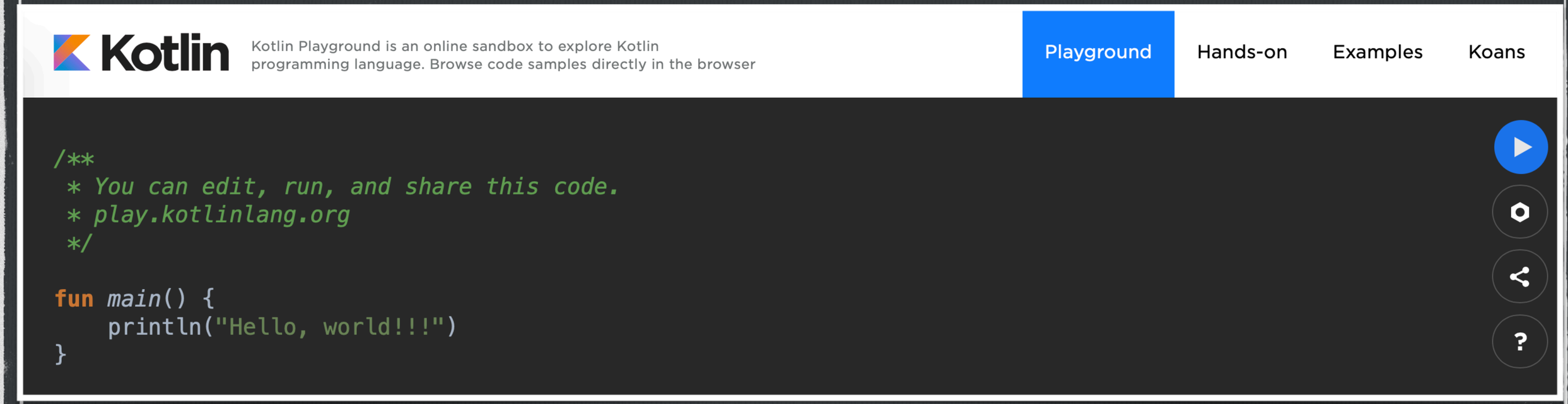
Kotlin

Rafael Vieira Coelho
rafaelvc2@gmail.com

https://kotlinlang.org

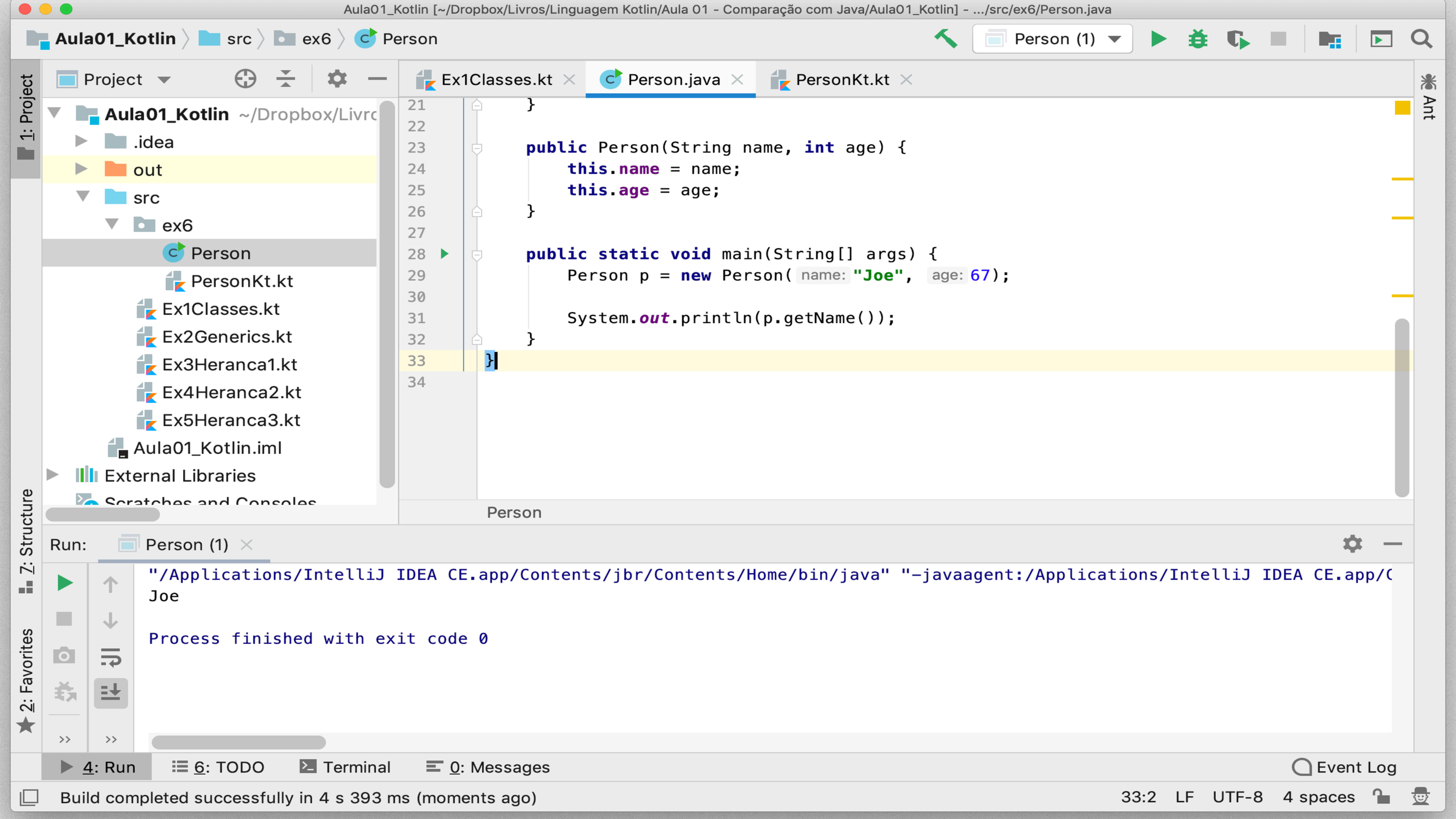
Kotlin Playground

- https://pl.kotl.in/J0X_3Z14l

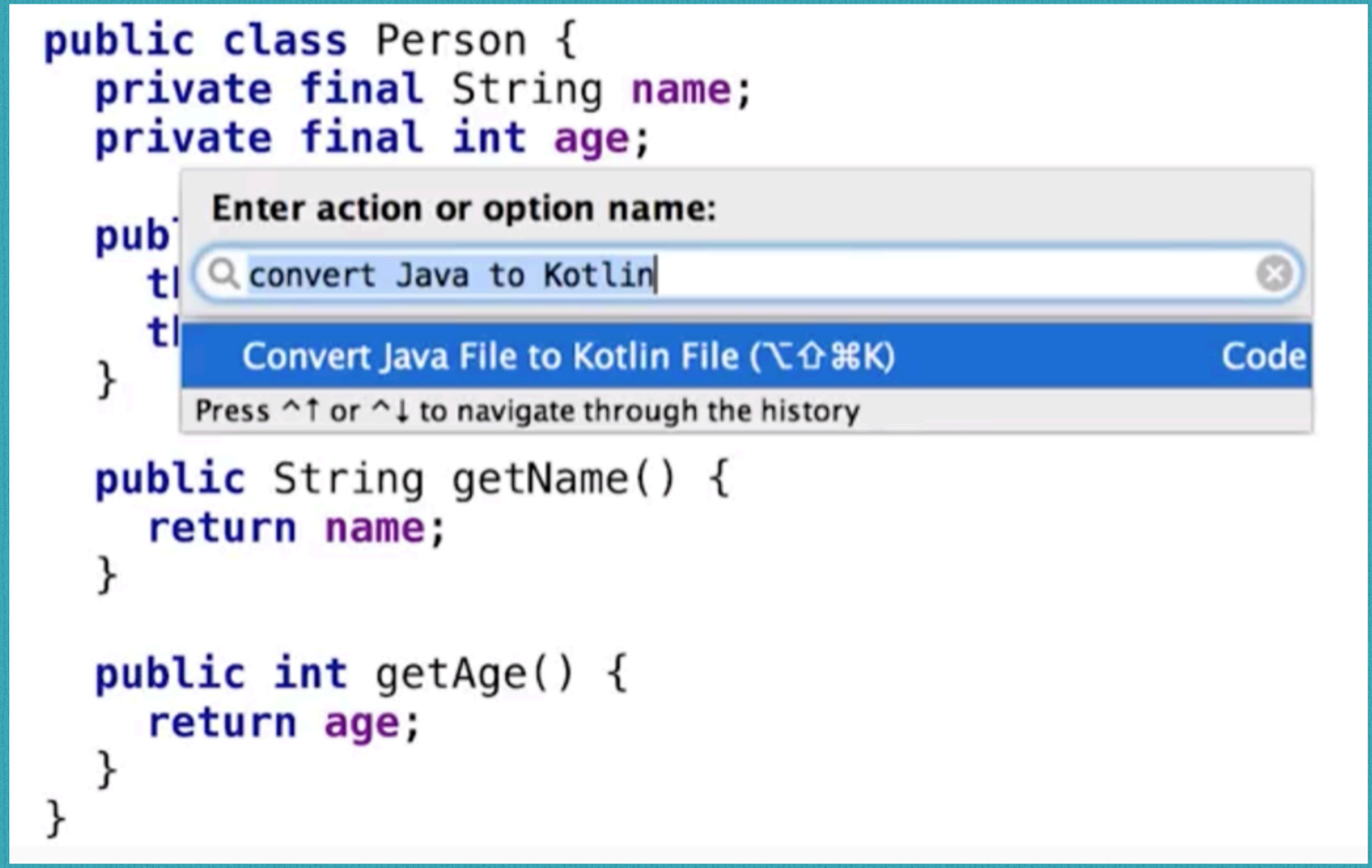


Classe Person em Java

```java
public class Person {
  private final String name;
  private final int age;

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

  public String getName() {
    return name;
  }

  public int getAge() {
    return age;
  }
}
```


Aula01_Kotlin [~/Dropbox/Livros/Linguagem Kotlin/Aula 01 - Comparação com Java/Aula01_Kotlin] - .../src/ex6/Person.java
Aula01_Kotlin
src
ex6
Person
Person (1)
Project
Aula01_Kotlin ~/Dropbox/Livro
.idea
out
src
ex6
Person
PersonKt.kt
Ex1Classes.kt
Ex2Generics.kt
Ex3Heranca1.kt
Ex4Heranca2.kt
Ex5Heranca3.kt
Aula01_Kotlin.iml
External Libraries
Scratches and Consoles
Ex1Classes.kt
Person.java
PersonKt.kt
21 }
22
23 public Person(String name, int age) {
24 this.name = name;
25 this.age = age;
26 }
27
28 public static void main(String[] args) {
29 Person p = new Person(name: "Joe", age: 67);
30
31 System.out.println(p.getName());
32 }
33 }
34
Person
Run:
Person (1)
"/Applications/IntelliJ IDEA CE.app/Contents/jbr/Contents/Home/bin/java" "-javaagent:/Applications/IntelliJ IDEA CE.app/C
Joe
Process finished with exit code 0
1: Project
7: Structure
2: Favorites
Ant
4: Run
6: TODO
Terminal
0: Messages
Event Log
Build completed successfully in 4 s 393 ms (moments ago)
33:2
LF
UTF-8
4 spaces

Podemos Converter Código Java em Kotlin com o IntelliJ



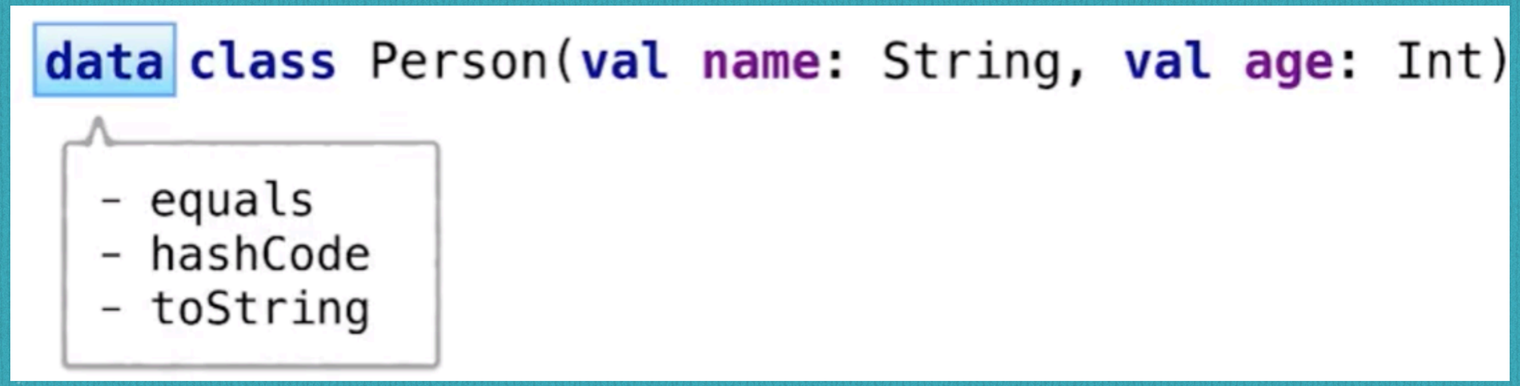
Mesma Classe em Kotlin

- Só precisamos definir os atributos:

```
class Person(val name: String, val age: Int)
```

Atributo data em Kotlin

- Adicionando a palavra data na declaração da classe, automaticamente são adicionados os métodos equals, hashCode e toString.



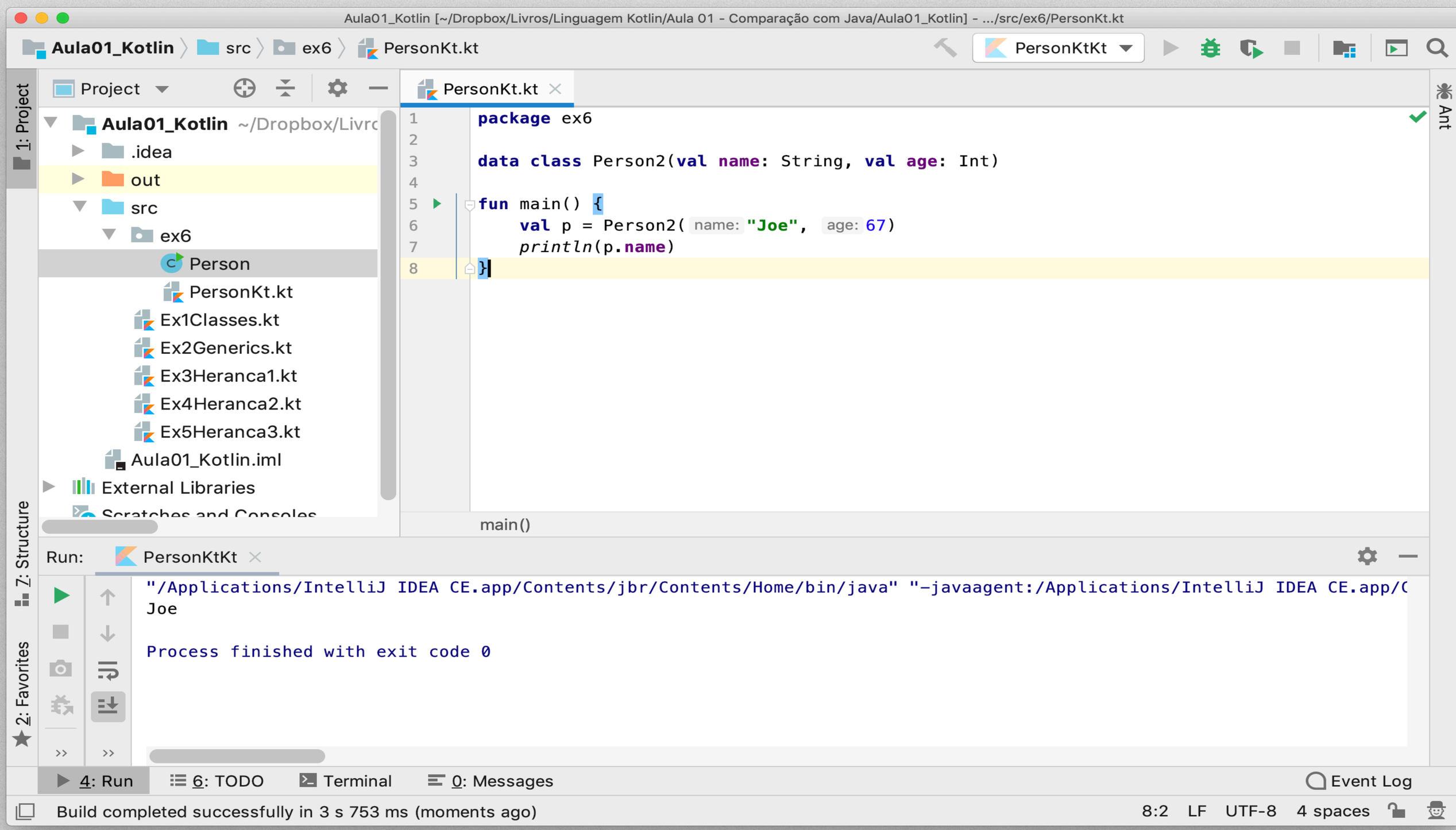
Person.java

```java
public class Person {
    private String name;
    private int age;

    public int getAge() {
        return age;
    }

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

    public String getName() {
        return name;
    }

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

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

    public static void main(String[] args) {
        Person p = new Person( name: "Joe",  age: 67);
        System.out.println(p.getName());
    }
}
```

PersonKt.kt

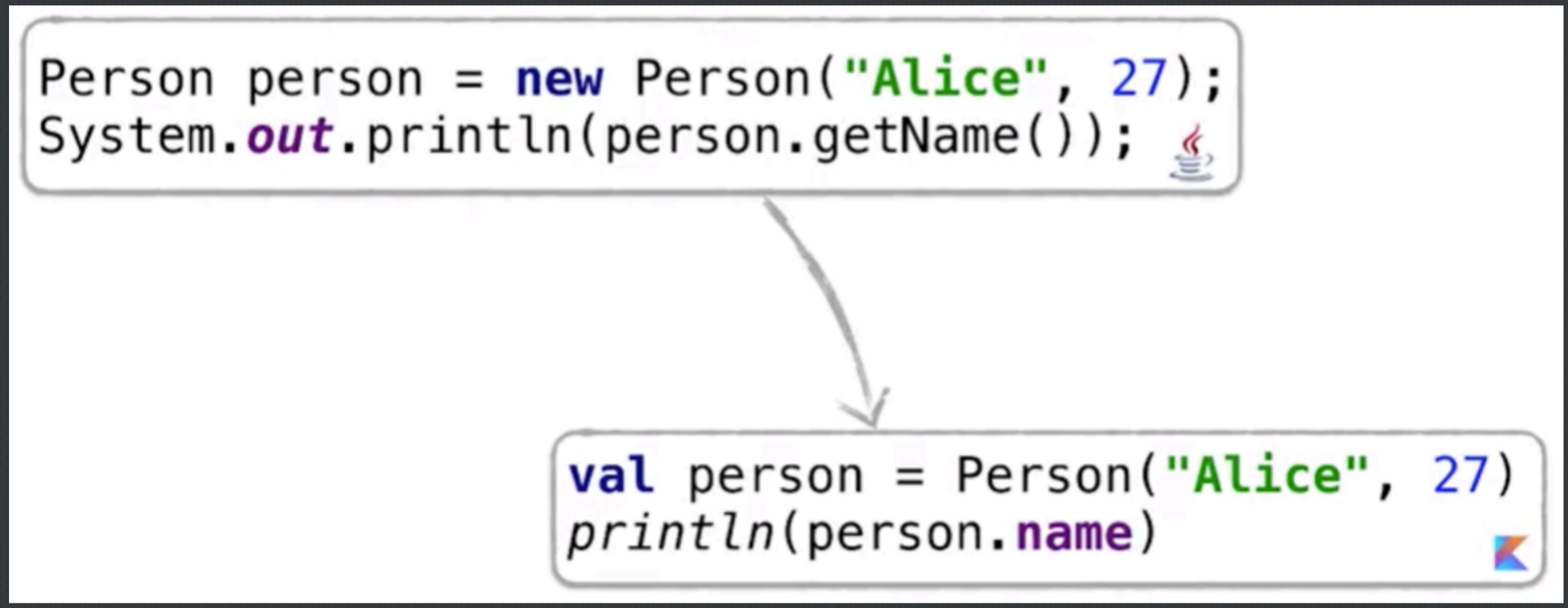
```kotlin
package ex6

data class Person2(val name: String, val age: Int)

fun main() {
    val p = Person2( name: "Joe",  age: 67)
    println(p.name)
}
```

Java x Kotlin


Aula01_Kotlin [~/Dropbox/Livros/Linguagem Kotlin/Aula 01 - Comparação com Java/Aula01_Kotlin] - .../src/ex6/PersonKt.kt
Aula01_Kotlin
src
ex6
PersonKt.kt
PersonKtKt
Project
Aula01_Kotlin ~/Dropbox/Livro
.idea
out
src
ex6
Person
PersonKt.kt
Ex1Classes.kt
Ex2Generics.kt
Ex3Heranca1.kt
Ex4Heranca2.kt
Ex5Heranca3.kt
Aula01_Kotlin.iml
External Libraries
Scratches and Consoles
PersonKt.kt
1 package ex6
2
3 data class Person2(val name: String, val age: Int)
4
5 fun main() {
6 val p = Person2(name: "Joe", age: 67)
7 println(p.name)
8 }
main()
Run:
PersonKtKt
"/Applications/IntelliJ IDEA CE.app/Contents/jbr/Contents/Home/bin/java" "-javaagent:/Applications/IntelliJ IDEA CE.app/C
Joe
Process finished with exit code 0
1: Project
7: Structure
2: Favorites
Ant
4: Run
6: TODO
Terminal
0: Messages
Event Log
Build completed successfully in 3 s 753 ms (moments ago)
8:2
LF
UTF-8
4 spaces

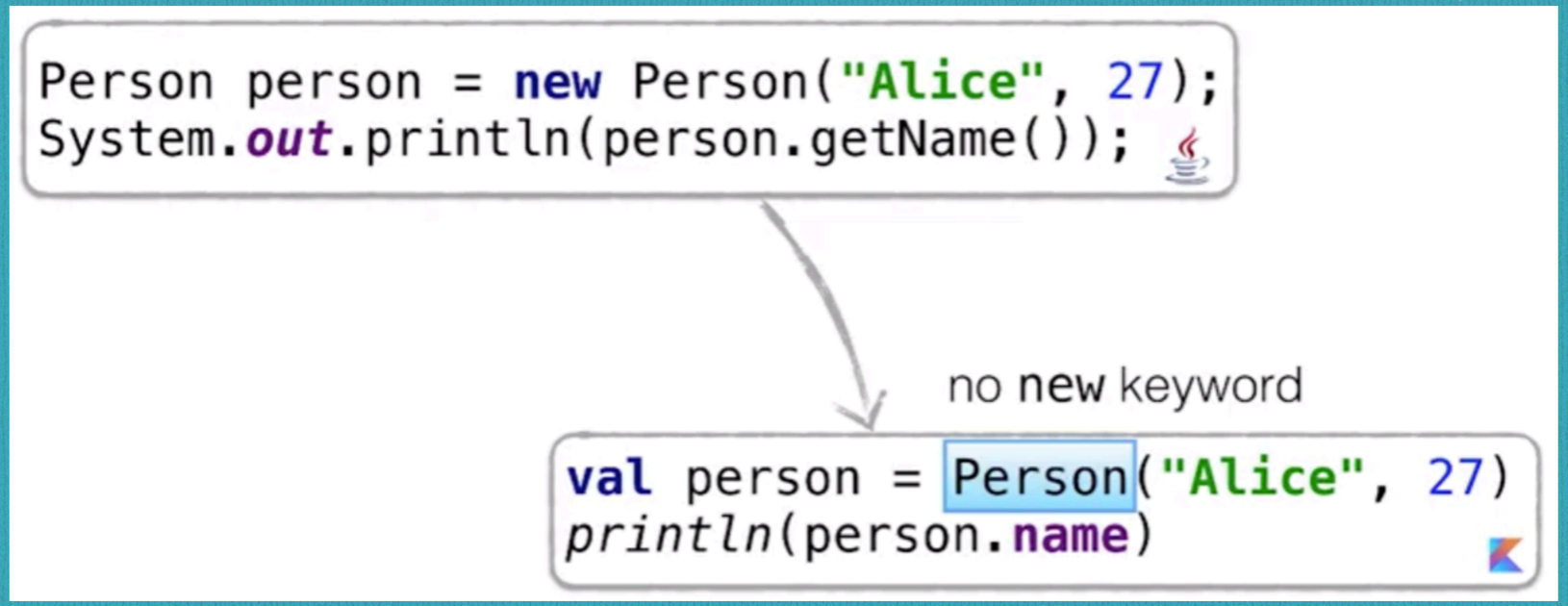
Uso de Person no main

```java
Person person = new Person("Alice", 27);
System.out.println(person.getName());
```



```kotlin
val person = Person("Alice", 27)
println(person.name)
```

Não se coloca a palavra reservada new



Função updateWeather em Java

```
public void updateWeather(int degrees) {
    String description;
    Color color;
    if (degrees < 10) {
        description = "cold";
        color = BLUE;
    } else if (degrees < 25) {
        description = "mild";
        color = ORANGE;
    } else {
        description = "hot";
        color = RED;
    }
    // ...
}
```

Função updateWeather em Kotlin

```kotlin
fun updateWeather(degrees: Int) {
    val description: String
    val color: Color
    if (degrees < 10) {
        description = "cold"
        color = BLUE
    } else if (degrees < 25) {
        description = "mild"
        color = ORANGE
    } else {
        description = "hot"
        color = RED
    }
    // ...
}
```

Java x Kotlin

UpdateWeatherJ.java

```java

public class UpdateWeatherJ {

    void updateWeather(int degrees) {
        String description;
        Color color;

        if (degrees < 10) {
            description = "cold";
            color = Color.BLUE;
        } else if (degrees < 25) {
            description = "mild";
            color = Color.ORANGE;
        } else {
            description = "hot";
            color = Color.RED;
        }
    }
}
```

UpdateWeatherK.kt

```kotlin


fun updateWeather(degrees: Int) {
    val description: String
    val color: Color

    if (degrees < 10) {
        description = "cold"
        color = Color.BLUE
    } else if (degrees < 25) {
        description = "mild"
        color = Color.ORANGE
    } else {
        description = "hot"
        color = Color.RED
    }
}
```

Podemos Agrupar em um Par

```
fun updateWeather(degrees: Int) {
    val (description: String, color: Color) =
        if (degrees < 10) {
            Pair("cold", BLUE)
        } else if (degrees < 25) {
            Pair("mild", ORANGE)
        } else {
            Pair("hot", RED)
        }
    // ...
}
```

Omitir o Tipo

```
fun updateWeather(degrees: Int) {
    val (description, color) =
        if (degrees < 10) {
            Pair("cold", BLUE)
        } else if (degrees < 25) {
            Pair("mild", ORANGE)
        } else {
            Pair("hot", RED)
        }
    // ...
}
```

Usar When e Lambda

```
fun updateWeather(degrees: Int) {
    val (description, color) = when {
        degrees < 10 -> Pair("cold", BLUE)
        degrees < 25 -> Pair("mild", ORANGE)
        else -> Pair("hot", RED)
    }
    // ...
}
```

Omitir a Instanciação de Pair

```
fun updateWeather(degrees: Int) {
    val (description, color) = when {
        degrees < 10 -> "cold" to BLUE
        degrees < 25 -> "mild" to ORANGE
        else -> "hot" to RED
    }
    // ...
}
```

Otimização da Função

```java
String description;
Color color;
if (degrees < 10) {
    description = "cold";
    color = BLUE;
} else if (degrees < 25) {
    description = "mild";
    color = ORANGE;
} else {
    description = "hot";
    color = RED;
}
```

```kotlin
val (description, colour) = when {
    degrees < 10 -> "cold" to BLUE
    degrees < 25 -> "mild" to ORANGE
    else -> "hot" to RED
}
```

Herança em Kotlin

1. As classes em Kotlin são final por padrão. Para usar ela em heranças, usa-se o modificador open
2. O mesmo ocorre com os métodos
3. Na herança, devemos usar a sintaxe : SuperClasse()
4. Podemos sobrescrever funções da superclasse com o modificador override

```
open class Dog {                        // 1
    open fun sayHello() {               // 2
        println("wow wow!")
    }
}

class Yorkshire : Dog() {               // 3
    override fun sayHello() {           // 4
        println("wif wif!")
    }
}

fun main() {
    val dog: Dog = Yorkshire()
    dog.sayHello()
}
```

Herança com Construtor Parametrizado

```
open class Tiger(val origin: String) {
    fun sayHello() {
        println("A tiger from $origin says: grrhhh!")
    }
}

class SiberianTiger : Tiger("Siberia")

fun main() {
    val tiger: Tiger = SiberianTiger()
    tiger.sayHello()
}
```

Passando Argumentos do Construtor para a Superclasse

1. O parâmetro name na classe Asiatic é passado para a superclasse Lion e o parâmetro origin tem o valor default "India" quando não recebido.

2. Criamos um objeto da classe Asiatic com o name Rufo. A chamada invoca o construtor da classe Lion e passa os argumentos "Rufo" e "India"

```
open class Lion(val name: String, val origin: String) {
    fun sayHello() {
        println("$name, the lion from $origin says: graoh!")
    }
}

class Asiatic(name: String) : Lion(name = name, origin = "India") // 1

fun main() {
    val lion: Lion = Asiatic("Rufo")                              // 2
    lion.sayHello()
}
```