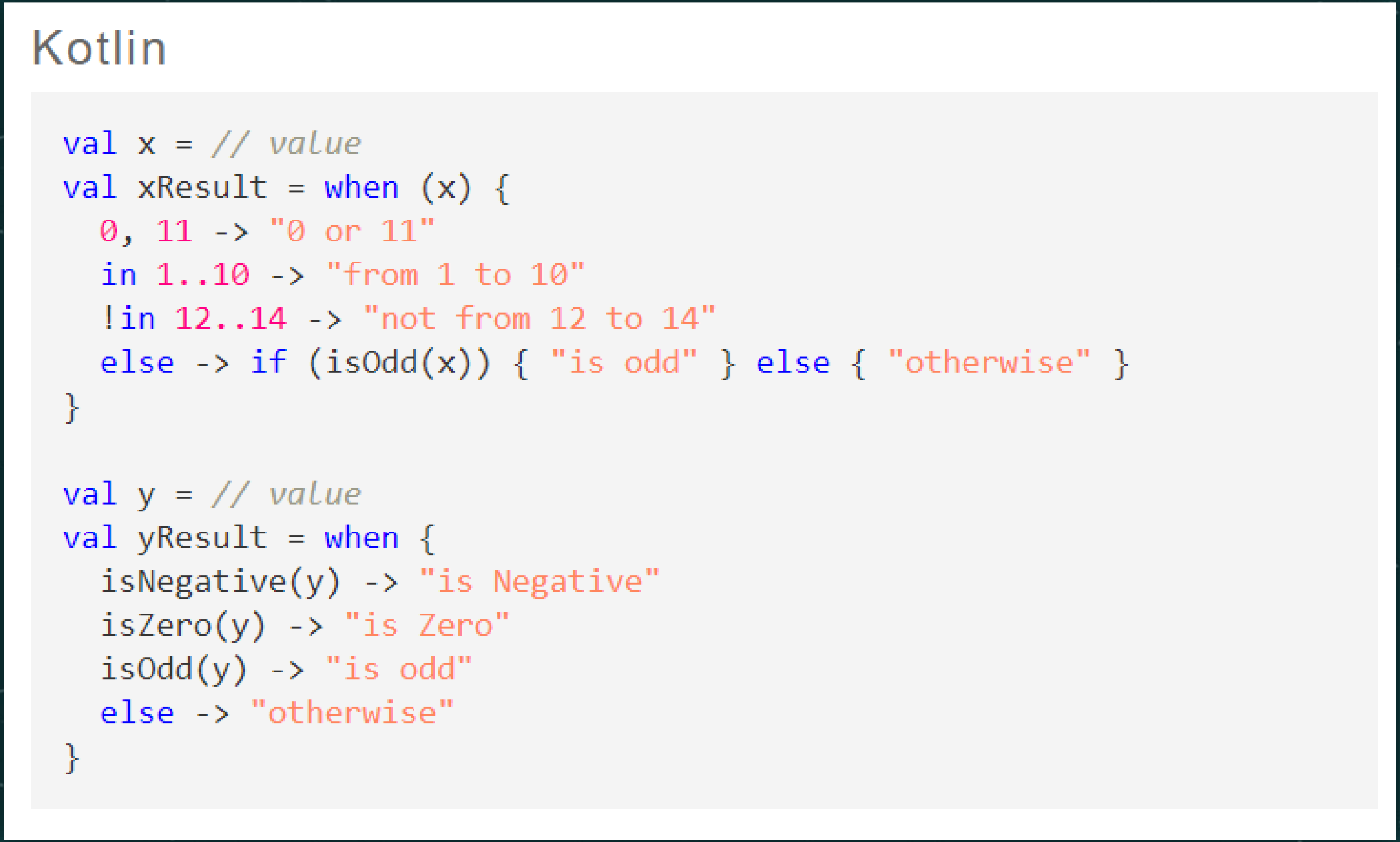
Java

```java
int score = // some score;
String grade;
switch (score) {
        case 10:
        case 9:
                grade = "Excellent";
                break;
        case 8:
        case 7:
        case 6:
                grade = "Good";
                break;
        case 5:
        case 4:
                grade = "OK";
                break;
        case 3:
        case 2:
        case 1:
                grade = "Fail";
                break;
        default:
            grade = "Fail";
}
```

Kotlin

```kotlin
var score = // some score
var grade = when (score) {
        9, 10 -> "Excellent"
        in 6..8 -> "Good"
        4, 5 -> "OK"
        in 1..3 -> "Fail"
        else -> "Fail"
}
```

Multiple Conditions (Switch case)



Kotlin

```kotlin
val x = // value
val xResult = when (x) {
  0, 11 -> "0 or 11"
  in 1..10 -> "from 1 to 10"
  !in 12..14 -> "not from 12 to 14"
  else -> if (isOdd(x)) { "is odd" } else { "otherwise" }
}

val y = // value
val yResult = when {
  isNegative(y) -> "is Negative"
  isZero(y) -> "is Zero"
  isOdd(y) -> "is odd"
  else -> "otherwise"
}
```

For-loops

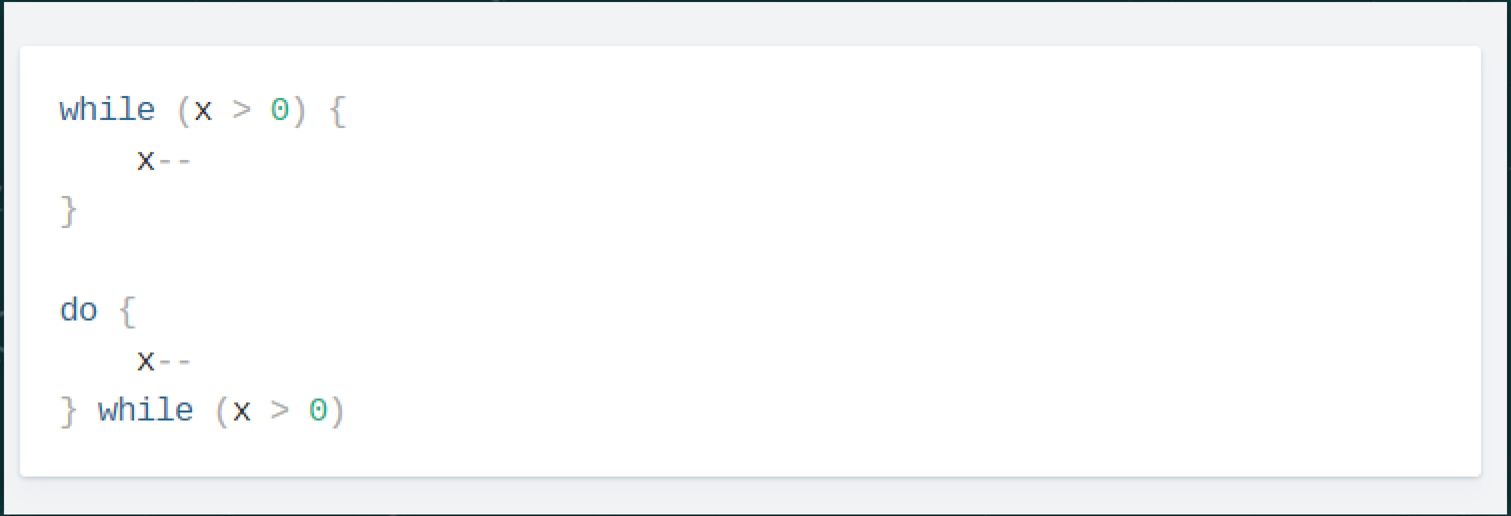
Java

```java
for (int i = 1; i <= 10 ; i++) { }

for (int i = 1; i < 10 ; i++) { }

for (int i = 10; i >= 0 ; i--) { }

for (int i = 1; i <= 10 ; i+=2) { }

for (int i = 10; i >= 0 ; i-=2) { }

for (String item : collection) { }

for (Map.Entry<String, String> entry: map.entrySet()) { }
```

Kotlin

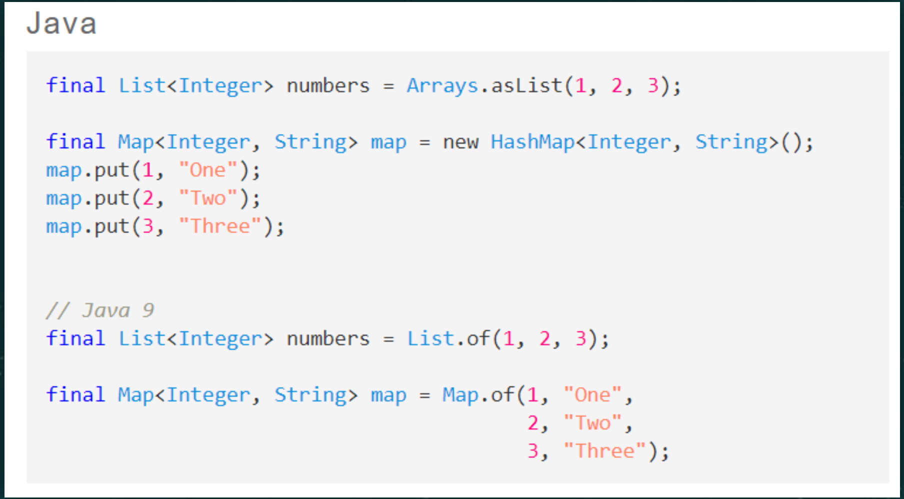
```kotlin
for (i in 1..10) { }

for (i in 1 until 10) { }

for (i in 10 downTo 0) { }

for (i in 1..10 step 2) { }

for (i in 10 downTo 0 step 2) { }

for (item in collection) { }

for ((key, value) in map) { }
```

While Loops

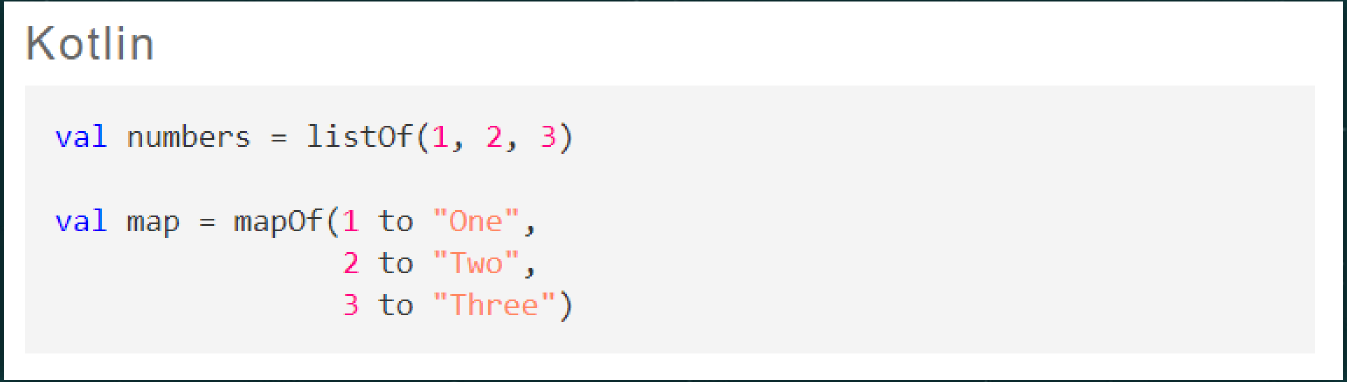


```
while (x > 0) {
    x--
}

do {
    x--
} while (x > 0)
```

Collections


Java

```java
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");
```


Kotlin

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

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

Collections

Java

```
for (int number : numbers) {
  System.out.println(number);
}

for (int number : numbers) {
  if(number > 5) {
    System.out.println(number);
  }
}
```

Kotlin

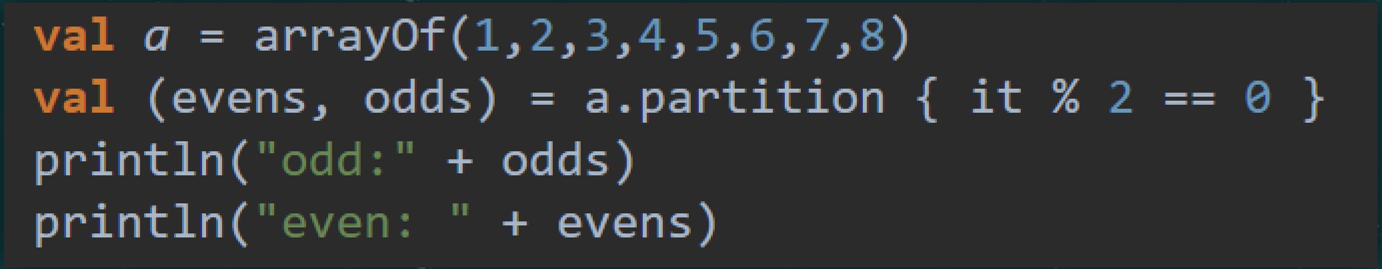
```
numbers.forEach {
    println(it)
}

numbers.filter  { it > 5 }
       .forEach { println(it) }
```

Collections

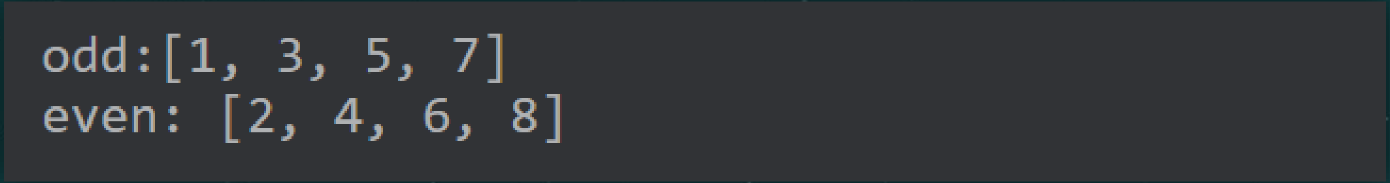
```kotlin
val a = arrayOf(1,2,3,4,5,6,7,8)
val groups = a.groupBy {
            if (it % 2 == 0) "even" else "odd"
         }
println(groups)
println("odd:" + groups["odd"])
println("even: " + groups["even"])
```

```
{odd=[1, 3, 5, 7], even=[2, 4, 6, 8]}
odd:[1, 3, 5, 7]
even: [2, 4, 6, 8]
```

Collections



```kotlin
val a = arrayOf(1,2,3,4,5,6,7,8)
val (evens, odds) = a.partition { it % 2 == 0 }
println("odd:" + odds)
println("even: " + evens)
```



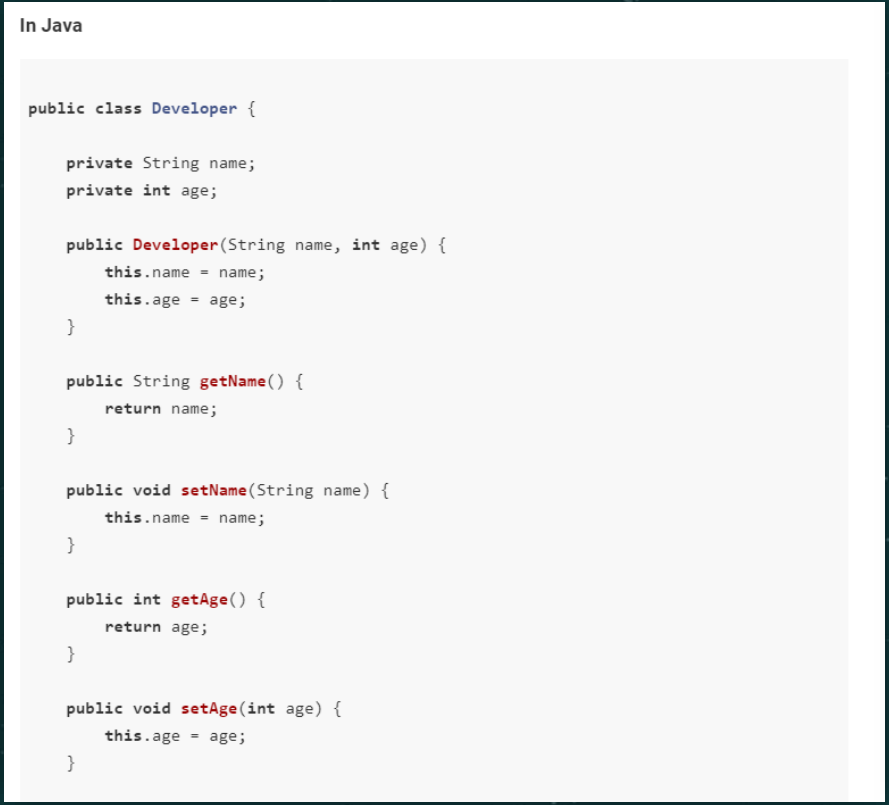
```
odd:[1, 3, 5, 7]
even: [2, 4, 6, 8]
```

Collections

```kotlin
val a = arrayOf(1,2,3,4)
print(a.joinToString(separator = " and ", prefix = "<", postfix = ">"))
```

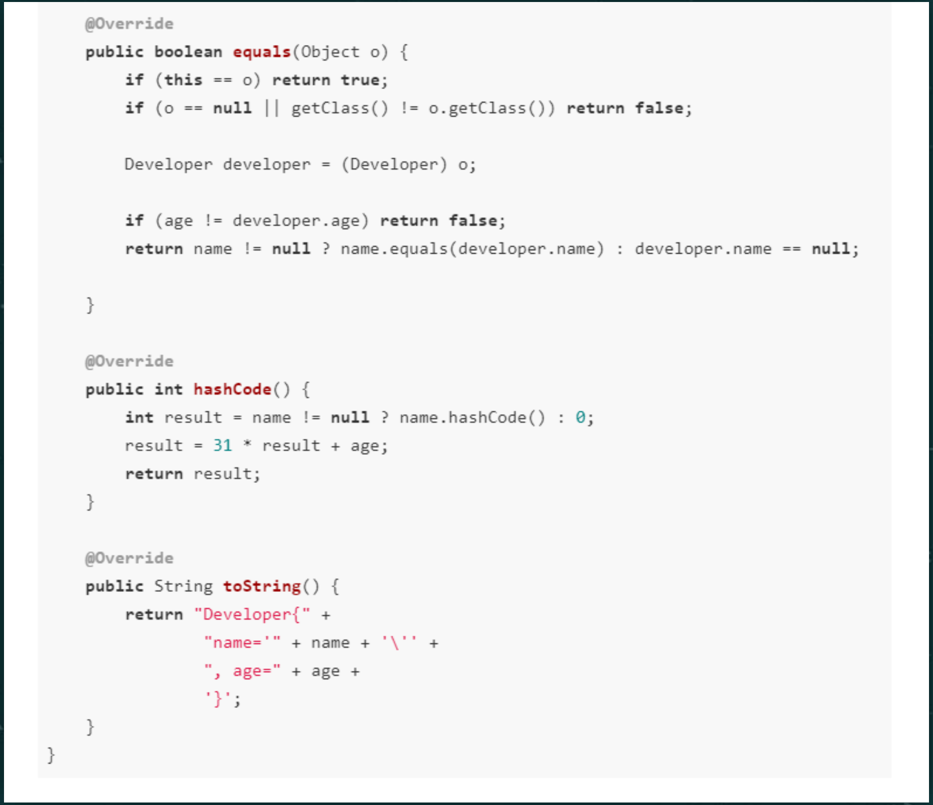
```
<1 and 2 and 3 and 4>
```

Create Data Class



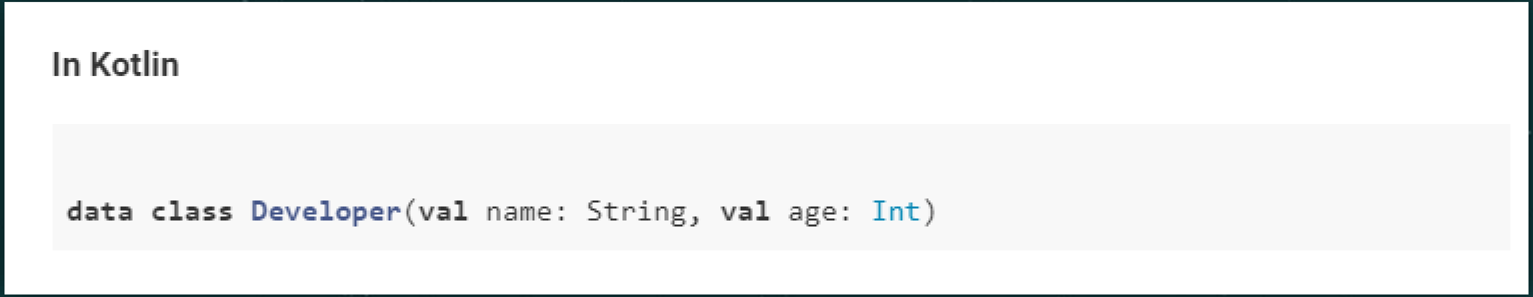
In Java

```java
public class Developer {

    private String name;
    private int age;

    public Developer(String name, int age) {
        this.name = name;
        this.age = age;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public int getAge() {
        return age;
    }

    public void setAge(int age) {
        this.age = age;
    }
```



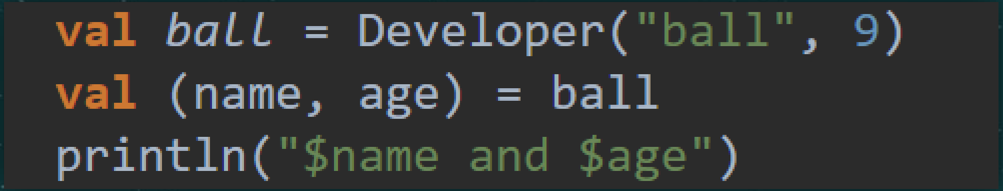
```java
    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;

        Developer developer = (Developer) o;

        if (age != developer.age) return false;
        return name != null ? name.equals(developer.name) : developer.name == null;

    }

    @Override
    public int hashCode() {
        int result = name != null ? name.hashCode() : 0;
        result = 31 * result + age;
        return result;
    }

    @Override
    public String toString() {
        return "Developer{" +
                "name='" + name + '\'' +
                ", age=" + age +
                '}';
    }
}
```

Create Data Class

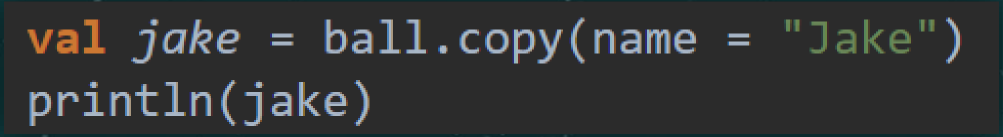


In Kotlin

```
data class Developer(val name: String, val age: Int)
```



```
val ball = Developer("ball", 9)
val (name, age) = ball
println("$name and $age")
```



```
val jake = ball.copy(name = "Jake")
println(jake)
```

Functions

```
fun printName() {
    print("Adam")
}

fun printName(person: Person) {
    print(person.name)
}

fun getGreeting(person: Person): String {
    return "Hello, ${person.name}"
}

fun getGreeting(person: Person): String = "Hello, ${person.name}"
fun getGreeting(person: Person) = "Hello, ${person.name}"
```