Sneaking inside Kotlin features

Chandra Sekhar

GDG Kolkata

WTM :Kolkata:

GDG Cloud Kolkata

Sneaking inside Kotlin features

GDG Kolkata

WTM :Kolkata:

GDG Cloud Kolkata

1. Variables

1.1 lateinit properties

lateinit properties

```
lateinit var user: User
```

lateinit properties

```
public final User user;

public final User getUser() {
    if (user == null) {
        throwUninitializedPropertyAccessException("user");
    }
    return user;
}

public final void setUser(User newUser) {
    checkParameterIsNotNull(newUser, "<set-?>");
    user = newUser;
}
```

@iChanSek

bit.ly/chansecode

lateinit properties

```
public final User getUser() {
    if (user == null) {
        throwUninitializedPropertyAccessException("user");
    }
    return user;
}

public final void setUser(User newUser) {
    checkParameterIsNotNull(newUser, "<set-?>");
```

@iChanSek

bit.ly/chansecode

lateinit properties

```
public final User getUser() {
    if (user == null) {
        throwUninitializedPropertyAccessException("user");
    }
    return user;
}

public final void setUser(User newUser) {
    checkParameterIsNotNull(newUser, "<set-?>");
    user = newUser;
}
```

@iChanSek

bit.ly/chansecode

1.2 Delegated Properties

Delegated Properties

```
class DelegatedProperties {
    val x by lazy { 10 }
}
```


@iChanSek


bit.ly/chansecode

Delegated Properties

```
public final class DelegatedProperties {
  static final KProperty[] props = new KProperty[]{ ... }
  private final Lazy xDelegate;

  public final int getX() {
    return ((Number)xDelegate.getValue()).intValue();
  }

  public DelegatedProperties() {
    this.xDelegate = LazyKt.lazy((Function0)2.INSTANCE);
  }
}
```

@iChanSek

bit.ly/chansecode

Delegated Properties

```
public final class DelegatedProperties {
  static final KProperty[] props = new KProperty[]{ ... }
  private final Lazy xDelegate;

  public final int getX() {
    return ((Number)xDelegate.getValue()).intValue();
  }

  public DelegatedProperties() {
    this.xDelegate = LazyKt.lazy((Function0)2.INSTANCE);
  }
}
```

@iChanSek

bit.ly/chansecode

Delegated Properties

```
public final class DelegatedProperties {
   static final KProperty[] props = new KProperty[]{ ... }
  private final Lazy xDelegate;

  public final int getX() {
    return ((Number)xDelegate.getValue()).intValue();
  }

  public DelegatedProperties() {
    this.xDelegate = LazyKt.lazy((Function0)2.INSTANCE);
  }
}
```

@iChanSek

bit.ly/chansecode

Delegated Properties

```
public final class DelegatedProperties {
  static final KProperty[] props = new KProperty[]{ ... }
    private final Lazy xDelegate;

  public final int getX() {
    return ((Number)xDelegate.getValue()).intValue();
  }

    public DelegatedProperties() {
      this.xDelegate = LazyKt.lazy((Function0)2.INSTANCE);
    }
}
```

@iChanSek

bit.ly/chansecode

Delegated Properties

```
public final class DelegatedProperties {
  static final KProperty[] props = new KProperty[]{ ... }
    private final Lazy xDelegate;

  public final int getX() {
    return ((Number)xDelegate.getValue()).intValue();
  }

    public DelegatedProperties() {
       this.xDelegate = LazyKt.lazy(() -> 10);
    }
}
```

@iChanSek

bit.ly/chansecode

Delegated Properties

```
public final class DelegatedProperties {
  static final KProperty[] props = new KProperty[]{ ... }
  private final Lazy xDelegate;

    public final int getX() {
       return ((Number)xDelegate.getValue()).intValue();
    }

  public DelegatedProperties() {
    this.xDelegate = LazyKt.lazy((Function0)2.INSTANCE);
  }
}
```

@iChanSek

bit.ly/chansecode

Delegated Properties

```
public final int getX() {
  return ((Number)xDelegate.getValue()).intValue();
}
```

```
fun main() {
    val delegatedProp = DelegatedProperties()
    println(delegatedProp.x)
    println(delegatedProp.x)
}
```

@iChanSek

bit.ly/chansecode

Delegated Properties

```
public final int getX() {
  return ((Number)xDelegate.getValue()).intValue();
}
```

```
class SynchronizedLazyImpl<out T>(initializer: () -> T, lock: Any? = null) : Lazy<T> {
    private var initializer: (() -> T)? = initializer
    @Volatile private var _value: Any? = UNINITIALIZED_VALUE
    private val lock = lock ?: this

    override val value: T
        get() {
            if (_value !== UNINITIALIZED_VALUE) {
                return _value as T
            }

            return synchronized(lock) {
                ...
            }
        }
}
```

```
fun main() {
    val delegatedProp = DelegatedProperties()
    println(delegatedProp.x)
    println(delegatedProp.x)
}
```

@iChanSek

bit.ly/chansecode

Delegated Properties

```
class SynchronizedLazyImpl<out T>(initializer: () -> T, lock: Any? = null) : Lazy<T> {
    private var initializer: (() -> T)? = initializer

    @Volatile private var _value: Any? = UNINITIALIZED_VALUE
    private val lock = lock ?: this

    override val value: T
        get() {
            if (_value !== UNINITIALIZED_VALUE) {
                return _value as T
            }

            return synchronized(lock) {
                ...
            }
        }
}
```

@iChanSek

bit.ly/chansecode

Delegated Properties

```
class SynchronizedLazyImpl<out T>(initializer: () -> T, lock: Any? = null) : Lazy<T> {
    private var initializer: (() -> T)? = initializer
    @Volatile private var _value: Any? = UNINITIALIZED_VALUE
    private val lock = lock ?: this

    override val value: T
        get() {
            if (_value !== UNINITIALIZED_VALUE) {
                return _value as T
            }

            return synchronized(lock) {
                ...
            }
        }
}
```

@iChanSek

bit.ly/chansecode

1.3 Nullable and Non-Nullable

Nullable and Non-Nullable

```
val nullableConf: Conference? = null
val confName = nullableConf?.name
```

Nullable and Non-Nullable

```
Conference conf = nullableConf;
String confName =
     conf != null ? conf.getName() : null
```

Nullable and Non-Nullable

```
val c = a?.b?.c
```

Nullable and Non-Nullable

```
if (a != null) {
    B b = a.b;
}
```


@iChanSek

bit.ly/chansecode

Nullable and Non-Nullable

```
C c;
if (a != null) {
      B b = a.b;
      if (b != null) {
            c = b.c;
      }
}
```

@iChanSek

bit.ly/chansecode

Nullable and Non-Nullable

```
C c;
label1: {
    if (a != null) {
        B b = a.b;
        if (b != null) {
            c = b.c;
            break label1;
        }
    }
    c = null;
}
```

@iChanSek

bit.ly/chansecode

Nullable and Non-Nullable

```
val d = a!!.b!!.c
```

Nullable and Non-Nullable

```
if (a == null) throwNpe();
B b = a.b;
```

Nullable and Non-Nullable

```
if (a == null) throwNpe();
B b = a.b;

if (b == null) throwNpe();
C c = b.c;
```

@iChanSek

bit.ly/chansecode

2. Control Flows

2.1 If Expression

If Expression

```
val max = if (a > b) a else b
```

If Expression

```
int max = a > b ? a : b
```

If Expression

```
val max = if (a > b) {
    print("a is bigger")
    a
} else {
    print("b is bigger")
    b
}
```

@iChanSek

bit.ly/chansecode

If Expression

```
int max;
if (a > b) {
    S.O.P("a is bigger");
    max = a;
} else {
    S.O.P("b is bigger");
    max = b;
}
```

@iChanSek

bit.ly/chansecode

2.2 When Expression

When Expression

```
when (x) {
     1 -> print("x == 1")
     2 -> print("x == 2")
     else -> print("x is neither 1 nor 2")
}
```

@iChanSek

bit.ly/chansecode

When Expression

```
switch (x) {
    case 1:
        S.O.P("x == 1");
        break;
    case 2:
        S.O.P("x == 2");
        break;
    default:
        S.O.P("x is neither 1 nor 2");
        break;
}
```

@iChanSek

bit.ly/chansecode

When Expression

```
when (x) {
     1 -> print("x == 1")
     2 -> print("x == 2")
     else -> print("x is neither 1 nor 2")
}
```

@iChanSek

bit.ly/chansecode

When Expression

```
when {
    x == 1 -> print("x == 1")
    x == 2 -> print("x == 2")
    else -> print("x is neither 1 nor 2")
}
```

@iChanSek

bit.ly/chansecode

When Expression

```
if (x == 1) {
    S.O.P("x == 1");
} else if (x == 2) {
    S.O.P("x == 2");
} else {
    S.O.P("x is neither 1 nor 2”);
}
```

@iChanSek

bit.ly/chansecode

When Expression

```
enum class Colour {
    RED, GREEN, BLUE
}

when (colour) {
    Colour.RED -> ...
    Colour.GREEN -> ...
    Colour.BLUE -> ...
}
```

@iChanSek

bit.ly/chansecode

When Expression

```
switch(WhenMappings.$EnumSwitchMapping$0[colour.ordinal()]) {
    case 1:
        ...
        break;
    case 2:
        ...
        break;
    case 3:
        ...
        break;
    ......
}
```

@iChanSek

bit.ly/chansecode

When Expression

```
switch(WhenMappings.$EnumSwitchMapping$0[colour.ordinal()]) {
    case 1:
        ...
        break;
    case 2:
        ...
        break;
    case 3:
        ...
        break;
    ......
}
```

```
public final class WhenKt$WhenMappings {
  public static final int[] $EnumSwitchMapping$0 = new
int[Colour.values().length];

  static {
    $EnumSwitchMapping$0[Colour.RED.ordinal()] = 1;
    $EnumSwitchMapping$0[Colour.GREEN.ordinal()] = 2;
    $EnumSwitchMapping$0[Colour.BLUE.ordinal()] = 3;
  }
}
```


@iChanSek

bit.ly/chansecode

When Expression

```
switch(WhenMappings.$EnumSwitchMapping$0[colour.ordinal()]) {
    case 1:
        ...
        break;
    case 2:
        ...
        break;
    case 3:
        ...
        break;
    ......
}
```

```
public final class WhenKt$WhenMappings {
  public static final int[] $EnumSwitchMapping$0 = new
int[Colour.values().length];

  static {
    $EnumSwitchMapping$0[Colour.RED.ordinal()] = 1;
    $EnumSwitchMapping$0[Colour.GREEN.ordinal()] = 2;
    $EnumSwitchMapping$0[Colour.BLUE.ordinal()] = 3;
  }
}
```

@iChanSek

bit.ly/chansecode

3. Classes & Interfaces

3.1 Data Classes

Data Classes

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

@iChanSek

bit.ly/chansecode

Destructuring Declarations

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

val user = User("Chandra", 10)
val (name, age) = user
```

@iChanSek

bit.ly/chansecode

Destructuring Declarations

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

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

Destructuring declaration initializer of type User must have a 'component1()' function

Destructuring declaration initializer of type User must have a 'component2()' function

@iChanSek

bit.ly/chansecode

Destructuring Declarations

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

val user = User("Chandra", 10)
val (name, age) = user
```

@iChanSek

bit.ly/chansecode

Destructuring Declarations

```
public final class User {
    ...

    public final String component1() {
        return this.name;
    }

    public final int component2() {
        return this.age;
    }
}
```

```
val user = User("Chandra", 10)
val (name, age) = user
```

@iChanSek

bit.ly/chansecode

Destructuring Declarations

```
public final class User {
    ...

    public final String component1() {
        return this.name;
    }

    public final int component2() {
        return this.age;
    }
}
```

```
String name = user.component1();
int age = user.component2();
```

@iChanSek

bit.ly/chansecode

3.2 Sealed Classes

Sealed Classes

```
sealed class Calculation
class Add(val num1: Int, val num2: Int) : Calculation()
class Sub(val num1: Int, val num2: Int) : Calculation()
class Mul(val num1: Int, val num2: Int) : Calculation()
class Div(val num1: Int, val num2: Int) : Calculation()
```

Sealed Classes

```
public abstract class Calculation {
    private Calculation() {}
}

class Add(val num1: Int, val num2: Int) : Calculation()
class Sub(val num1: Int, val num2: Int) : Calculation()
class Mul(val num1: Int, val num2: Int) : Calculation()
class Div(val num1: Int, val num2: Int) : Calculation()
```

@iChanSek

bit.ly/chansecode

Sealed Classes

```
public abstract class Calculation {
    private Calculation() {}
}

public final class Add extends Calculation { ... }
public final class Sub extends Calculation { ... }
public final class Mul extends Calculation { ... }
public final class Div extends Calculation { ... }
```

@iChanSek

bit.ly/chansecode

Sealed Classes

```
private fun calculate(calculator: Calculation) {
    when(calculator) {
        is Add -> calculator.num1 + calculator.num2
        is Sub -> calculator.num1 - calculator.num2
        is Mul -> calculator.num1 * calculator.num2
        is Div -> calculator.num1 / calculator.num2
    }
}
```

@iChanSek

bit.ly/chansecode

Sealed Classes

```
private static final void calculate(Calculation calculator) {
    int result;
    if (calculator instanceof Add) {
        result = ((Add)calculator).getNum1() + ((Add)calculator).getNum2();
    } else if (calculator instanceof Sub) {
        result = ((Sub)calculator).getNum1() - ((Sub)calculator).getNum2();
    } else if (calculator instanceof Mul) {
        result = ((Mul)calculator).getNum1() * ((Mul)calculator).getNum2();
    } else if (calculator instanceof Div) {
        result = ((Div)calculator).getNum1() / ((Div)calculator).getNum2();
    }
}
```

@iChanSek

bit.ly/chansecode

3.3 Inline Classes

Inline Classes

```
inline class UserId(val value: Long)
```


@iChanSek

bit.ly/chansecode

Inline Classes

```
public final class UserId {
    private final long id;

    public static long constructor_impl(long id) {
        return id;
    }

    public static final UserId box_impl(long v) {
        return new UserId(v);
    }

    public final long unbox_impl() {
        return this.id;
    }
}
```

@iChanSek

bit.ly/chansecode

Inline Classes

```
public final class UserId {
    private final long id;

    public static long constructor_impl(long id) {
        return id;
    }

    public static final UserId box_impl(long v) {
        return new UserId(v);
    }

    public final long unbox_impl() {
        return this.id;
    }
}
```

@iChanSek

bit.ly/chansecode

Inline Classes

```
public final class UserId {
    private final long id;

    public static long constructor_impl(long id) {
        return id;
    }

    public static final UserId box_impl(long v) {
        return new UserId(v);
    }

    public final long unbox_impl() {
        return this.id;
    }
}
```

@iChanSek

bit.ly/chansecode

Inline Classes

```
public final class UserId {
    private final long id;

    public static long constructor_impl(long id) {
        return id;
    }

    public static final UserId box_impl(long v) {
        return new UserId(v);
    }

    public final long unbox_impl() {
        return this.id;
    }
}
```

@iChanSek

bit.ly/chansecode

Inline Classes

```
fun main() {
    val userId = UserId(10L)
    getUser(userId)
}

private fun getUser(id: UserId) {
    // Do something with id
}
```

@iChanSek

bit.ly/chansecode

Inline Classes

```
public static final void main() {
    long userId = UserId.constructor_impl(10L);
    getUser(userId);
}

// UserId became long in the argument
private static final void getUser(long id) {
    // Do something with id
}
```

@iChanSek

bit.ly/chansecode

3.4 Default Methods

Default Methods

```
interface Calculator {
    fun div(a: Int, b: Int) = ...
}
```

Default Methods

```
public interface Calculator {
    int div(int a, int b);

    public static final class DefaultImpls {
        public static int div(Calculator $this, int a, int b) {
            ...
        }
    }
}
```

Default Methods

```
class DefaultCalculatorImpl : Calculator

class CalculatorImpl : Calculator {
    override fun div(a: Int, b: Int): Int {
        ...
    }
}
```

@iChanSek

bit.ly/chansecode

Default Methods

```
public final class DefaultCalculatorImpl implements Calculator {
    public int div(int a, int b) {
        return Calculator.DefaultImpls.div(this, a, b);
    }
}

public final class CalculatorImpl implements Calculator {
    public int div(int a, int b) {
        ...
    }
}
```

@iChanSek

bit.ly/chansecode

4. Functions

4.1 Extension Function

Extension Function

```
fun IntArray.swap(index1: Int, index2: Int) {
    val temp = this[index1]
    this[index1] = this[index2]
    this[index2] = temp
}
```

@iChanSek

bit.ly/chansecode

Extension Function

```
public static final void swap(int[] arr, int index1, int index2) {
    Intrinsics.checkParameterIsNotNull(arr, "arr");
    int temp = arr[index1];
    arr[index1] = arr[index2];
    arr[index2] = temp;
}
```

@iChanSek

bit.ly/chansecode

Extension Function

```
val arr = intArrayOf(1, 2, 3, 4, 5)
arr.swap(2, 4)
```

Extension Function

```
int[] arr = new int[]{1, 2, 3, 4, 5};
swap(arr, 2, 4);
```

4.2 Default Arguments

Default Arguments

```
fun add(a: Int,
        b : Int,
        c: Int = 100,
        d: Int = 200,
        e: Int = 300) =
    a + b + c + d + e
```


@iChanSek


bit.ly/chansecode

Default Arguments

```
... int add(int a, int b, int c, int d, int e) {
  return a + b + c + d + e;
}
```

Default Arguments

```
... int add(int a, int b, int c, int d, int e) {
  return a + b + c + d + e;
}

fun main() {
    add(2, 3, 4, 5, 6)
}
```

@iChanSek

bit.ly/chansecode

Default Arguments

```
... int add(int a, int b, int c, int d, int e) {
  return a + b + c + d + e;
}

fun main() {
    add(2, 3, 4, 5, 6)
    add(2, 3)
}
```

@iChanSek

bit.ly/chansecode

Default Arguments

```
... int add(int a, int b, int c, int d, int e) {
  return a + b + c + d + e;
}

fun main() {
    add(2, 3, 4, 5, 6)
    add(2, 3)
    add(2, 3, 4, 5)
}
```

@iChanSek

bit.ly/chansecode

Default Arguments

```
... int add(int a, int b, int c, int d, int e) {
  return a + b + c + d + e;
}

public static final void main() {
  add(2, 3, 4, 5, 6);
  add(2, 3, 0, 0, 0, 28, _);
  add(2, 3, 4, 5, 0, 16, _);
}
```

@iChanSek

bit.ly/chansecode

Default Arguments

```
... int add$default(int a, int b, int c, int d, int e, int var5, _) {
  ...
}

public static final void main() {
   add(2, 3, 4, 5, 6);
   add$default(2, 3, 0, 0, 0, 28, _);
   add$default(2, 3, 4, 5, 0, 16, _);
}
```

@iChanSek

bit.ly/chansecode

Default Arguments

```
... int add$default(int a, int b, int c, int d, int e, int var5, …) {
  ...
}

public static final void main() {
  add(2, 3, 4, 5, 6);
  add$default(2, 3, 0, 0, 0, 2² + 2³ + 2⁴, _);
  add$default(2, 3, 4, 5, 0, 2⁴, _);
}
```

@iChanSek

bit.ly/chansecode

Default Arguments

```
... int add$default(int a, int b, int c, int d, int e, int var5, _) {
  if ((var5 & 4) != 0) c = 100;
  if ((var5 & 8) != 0) d = 200;
  if ((var5 & 16) != 0) e = 300;

  return add(a, b, c, d, e);
}
```

@iChanSek

bit.ly/chansecode

4.3 Local Functions

Local Functions

```
fun outer() {
    fun inner() {
        println(“ Kolkata”)
    }
    println(“Welcome to DevFest")
    if (event == "DevFest Kolkata") inner()
}
```

@iChanSek

bit.ly/chansecode

Local Functions

```
public static final void outer() {
  Inner inner = Inner.INSTANCE;
  System.out.println("Welcome to DevFest");
  if (Intrinsics.areEqual(event, "DevFest Kolkata")) {
    inner.invoke();
  }
}
```


@iChanSek


bit.ly/chansecode

Local Functions

```
public static final void outer() {
    Inner inner = Inner.INSTANCE;
  System.out.println("Welcome to DevFest");
  if (Intrinsics.areEqual(event, "DevFest Kolkata")) {
    inner.invoke();
  }
}
```

```
final class Inner extends Lambda
                  implements Function0 {
  ... Inner INSTANCE = new Inner();

  public final void invoke() {
    System.out.println(" Kolkata");
  }

  Inner() { super(0); }
}
```

@iChanSek

bit.ly/chansecode

4.4 Lambda Functions

Lambda Functions

```
val sum: (Int, Int) -> Int = { x, y -> x + y }
```

@iChanSek

bit.ly/chansecode

Lambda Functions

```
val sum: (Int, Int) -> Int = { x, y -> x + y }
```

```
... Function2<_, _, _> sum = new Function2<_, _, _>() {
    @Override
    public final int invoke(int a, int b) {
        return a + b;
    }
};
```

@iChanSek

bit.ly/chansecode

Lambda Functions

```
() -> Unit == class XXX implements Function0<Unit>
```

Lambda Functions

```
        () -> Unit == class XXX implements Function0<Unit>
(Int, Int) -> Int == class XXX implements Function2<Int, Int, Int>
```

4.5 Lambda with Receiver

Lambda with Receiver

```
val sum: (Int, Int) -> Int = { x, y -> x + y }
```


@iChanSek

bit.ly/chansecode

Lambda with Receiver

```
val sum: Int.(Int) -> Int = { this + it }
```


@iChanSek

bit.ly/chansecode

Lambda with Receiver

```
val sum: Int.(Int) -> Int = { this + it }
```

```
... Function2<_, _, _> sum = new Function2<_, _, _>() {
    @Override
    public final int invoke(int $receiver, int a) {
        return $receiver + a;
    }
};
```

@iChanSek

bit.ly/chansecode

Lambda Functions

```
        () -> Unit == class XXX implements Function0<Unit>
(Int, Int) -> Int == class XXX implements Function2<Int, Int, Int>
```

Lambda Functions

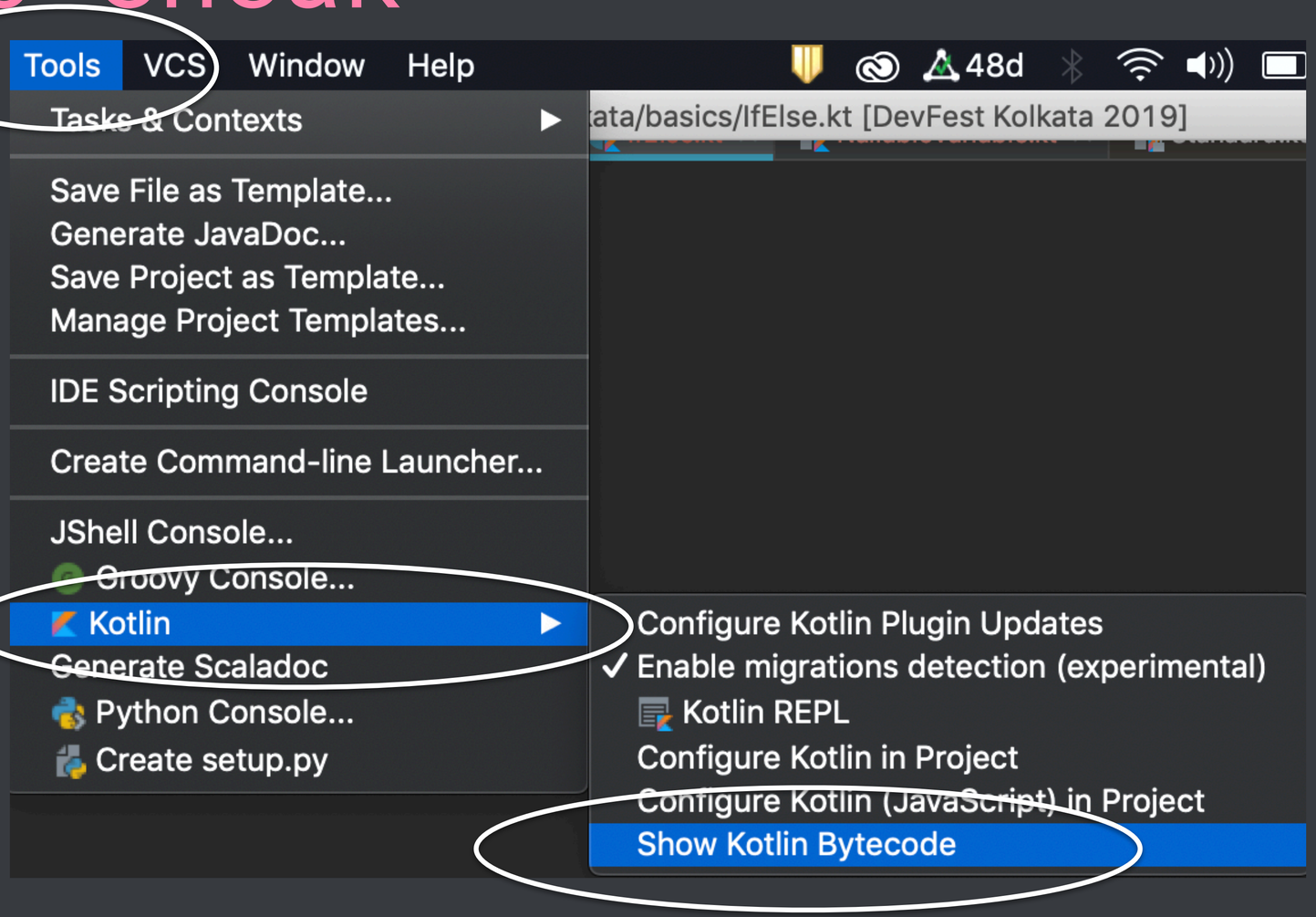
```
        () -> Unit == class XXX implements Function0<Unit>
(Int, Int) -> Int == class XXX implements Function2<Int, Int, Int>
 Int.(Int) -> Int == class XXX implements Function2<Int, Int, Int>
```


@iChanSek

bit.ly/chansecode

5. How to Sneak

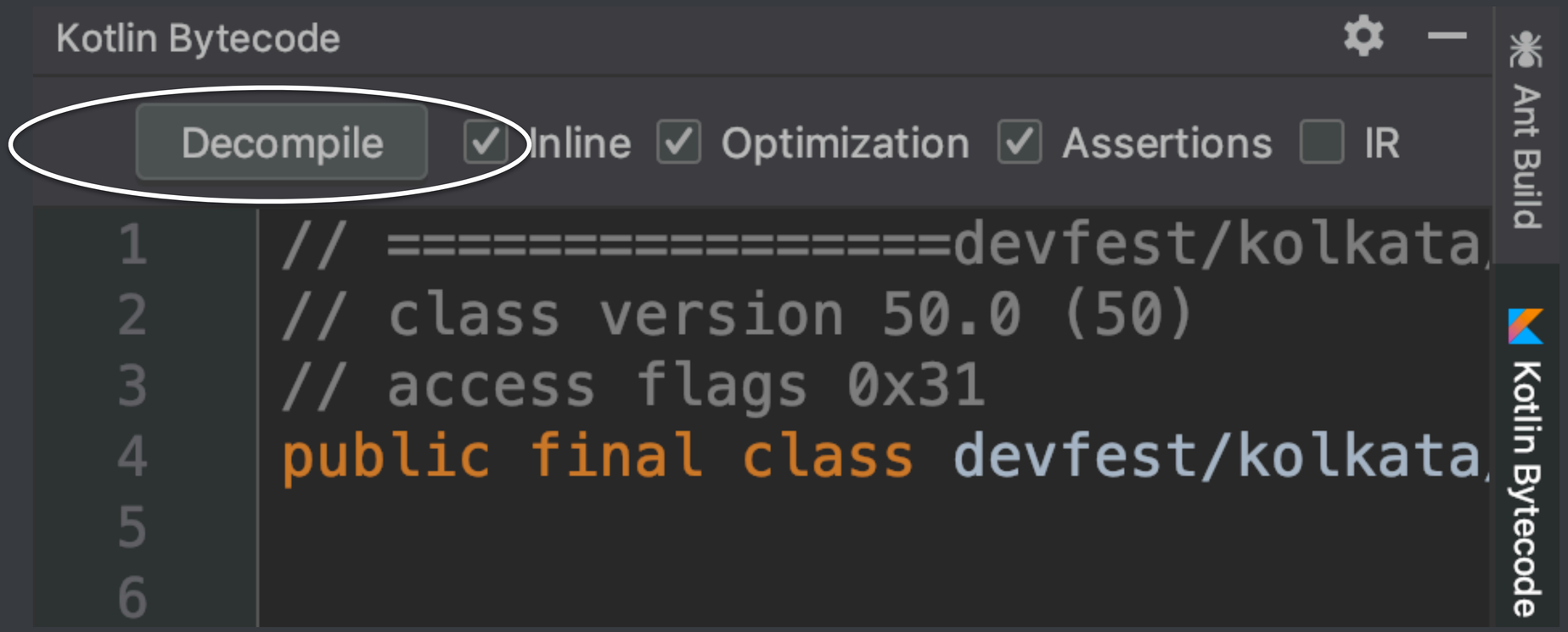
How to Sneak



@iChanSek

bit.ly/chansecode

How to Sneak



@iChanSek

bit.ly/chansecode

Thank You

Chandra Sekhar

@iChanSek

bit.ly/chansecode