Kotlin 1.0!

WHAT?

Pragmatic Language for JVM and Android

Why?

- interoperability
- safety
- clarity
- tooling

interoperability

Data Class

Kotlin Data Class

```
// KotlinFile.kt

data class Niceeee(val such: String, val so: String)

fun main(args: Array<String>) {
    val nice1 = Niceeee("nice", "cool")
    val nice2 = Niceeee(such = "nice", so = "cool")
    println(nice1)
    println(nice2)
    println(nice1 == nice2)
}

// Niceeee(such=nice, so=cool)
// Niceeee(such=nice, so=cool)
// true
```

Data Class

Kotlin

```kotlin
// KotlinFile.kt

data class Niceeee(val such: String, val so: String)

fun main(args: Array<String>) {
    val nice1 = Niceeee("nice", "cool")
    val nice2 = Niceeee(such = "nice", so = "cool")
    println(nice1)
    println(nice2)
    println(nice1 == nice2)
}

// Niceeee(such=nice, so=cool)
// Niceeee(such=nice, so=cool)
// true
```

Java

```java
// JavaClass.java

public class JavaClass {
    public static void main(String[] args) {
        Niceeee niceeee1 = new Niceeee("nice", "cool");
        Niceeee niceeee2 = new Niceeee("nice", "cool");
        System.out.println(niceeee1);
        System.out.println(niceeee2);
        System.out.println(niceeee1.equals(niceeee2));
    }
}

// Niceeee(such=nice, so=cool)
// Niceeee(such=nice, so=cool)
// true
```

Extension Functions

Extension Functions

Kotlin

```kotlin
// KotlinFile.kt

fun main(args: Array<String>) {
    println("Hello, World".allBetterWithBang())
}

fun String.allBetterWithBang(): String {
    return "${this}!"
}

// Hello, World!
```

Extension Functions

Kotlin

```kotlin
// KotlinFile.kt

fun main(args: Array<String>) {
    println("Hello, World".allBetterWithBang())
}

fun String.allBetterWithBang(): String {
    return "${this}!"
}

// Hello, World!
```

Java

```java
// JavaClass.java

public class JavaClass {
    public static void main(String[] args) {
        System.out.println(KotlinFileKt.allBetterWithBang("Hello, World"));
    }
}


// Hello, World!
```

Singleton

Singleton

Kotlin

```
// KotlinFile.kt

object Lonely {
    var state = "I hold state!"
}
```

Singleton

Kotlin

```kotlin
// KotlinFile.kt

object Lonely {
    var state = "I hold state!"
}
```

Java

```java
// JavaClass.java

public class JavaClass {
    public static void main(String[] args) {
        Lonely.INSTANCE.getState();
        Lonely.INSTANCE.setState("No! Java with you!");
    }
}
```

safety

Nope! Pointer Exceptions

Nope! Pointer Exceptions

Java

```java
// JavaClass.java

public class JavaClass {
    public static void main(String[] args) {
        String str = null;
        str.length();
    }
}

// Java: Ok!
```

Nope! Pointer Exceptions

Java

```
// JavaClass.java

public class JavaClass {
    public static void main(String[] args) {
        String str = null;
        str.length();
    }
}

// Java: Ok!
```

Kotlin

```
// KotlinFile.kt

fun main(args: Array<String>) {
    val str: String = null
}

// Kotlin: Null can not be a value of a non-null type kotlin.String
```

Nope! Pointer Exceptions

Nope! Pointer Exceptions

Kotlin

Nope! Pointer Exceptions

Kotlin

```kotlin
// KotlinFile.kt

fun main(args: Array<String>) {
    val str: String? = null // Kotlin: Ok!
}
```

Nope! Pointer Exceptions

Kotlin

```kotlin
// KotlinFile.kt

fun main(args: Array<String>) {
    val str: String? = null // Kotlin: Ok!
}
```

```kotlin
// KotlinFile.kt

fun main(args: Array<String>) {
    val str: String? = null

    str.length
}

// Kotlin: Only safe (?.) or non-null asserted (!!.) calls are allowed on
//   a nullable receiver of type kotlin.String?
```

Nope! Pointer Exceptions

Nope! Pointer Exceptions

Kotlin

Nope! Pointer Exceptions

Kotlin

```
// Safe-call
fun main(args: Array<String>) {
    val str: String? = null

    println(str?.length)
}

// null
```

Nope! Pointer Exceptions

Kotlin

```kotlin
// Safe-call
fun main(args: Array<String>) {
    val str: String? = null

    println(str?.length)
}

// null
```

```kotlin
// Extension function
fun main(args: Array<String>) {
    val str: String? = null

    str?.let {
        println(str.length)
    }
}

// nothing
```

Nope! Pointer Exceptions

Kotlin

```kotlin
// Safe-call
fun main(args: Array<String>) {
    val str: String? = null

    println(str?.length)
}

// null
```

```kotlin
// Extension function
fun main(args: Array<String>) {
    val str: String? = null

    str?.let {
        println(str.length)
    }
}

// nothing
```

```kotlin
// Elvis
fun main(args: Array<String>) {
    val str: String? = null

    val nonNull = str?.length ?: 0

    println(nonNull)
}

// 0
```

Cast?

Cast?

Java

```
public class JavaClass {
    public static void main(String[] args) {
        Object master = "Yoda";

        if (master instanceof String) {
            System.out.println(((String) master).length());
        }
    }
}

// 4
```

Cast?

Java

```java
public class JavaClass {
    public static void main(String[] args) {
        Object master = "Yoda";

        if (master instanceof String) {
            System.out.println(((String) master).length());
        }
    }
}

// 4
```

Kotlin

```kotlin
fun main(args: Array<String>) {
    val master: Any = "Yoda"

    if (master is String) {
        println(master.length)
    }
}

// 4
```

clarity

Kotlin

Kotlin

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

Kotlin

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

```
val (name, age) = person
```

Kotlin

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

```kotlin
val (name, age) = person
```

```kotlin
for ((name, age) in persons) {
     // ...
}
```

Kotlin

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

```kotlin
val (name, age) = person
```

```kotlin
for ((name, age) in persons) {
     // ...
}
```

```kotlin
fun function(...): Person {
    // computations

    return person
}

// Now, to use this function:
val (name, age) = function(...)
```

Kotlin

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

```
val (name, age) = person
```

```
for ((name, age) in persons) {
     // ...
}
```

```
fun function(...): Person {
    // computations

    return person
}

// Now, to use this function:
val (name, age) = function(...)
```

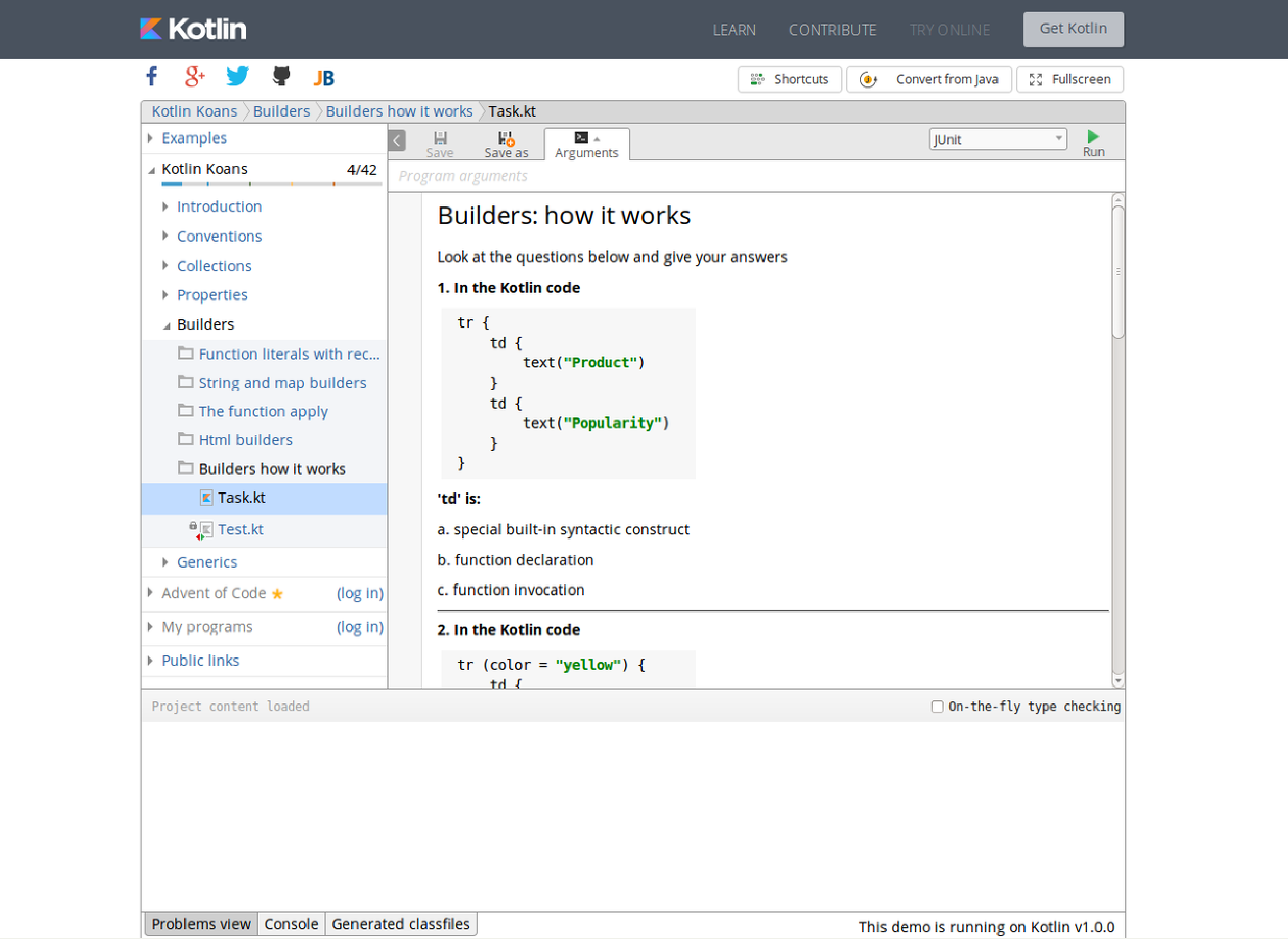
And more, and more...

tooling

- Idea (Ultimate and Community)
- Android Studio
- Eclipse
- Gradle, Maven, Ant and **Kobalt**

Try it!

try.kotl.in



Builders: how it works

Look at the questions below and give your answers

1. In the Kotlin code

```
tr {
    td {
        text("Product")
    }
    td {
        text("Popularity")
    }
}
```

'td' is:

a. special built-in syntactic construct

b. function declaration

c. function invocation

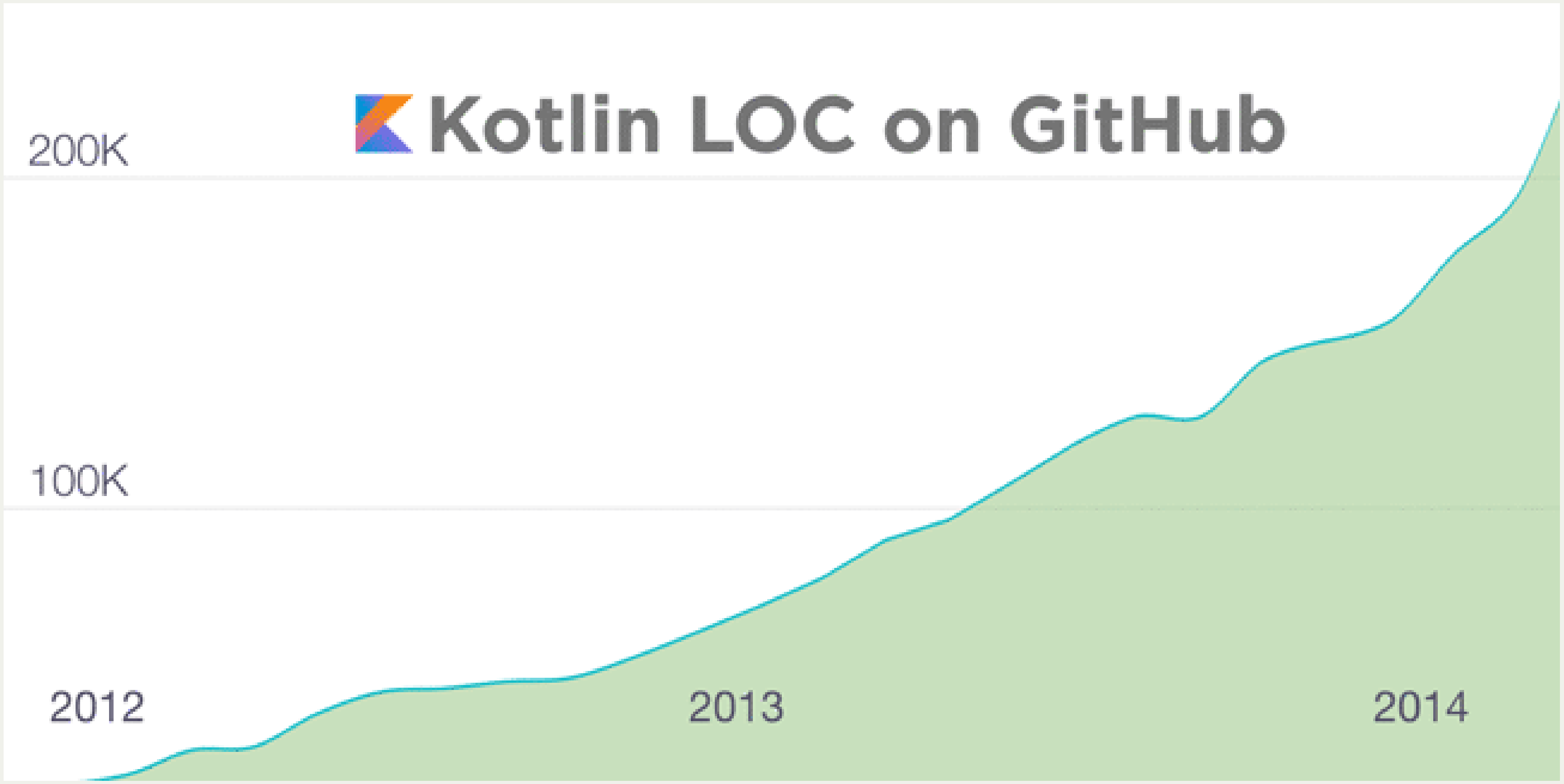
---

2. In the Kotlin code

```
tr (color = "yellow") {
    td {
```

```
Project content loaded
```

- **11K+** people were using Kotlin last month
- **Hundreds** of StackOverflow answers;
- **Two books**: Kotlin in Action and Kotlin for Android Developers;
- **About 1700 people** on Slack;
- **Over 500K** lines of Kotlin code in projects such as IntelliJ IDEA and Rider.


Kotlin LOC on GitHub
200K
100K
2012
2013
2014

Awesome Kotlin

http://kotlin.link

Quiz

In which Class PSVM?

```kotlin
// File.kt

package foo.bar.test

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

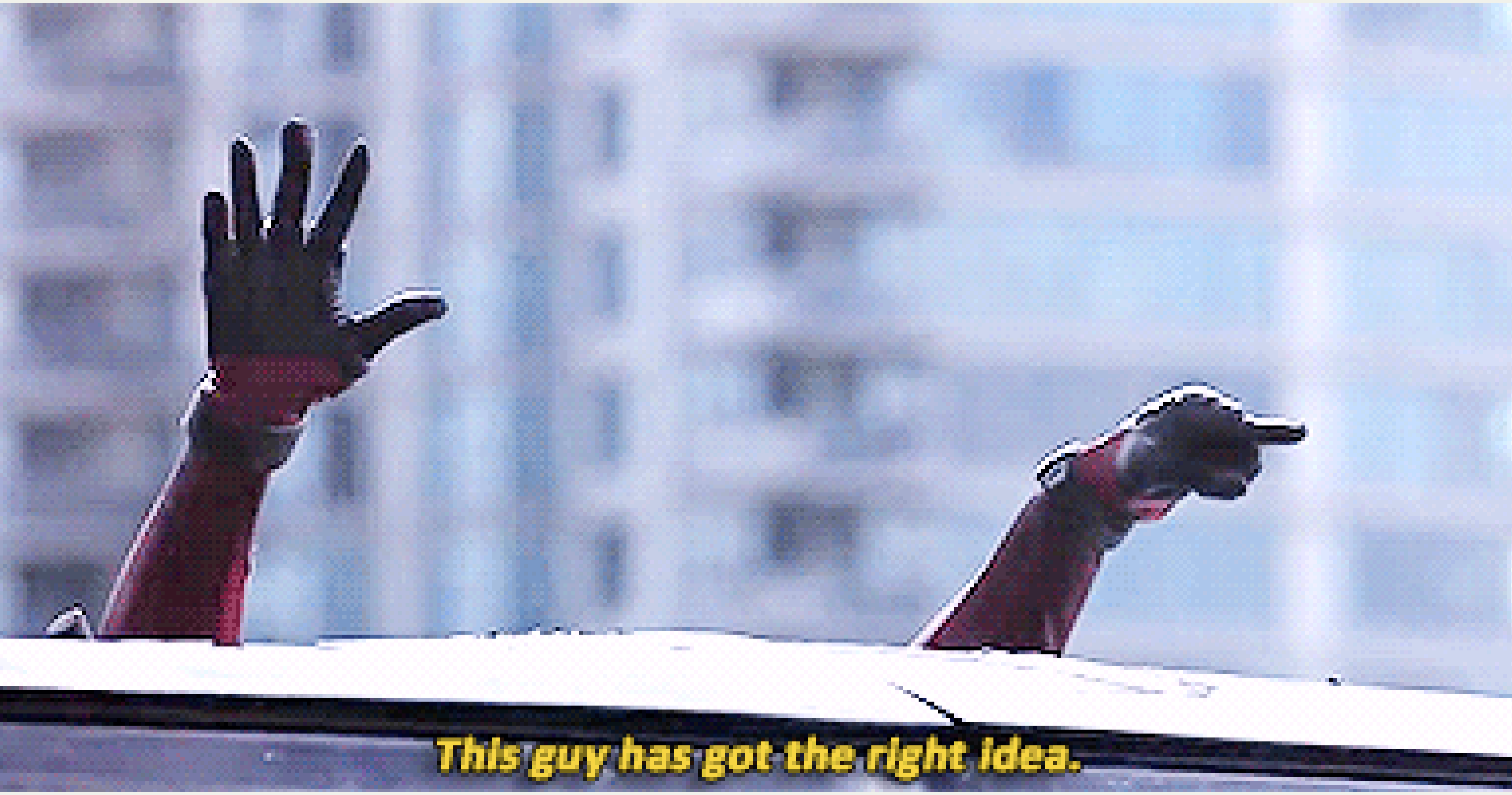
In which Class PSVM?

```kotlin
// File.kt

package foo.bar.test

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

```java
foo.bar.test.FileKt.main(...);
```


This guy has got the right idea.