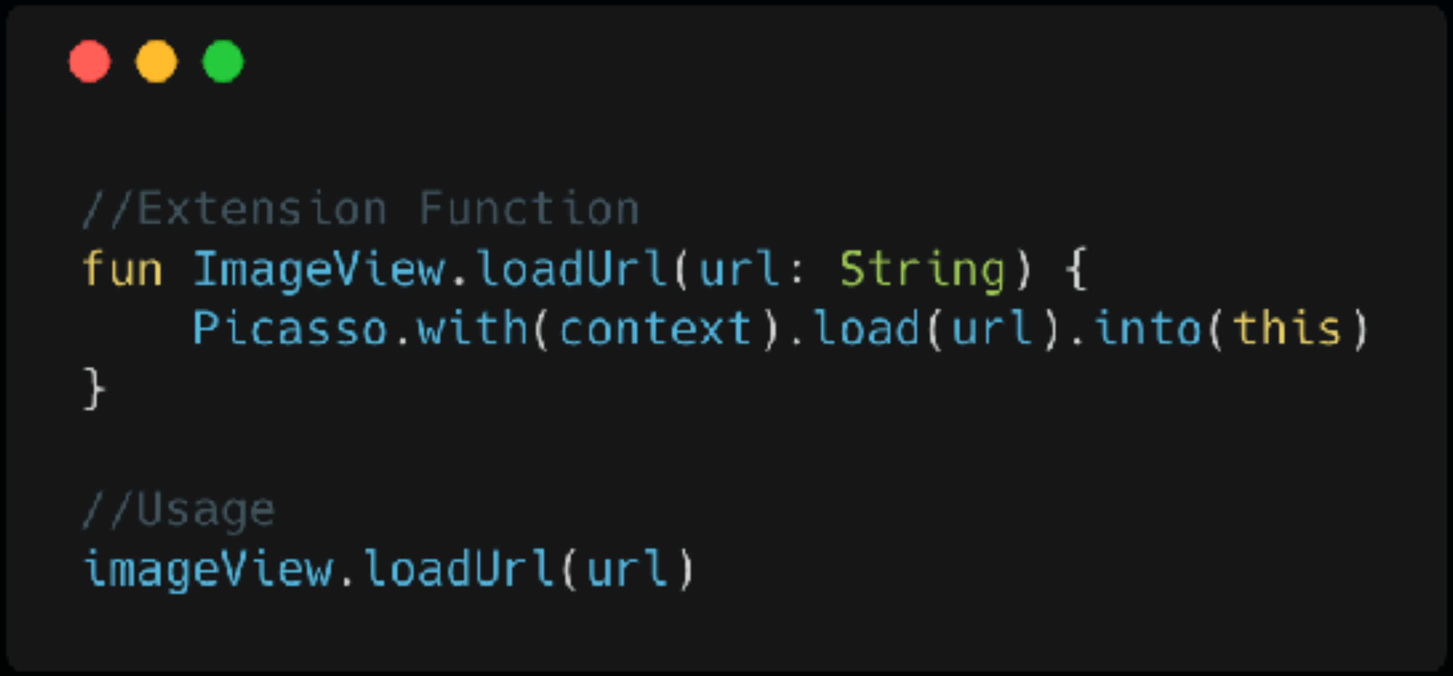
Kotlin-III

Extensions

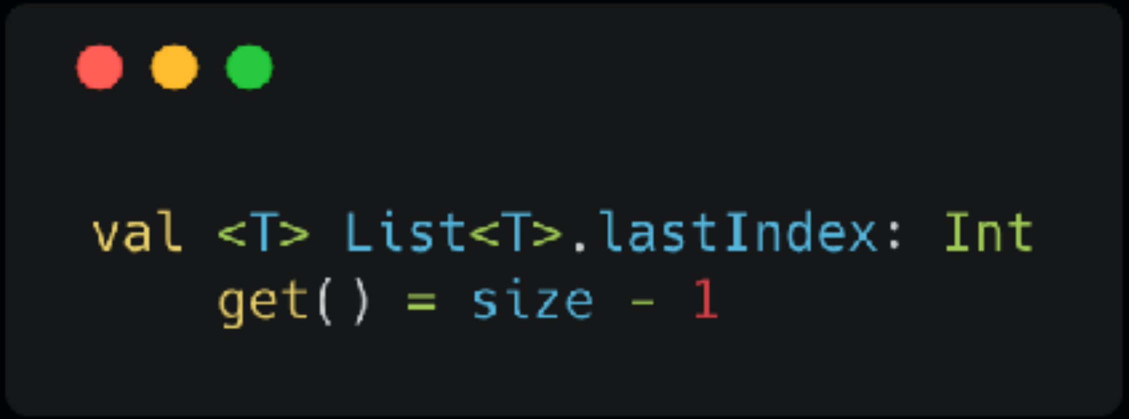
- Ability to inherit the class with new functionality without explicitly inheriting it.
- No need of extra Decorator design patterns.
- Two type of extensions: extension properties and extension functions.
- Extension function are resolved **statically**.

Extension Functions

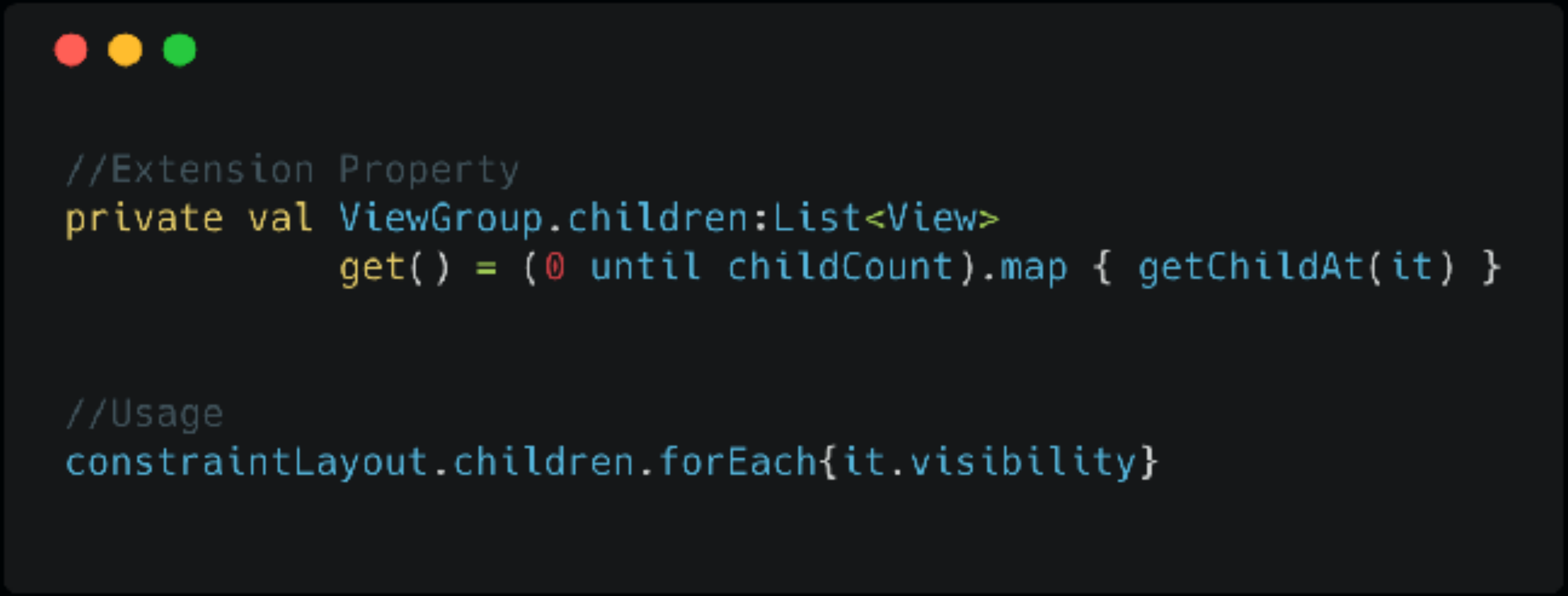


```
//Extension Function
fun ImageView.loadUrl(url: String) {
    Picasso.with(context).load(url).into(this)
}

//Usage
imageView.loadUrl(url)
```

Extension Properties



```
val <T> List<T>.lastIndex: Int
    get() = size - 1
```



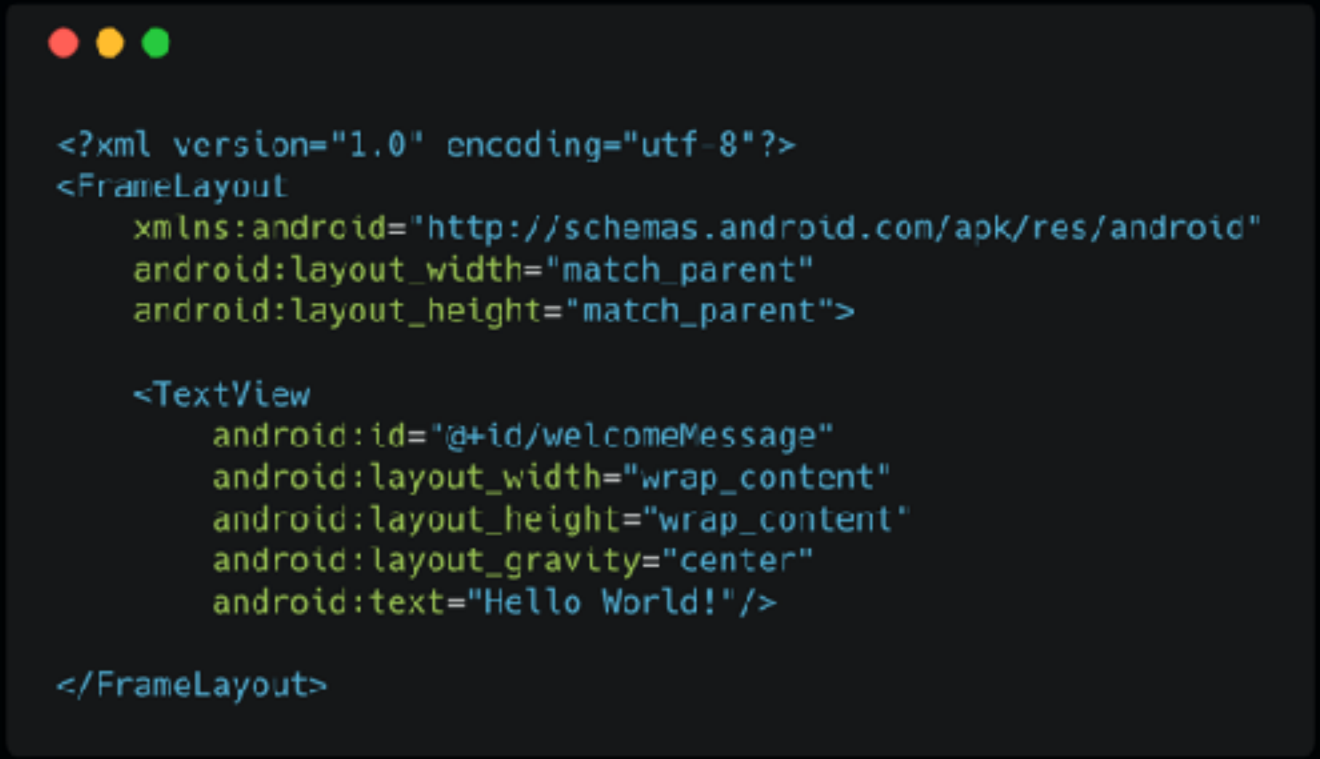
```
//Extension Property
private val ViewGroup.children:List<View>
            get() = (0 until childCount).map { getChildAt(it) }


//Usage
constraintLayout.children.forEach{it.visibility}
```

Example 2

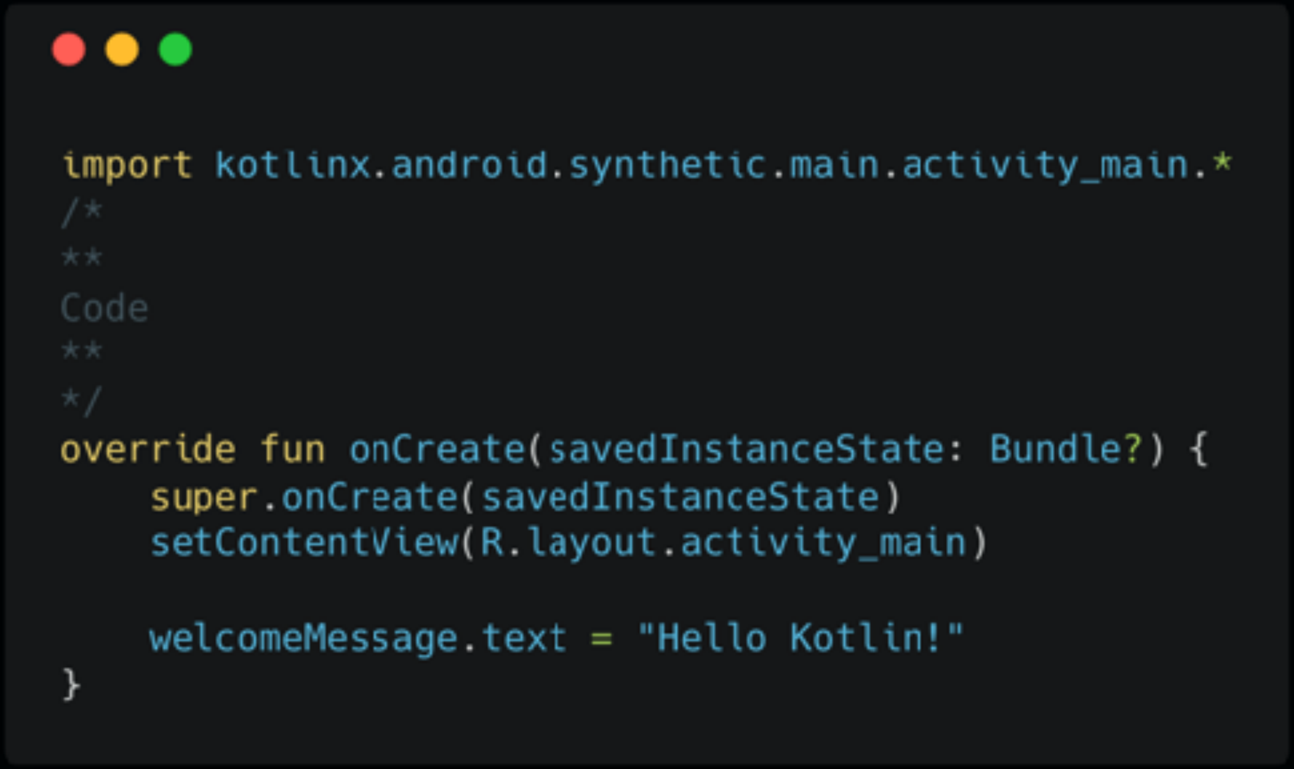
Get Ready

Say GoodBye to findViewById

- No findViewById
- No ButterKnife
- It will allow you to access views in XML, just as they were the properties.
- Let us see the code:

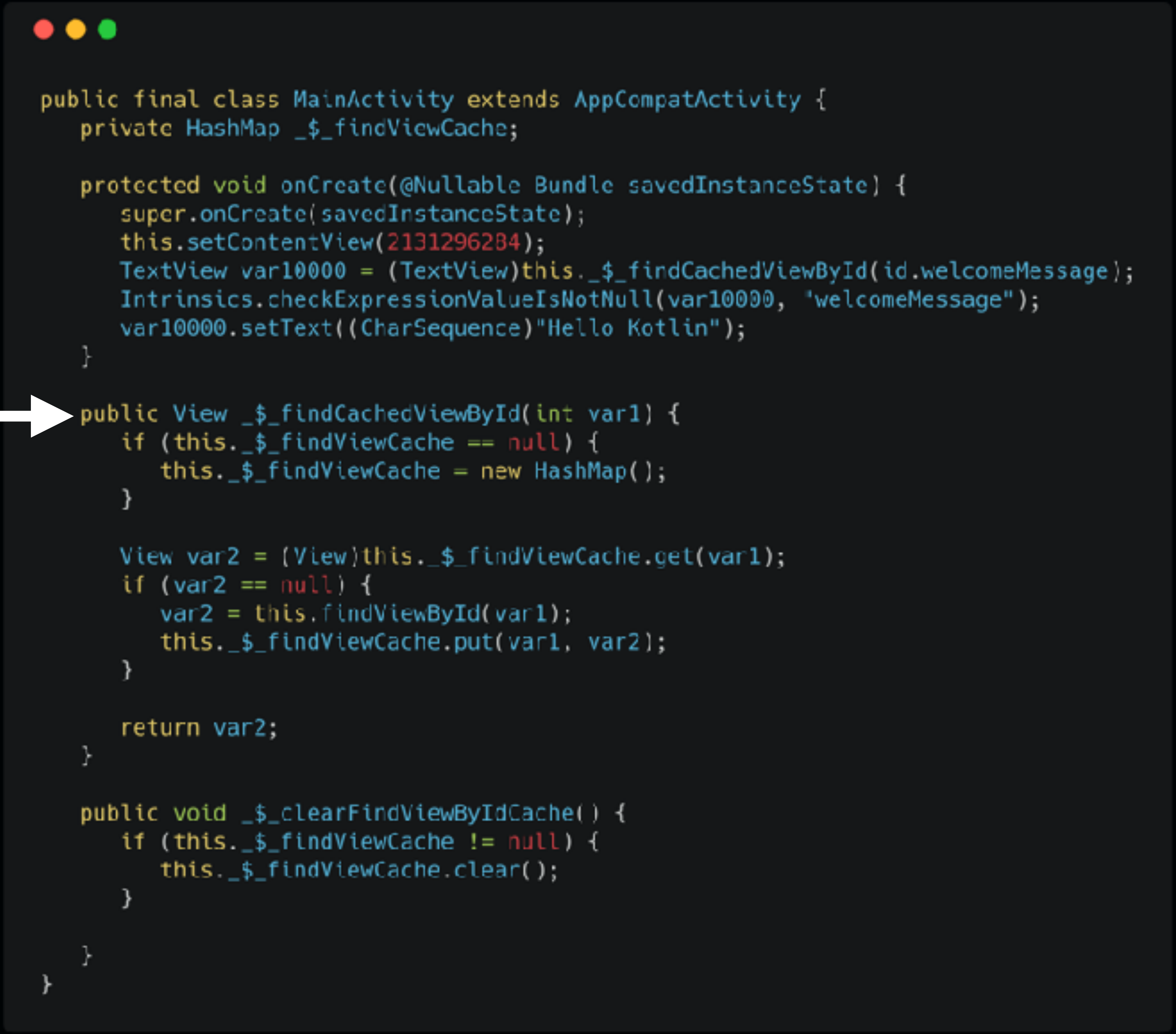


```xml
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <TextView
        android:id="@+id/welcomeMessage"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:text="Hello World!"/>

</FrameLayout>
```



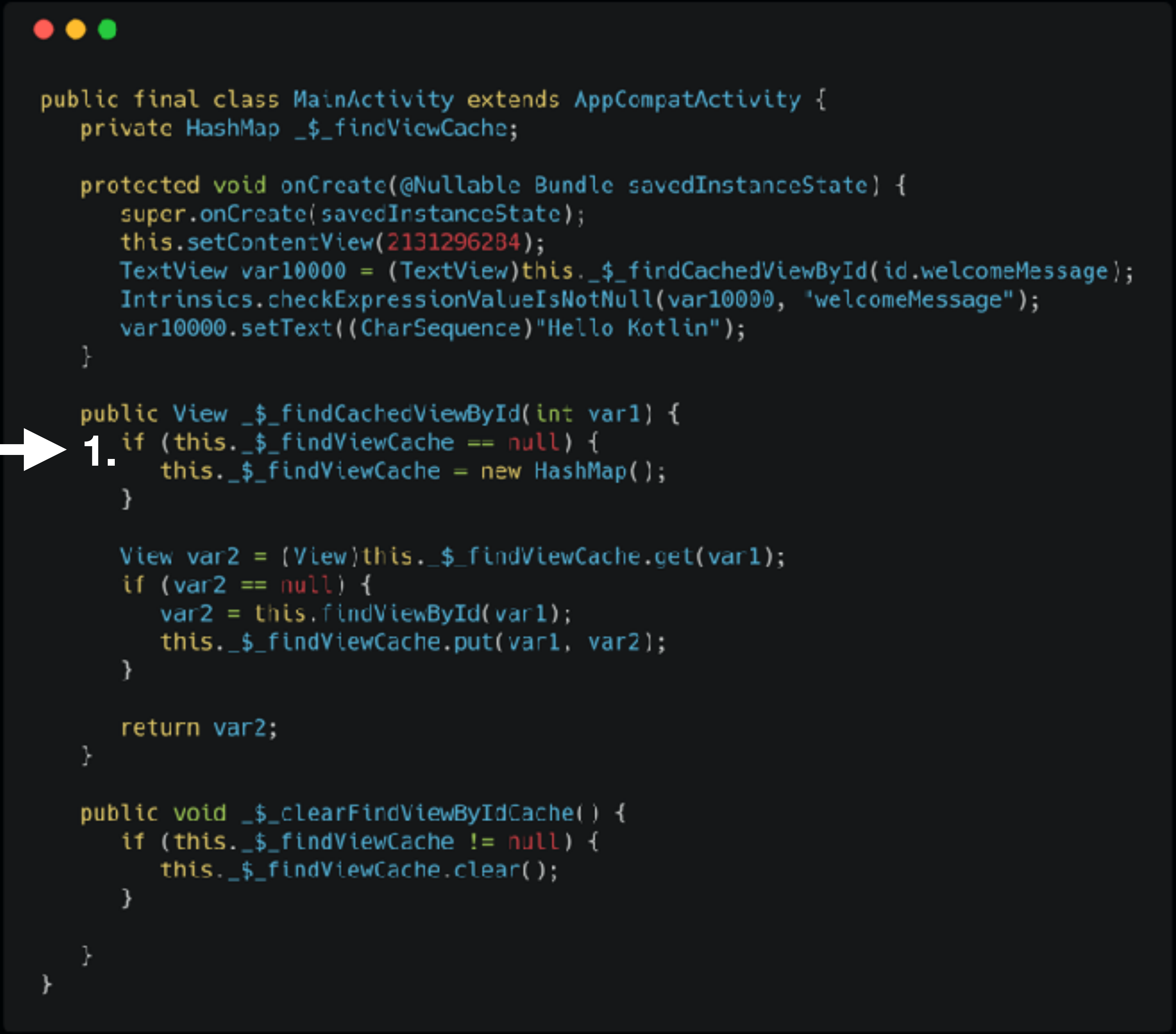
```kotlin
import kotlinx.android.synthetic.main.activity_main.*
/*
**
Code
**
*/
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_main)

    welcomeMessage.text = "Hello Kotlin!"
}
```

But How?



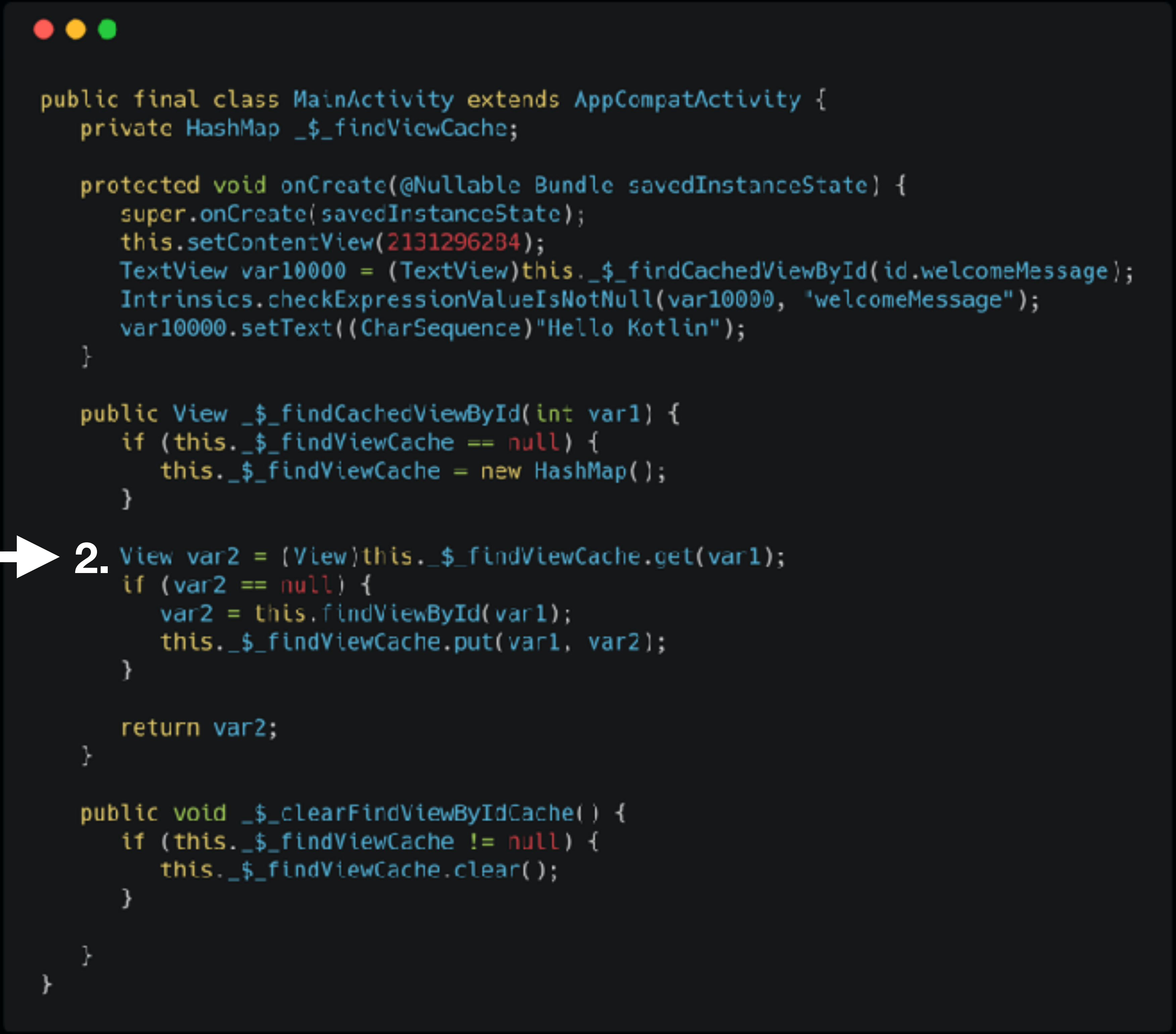
```
public final class MainActivity extends AppCompatActivity {
   private HashMap _$_findViewCache;

   protected void onCreate(@Nullable Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      this.setContentView(2131296284);
      TextView var10000 = (TextView)this._$_findCachedViewById(id.welcomeMessage);
      Intrinsics.checkExpressionValueIsNotNull(var10000, "welcomeMessage");
      var10000.setText((CharSequence)"Hello Kotlin");
   }

   public View _$_findCachedViewById(int var1) {
      if (this._$_findViewCache == null) {
         this._$_findViewCache = new HashMap();
      }

      View var2 = (View)this._$_findViewCache.get(var1);
      if (var2 == null) {
         var2 = this.findViewById(var1);
         this._$_findViewCache.put(var1, var2);
      }

      return var2;
   }

   public void _$_clearFindViewByIdCache() {
      if (this._$_findViewCache != null) {
         this._$_findViewCache.clear();
      }

   }
}
```

But How?



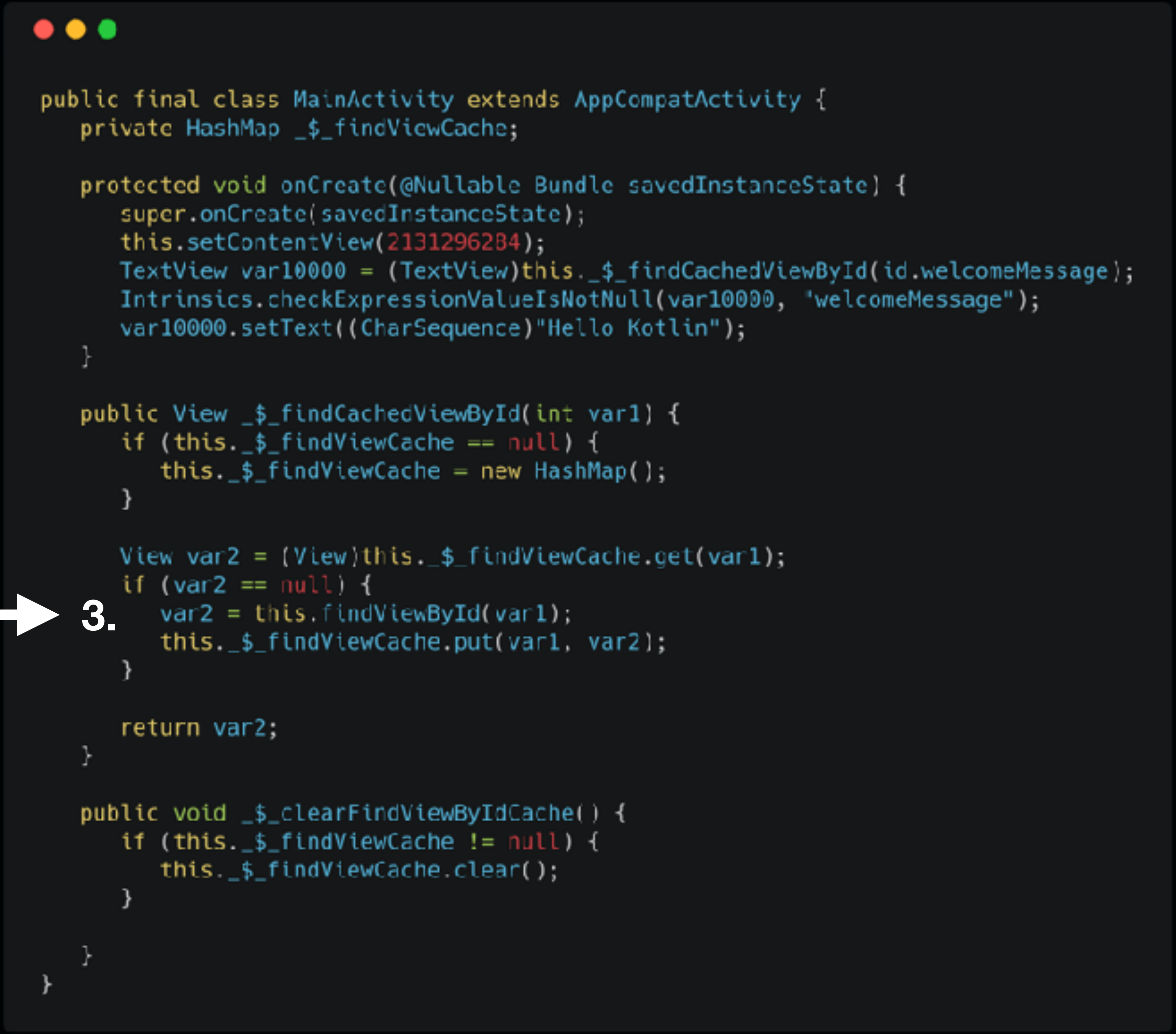
```
public final class MainActivity extends AppCompatActivity {
   private HashMap _$_findViewCache;

   protected void onCreate(@Nullable Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      this.setContentView(2131296284);
      TextView var10000 = (TextView)this._$_findCachedViewById(id.welcomeMessage);
      Intrinsics.checkExpressionValueIsNotNull(var10000, "welcomeMessage");
      var10000.setText((CharSequence)"Hello Kotlin");
   }

   public View _$_findCachedViewById(int var1) {
1.    if (this._$_findViewCache == null) {
         this._$_findViewCache = new HashMap();
      }

      View var2 = (View)this._$_findViewCache.get(var1);
      if (var2 == null) {
         var2 = this.findViewById(var1);
         this._$_findViewCache.put(var1, var2);
      }

      return var2;
   }

   public void _$_clearFindViewByIdCache() {
      if (this._$_findViewCache != null) {
         this._$_findViewCache.clear();
      }

   }
}
```

But How?



```
public final class MainActivity extends AppCompatActivity {
   private HashMap _$_findViewCache;

   protected void onCreate(@Nullable Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      this.setContentView(2131296284);
      TextView var10000 = (TextView)this._$_findCachedViewById(id.welcomeMessage);
      Intrinsics.checkExpressionValueIsNotNull(var10000, "welcomeMessage");
      var10000.setText((CharSequence)"Hello Kotlin");
   }

   public View _$_findCachedViewById(int var1) {
      if (this._$_findViewCache == null) {
         this._$_findViewCache = new HashMap();
      }

   2. View var2 = (View)this._$_findViewCache.get(var1);
      if (var2 == null) {
         var2 = this.findViewById(var1);
         this._$_findViewCache.put(var1, var2);
      }

      return var2;
   }

   public void _$_clearFindViewByIdCache() {
      if (this._$_findViewCache != null) {
         this._$_findViewCache.clear();
      }

   }
}
```

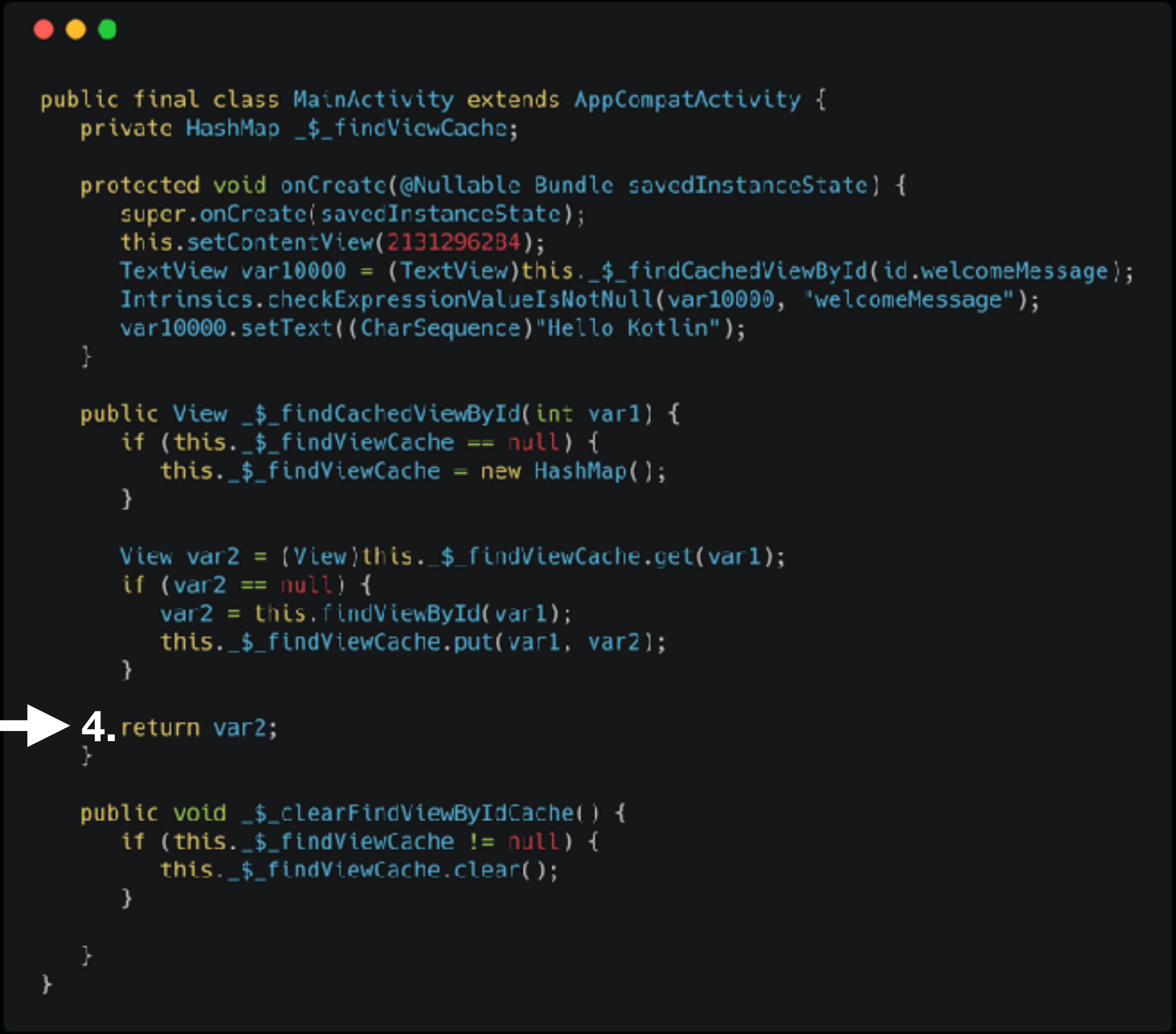
But How?



```
public final class MainActivity extends AppCompatActivity {
   private HashMap _$_findViewCache;

   protected void onCreate(@Nullable Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      this.setContentView(2131296284);
      TextView var10000 = (TextView)this._$_findCachedViewById(id.welcomeMessage);
      Intrinsics.checkExpressionValueIsNotNull(var10000, "welcomeMessage");
      var10000.setText((CharSequence)"Hello Kotlin");
   }

   public View _$_findCachedViewById(int var1) {
      if (this._$_findViewCache == null) {
         this._$_findViewCache = new HashMap();
      }

      View var2 = (View)this._$_findViewCache.get(var1);
      if (var2 == null) {
         var2 = this.findViewById(var1);
         this._$_findViewCache.put(var1, var2);
      }

      return var2;
   }

   public void _$_clearFindViewByIdCache() {
      if (this._$_findViewCache != null) {
         this._$_findViewCache.clear();
      }

   }
}
```

3.

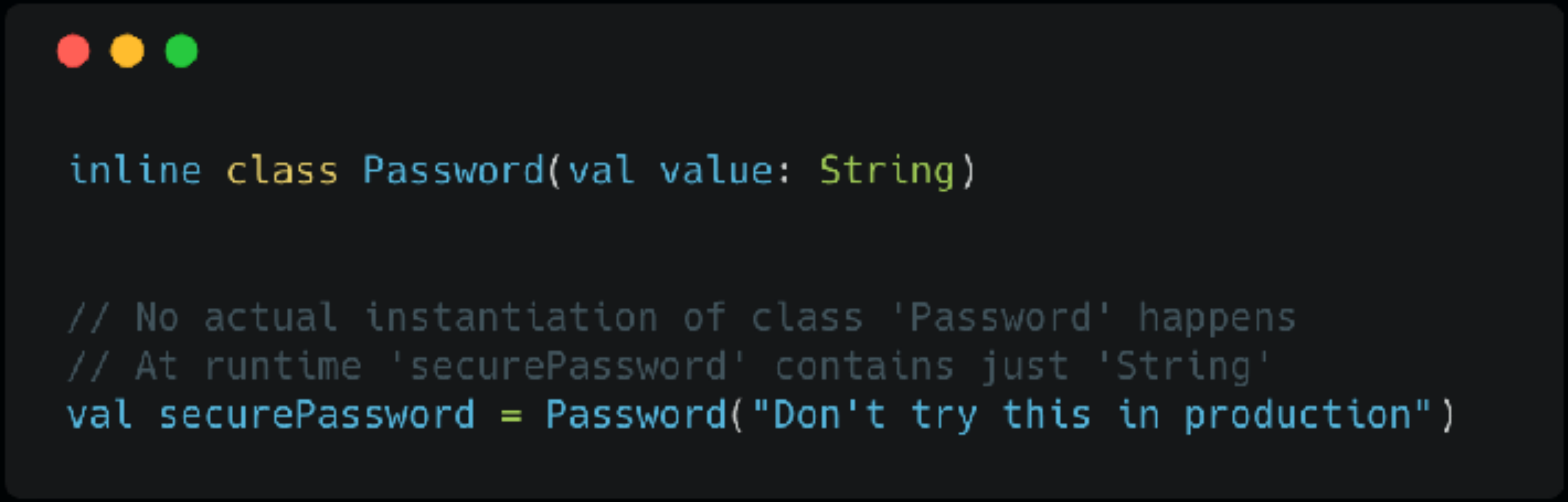
But How?



```
public final class MainActivity extends AppCompatActivity {
   private HashMap _$_findViewCache;

   protected void onCreate(@Nullable Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      this.setContentView(2131296284);
      TextView var10000 = (TextView)this._$_findCachedViewById(id.welcomeMessage);
      Intrinsics.checkExpressionValueIsNotNull(var10000, "welcomeMessage");
      var10000.setText((CharSequence)"Hello Kotlin");
   }

   public View _$_findCachedViewById(int var1) {
      if (this._$_findViewCache == null) {
         this._$_findViewCache = new HashMap();
      }

      View var2 = (View)this._$_findViewCache.get(var1);
      if (var2 == null) {
         var2 = this.findViewById(var1);
         this._$_findViewCache.put(var1, var2);
      }

   4. return var2;
   }

   public void _$_clearFindViewByIdCache() {
      if (this._$_findViewCache != null) {
         this._$_findViewCache.clear();
      }

   }
}
```

Inline

- Creating wrappers is not cool every time if you think as compiler.
- Runtime overhead due to heap allocations
- Inline keyword introduced
- Runtime replacement to the wrappers



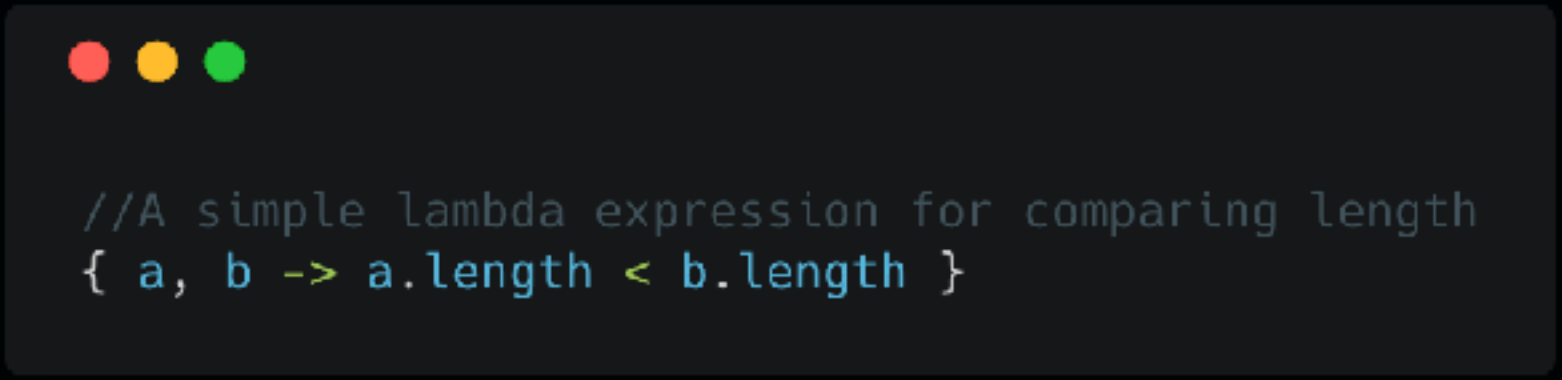
```kotlin
inline class Password(val value: String)


// No actual instantiation of class 'Password' happens
// At runtime 'securePassword' contains just 'String'
val securePassword = Password("Don't try this in production")
```

High Order Functions

- Kotlin : Functions are first class
- Functions can be stored in data structures, variables, etc.
- Higher Order Function: Function that take functions as a parameter, or returns a function.
- Example of **function type**: ( ) -> String , (String) -> String

Lambda expressions

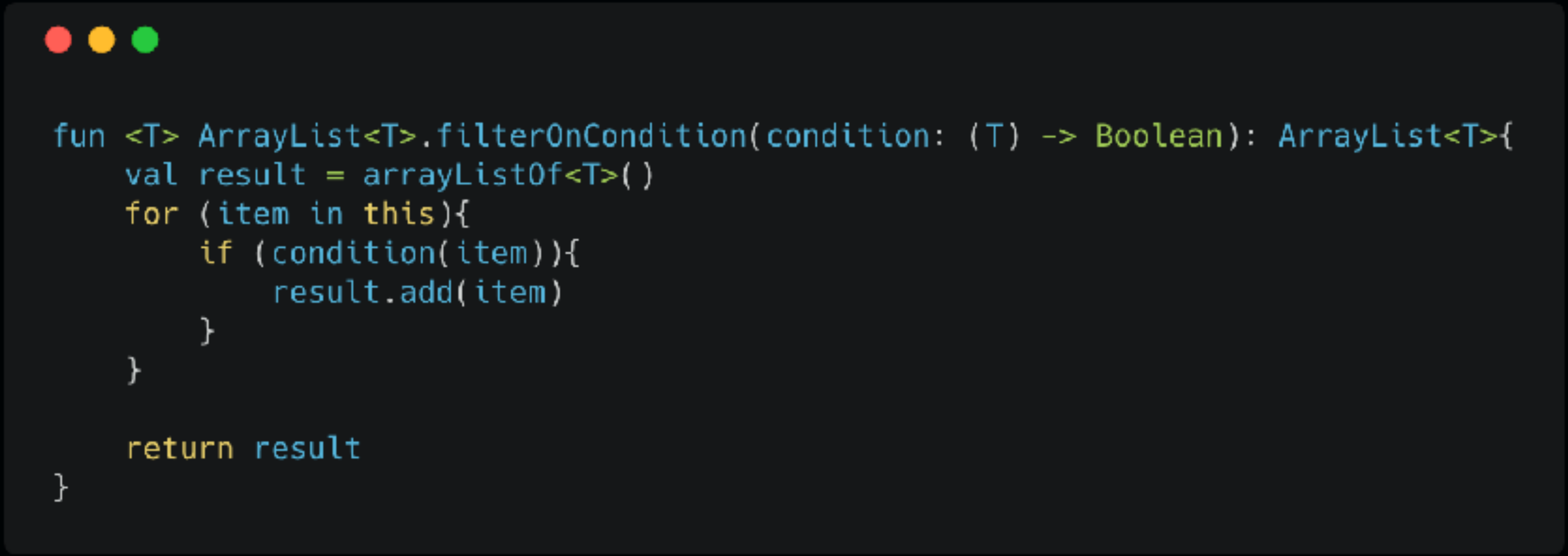
- Also called function literals.
- Functions that are not declared
- Immediately passed as an expression
- **Example:**



```
//A simple lambda expression for comparing length
{ a, b -> a.length < b.length }
```

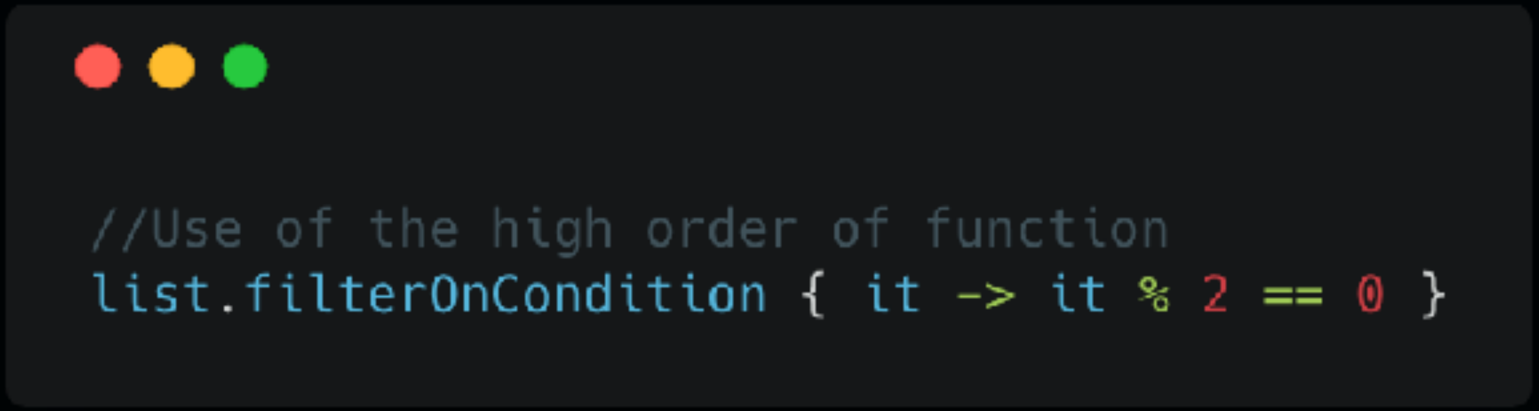
Examples of High Order Functions

- **Filter Conditions made easy**

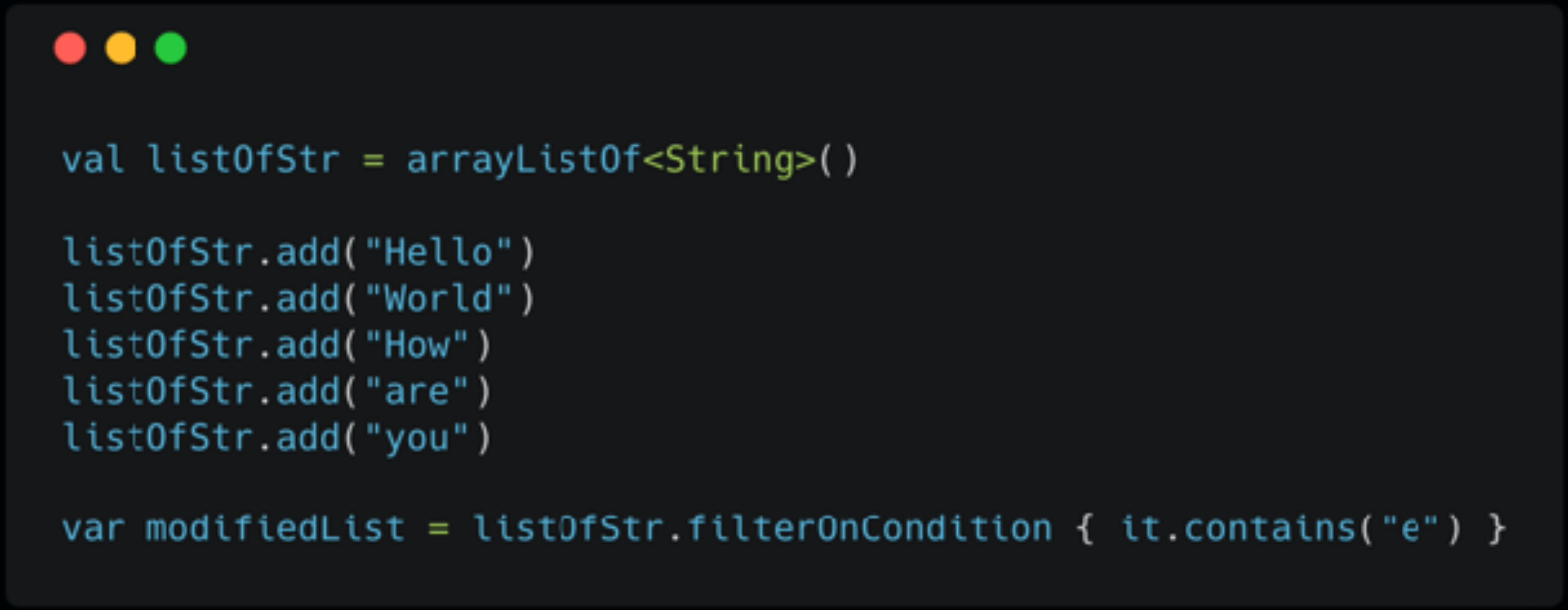


```kotlin
fun <T> ArrayList<T>.filterOnCondition(condition: (T) -> Boolean): ArrayList<T>{
    val result = arrayListOf<T>()
    for (item in this){
        if (condition(item)){
            result.add(item)
        }
    }

    return result
}
```

- **Usage of filterOnCondition**



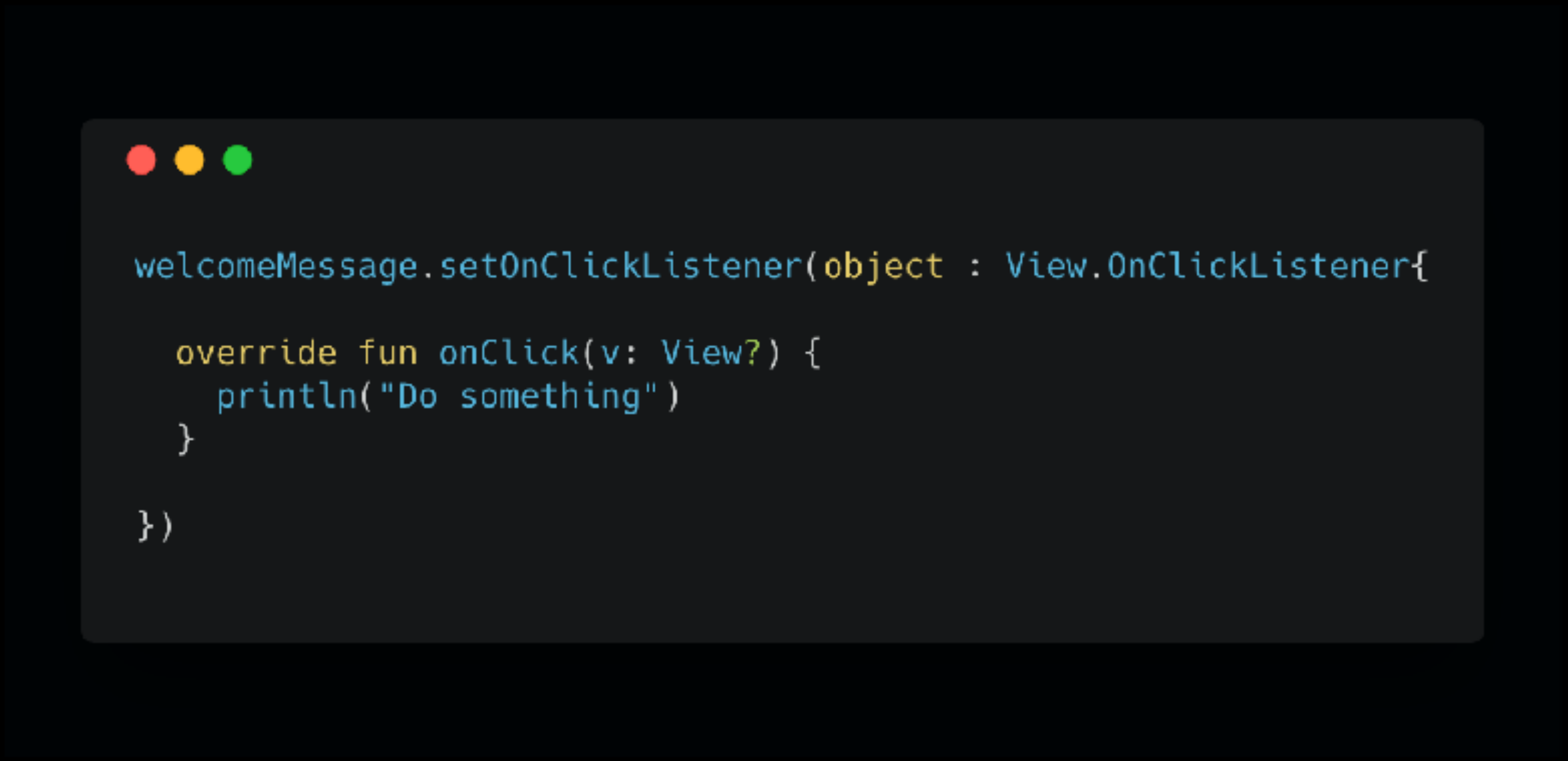
```kotlin
//Use of the high order of function
list.filterOnCondition { it -> it % 2 == 0 }
```



```kotlin
val listOfStr = arrayListOf<String>()

listOfStr.add("Hello")
listOfStr.add("World")
listOfStr.add("How")
listOfStr.add("are")
listOfStr.add("you")

var modifiedList = listOfStr.filterOnCondition { it.contains("e") }
```

Object

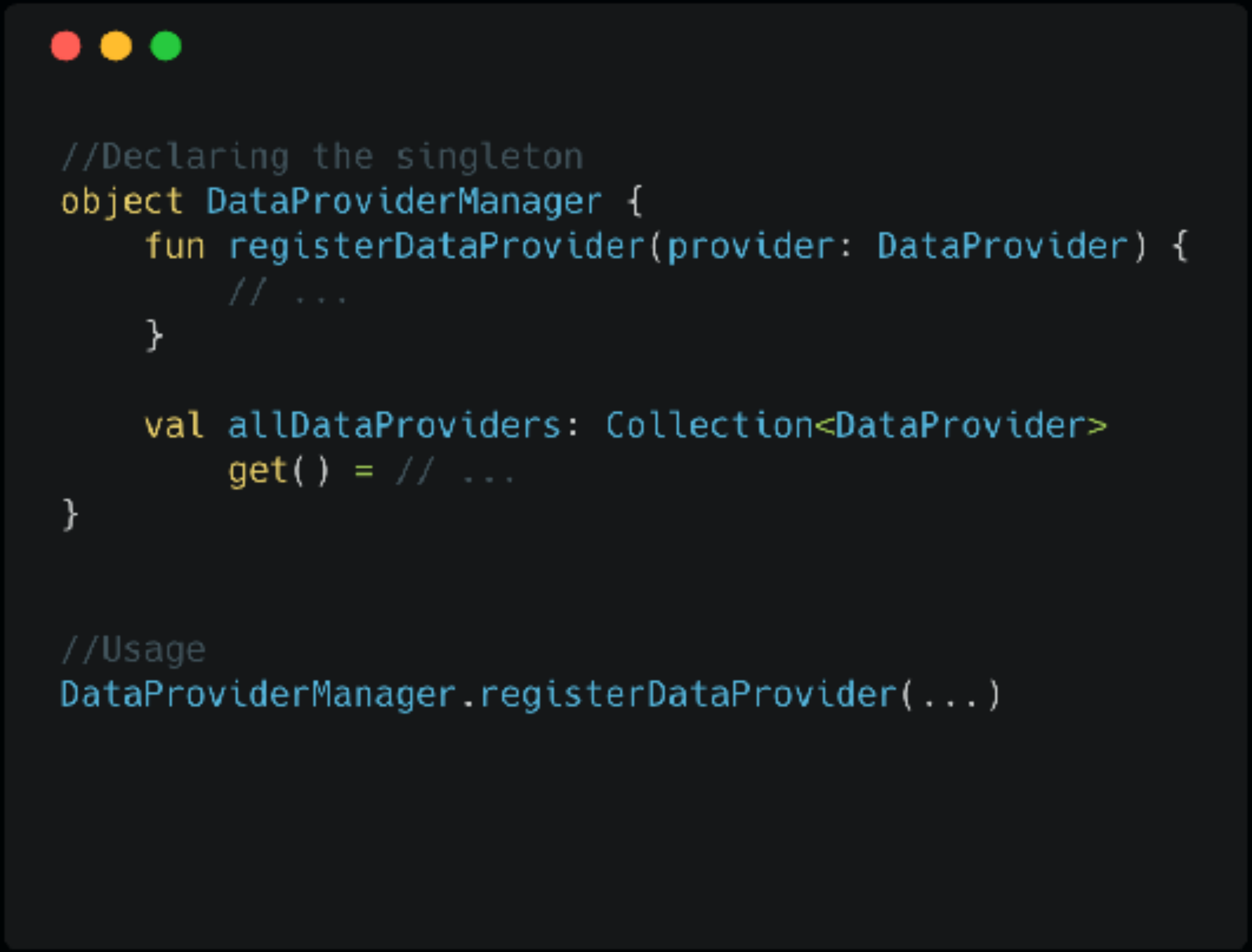
- Make a subclass without explicitly declaring a inner class that is through anonymous class.



```kotlin
welcomeMessage.setOnClickListener(object : View.OnClickListener{

  override fun onClick(v: View?) {
    println("Do something")
  }

})
```

Singletons?

- Easy with the keyword **object**

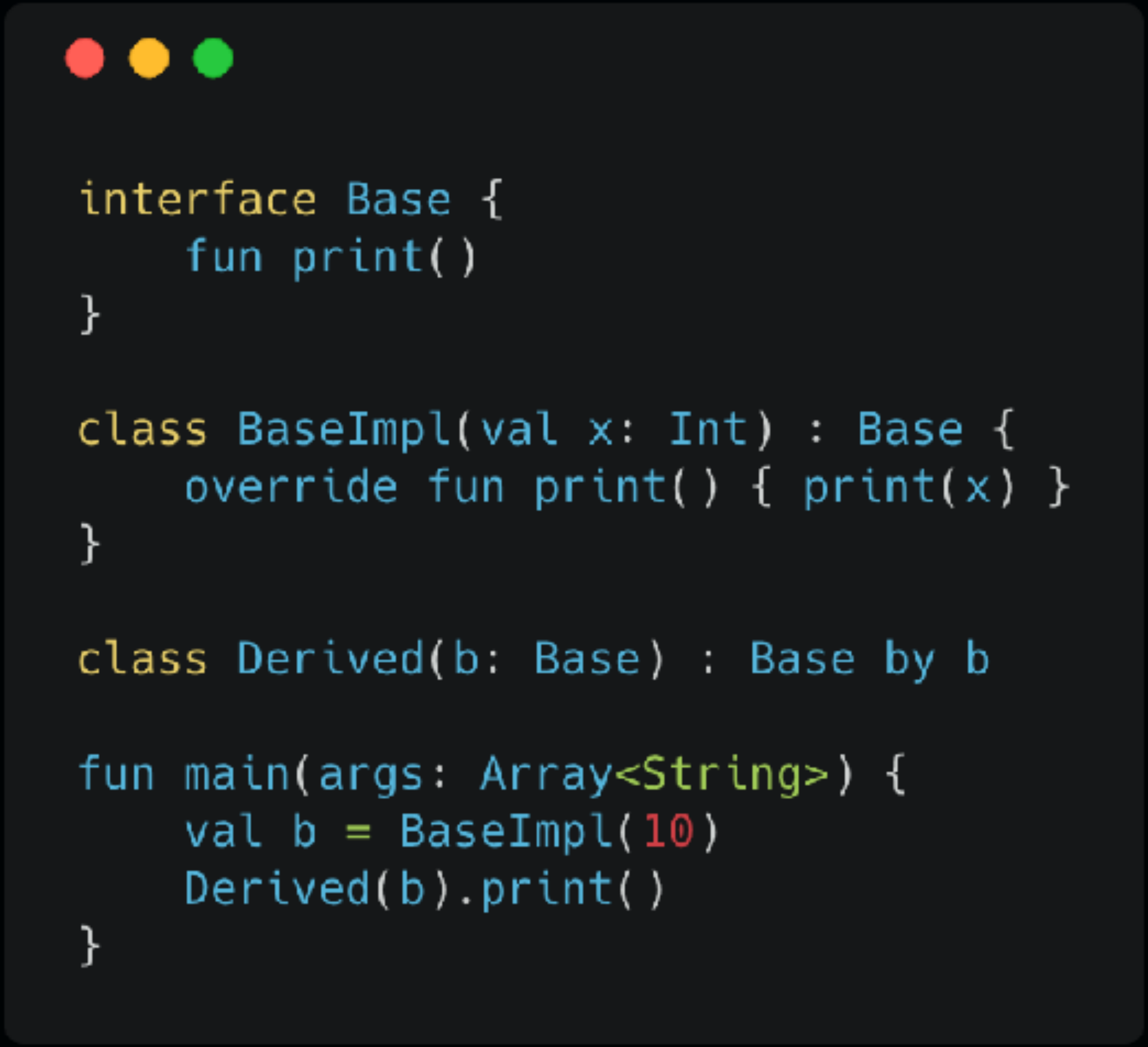


```kotlin
//Declaring the singleton
object DataProviderManager {
    fun registerDataProvider(provider: DataProvider) {
        // ...
    }

    val allDataProviders: Collection<DataProvider>
        get() = // ...
}


//Usage
DataProviderManager.registerDataProvider(...)
```

Delegation

- Inheritance is not good every time
- Feeling of asking or delegating a request to another entity mostly class in our context
- It can be viewed as relationship between objects where one object forward certain method calls to another object called its delegate.
- It does not forces you to accept all the methods from super class which happens in inheritance.

Example of Delegation



```kotlin
interface Base {
    fun print()
}

class BaseImpl(val x: Int) : Base {
    override fun print() { print(x) }
}

class Derived(b: Base) : Base by b

fun main(args: Array<String>) {
    val b = BaseImpl(10)
    Derived(b).print()
}
```

Thank You🙋‍♂️