

Kotlin Workshop

Patrick Moen Allport
Utvikler

Hvem er vi?



Patrick
2 år i Bekk



Frikk
4 år i Bekk



Ole Reidar
7 år i Bekk

Agenda

1. Lynkurs i Kotlin ⚡
2. Det progges 😎

Hva er Kotlin?

- Java, dratt inn i den moderne verdenen 🔥
 - Sterkt fokus på lesbarhet
 - Så og si 100% interoperability med java

Lynkurs! ⚡

Variabler

```
int tallEn = 0
```

Variabler

```
int tallEn = 0
```

```
var tallTo: Int = 0
```

```
var tallTo = 0
```

```
tallTo = 1 // == 1
```

```
tallTo += 1 // == 2
```

```
tallTo ++ // == 3
```

Variabler

```
int tallEn = 0
```

```
var tallTo: Int = 0
```

```
var tallTo = 0
```

```
tallTo = 1 // == 1
```

```
tallTo += 1 // == 2
```

```
tallTo ++ // == 3
```

```
val tallTre = 0
```

```
tallTre = 1 // ikke lov
```

```
val tallFire = tallTre + 1 // == 1
```


Funksjoner

```
public int add(int a, int b) {  
    return a + b;  
}
```

Funksjoner

```
fun add(a: Int, b: Int): Int {  
    return a + b  
}
```

Funksjoner

```
fun add(a: Int, b: Int): Int {  
    return a + b  
}
```

```
fun add(a: Int, b: Int) = a + b
```

If-expressions

```
fun maxOf(a: Int, b: Int): Int {  
    if (a > b) {  
        return a  
    } else {  
        return b  
    }  
}
```

```
fun maxOf(a: Int, b: Int) = if (a > b) a else b
```

Nullability

```
int stringLength(String a) {  
    return a.length();  
}  
  
void main() {  
    stringLength(null); // Kaster en `NullPointerException`  
}
```

Nullability

```
int stringLength(String a) {  
    return a.length();  
}
```

```
void main() {  
    stringLength(null); // Kaster en `NullPointerException`  
}
```

```
fun stringLength(a: String?) = a.length // Kaster en `NullPointerException`  
  
fun main() {  
    stringLength(null)  
}
```

Nullability

```
int stringLength(String a) {  
    return a.length();  
}
```

```
void main() {  
    stringLength(null); // Kaster en `NullPointerException`  
}
```

```
fun stringLength(a: String) = a.length
```

```
fun main() {  
    stringLength(null) // Ikke lov!  
}
```

Nullability

```
fun stringLength(a: String?): Int = if (a != null) a.length else 0
stringLength(null) // == 0
stringLength("Hallo") // == 5
```


Nullability

```
fun stringLength(a: String?): Int = if (a != null) a.length else 0
stringLength(null) // == 0
stringLength("Hallo") // == 5
```

```
fun stringLengthOrNull(a: String?): Int? = a?.length
stringLengthOrNull(null) // == null
stringLengthOrNull("Hallo") // == 5
```

Nullability

```
fun stringLength(a: String?): Int = if (a != null) a.length else 0
stringLength(null) // == 0
stringLength("Hallo") // == 5
```

```
fun stringLengthOrNull(a: String?): Int? = a?.length
stringLengthOrNull(null) // == null
stringLengthOrNull("Hallo") // == 5
```

```
fun stringLengthOrZero(a: String?): Int = a?.length ?: 0
stringLengthOrZero(null) // == 0
stringLengthOrZero("Hallo") // == 5
```

Klasser

```
public class BootcampCoach {  
    private final String name;  
    private int yearsInBekk;  
  
    public BootcampCoach(String name, int yearsInBekk) {  
        this.name = name;  
        this.yearsInBekk = yearsInBekk;  
    }  
  
    public String getName() {  
        return name;  
    }  
  
    public int getYearsInBekk() {  
        return yearsInBekk;  
    }  
  
    public void setYearsInBekk(  
        int yearsInBekk  
    ) {  
        this.yearsInBekk = yearsInBekk;  
    }  
  
    public void introduce() {  
        System.out.println(  
            "Hei, jeg heter " + name +  
            " og har vært i Bekk i " + yearsInBekk + " år");  
    }  
}
```

```
class BootcampCoach(  
    val name: String,  
    var yearsInBekk: Int  
) {  
    fun introduce() {  
        println(  
            "Hei, jeg heter $name " +  
            "og har vært i Bekk i $yearsInBekk år")  
    }  
}
```

Klasser

```
public class BootcampCoach {  
    private final String name;  
    private int yearsInBekk;  
  
    public BootcampCoach(  
        String name,  
        int yearsInBekk  
    ) {  
        this.name = name;  
        this.yearsInBekk = yearsInBekk;  
    }  
}
```

```
class BootcampCoach(  
    val name: String,  
    var yearsInBekk: Int  
)
```

Klasser

```
public class BootcampCoach {  
    private final String name;  
    private int yearsInBekk;  
  
    public BootcampCoach(  
        String name,  
        int yearsInBekk  
    ) {  
        this.name = name;  
        this.yearsInBekk = yearsInBekk;  
    }  
}
```

```
class BootcampCoach(  
    val name: String,  
    var yearsInBekk: Int  
)  
  
val bootcampCoach =  
    BootcampCoach(  
        name = "Patrick",  
        yearsInBekk = 2  
    )
```

Klasser

```
public class BootcampCoach {
    private final String name;
    private int yearsInBekk;

    public BootcampCoach(String name, int yearsInBekk) {
        this.name = name;
        this.yearsInBekk = yearsInBekk;
    }

    public String getName() {
        return name;
    }

    public int getYearsInBekk() {
        return yearsInBekk;
    }

    public void setYearsInBekk(
        int yearsInBekk
    ) {
        this.yearsInBekk = yearsInBekk;
    }

    public void introduce() {
        System.out.println(
            "Hei, jeg heter " + name +
            " og har vært i Bekk i " + yearsInBekk + " år");
    }
}
```

```
class BootcampCoach(
    val name: String,
    var yearsInBekk: Int
) {
    fun introduce() {
        println(
            "Hei, jeg heter $name " +
            "og har vært i Bekk i $yearsInBekk år")
    }
}
```

Klasser

```
public String getName() {  
    return name;  
}
```

```
public int getYearsInBekk() {  
    return yearsInBekk;  
}
```

```
public void setYearsInBekk(  
    int yearsInBekk  
) {  
    this.yearsInBekk = yearsInBekk;  
}
```

```
class BootcampCoach(  
    val name: String,  
    var yearsInBekk: Int  
)
```

Klasser

```
public String getName() {  
    return name;  
}  
  
public int getYearsInBekk() {  
    return yearsInBekk;  
}  
  
public void setYearsInBekk(  
    int yearsInBekk  
) {  
    this.yearsInBekk = yearsInBekk;  
}
```

```
class BootcampCoach(  
    val name: String,  
    var yearsInBekk: Int  
)  
  
val bootcampCoach = ...  
println(bootcampCoach.name)  
bootcampCoach.yearsInBekk = 3
```


Klasser

```
public class BootcampCoach {  
    private final String name;  
    private int yearsInBekk;  
  
    public BootcampCoach(String name, int yearsInBekk) {  
        this.name = name;  
        this.yearsInBekk = yearsInBekk;  
    }  
  
    public String getName() {  
        return name;  
    }  
  
    public int getYearsInBekk() {  
        return yearsInBekk;  
    }  
  
    public void setYearsInBekk(  
        int yearsInBekk  
    ) {  
        this.yearsInBekk = yearsInBekk;  
    }  
  
    public void introduce() {  
        System.out.println(  
            "Hei, jeg heter " + name +  
            " og har vært i Bekk i " + yearsInBekk + " år");  
    }  
}
```

```
class BootcampCoach(  
    val name: String,  
    var yearsInBekk: Int  
) {  
    fun introduce() {  
        println(  
            "Hei, jeg heter $name " +  
            "og har vært i Bekk i $yearsInBekk år")  
    }  
}
```

Klasser

```
public void introduce() {  
    System.out.println(  
        "Hei, jeg heter " + name +  
        " og har vært i Bekk i " + yearsInBekk + " år");  
}
```

```
fun introduce() {  
    println(  
        "Hei, jeg heter $name " +  
        "og har vært i Bekk i $yearsInBekk år")  
}
```

Klasser

```
public class BootcampCoach {  
    private final String name;  
    private int yearsInBekk;  
  
    public BootcampCoach(String name, int yearsInBekk) {  
        this.name = name;  
        this.yearsInBekk = yearsInBekk;  
    }  
  
    public String getName() {  
        return name;  
    }  
  
    public int getYearsInBekk() {  
        return yearsInBekk;  
    }  
  
    public void setYearsInBekk(  
        int yearsInBekk  
    ) {  
        this.yearsInBekk = yearsInBekk;  
    }  
  
    public void introduce() {  
        System.out.println(  
            "Hei, jeg heter " + name +  
            " og har vært i Bekk i " + yearsInBekk + " år");  
    }  
}
```

```
class BootcampCoach(  
    val name: String,  
    var yearsInBekk: Int  
) {  
    fun introduce() {  
        println(  
            "Hei, jeg heter $name " +  
            "og har vært i Bekk i $yearsInBekk år")  
    }  
}
```


Kule ting i kotlin (Som gjør ting mer leselig <3)!

Data class

```
class Person(  
    val navn: String,  
    val alder: Int  
)
```

Data class

```
class Person(  
    val navn: String,  
    val alder: Int  
)
```



```
val person1 = Person(  
    navn = "Patrick",  
    alder = 27  
)
```

```
val person2 = Person(  
    navn = "Patrick",  
    alder = 27  
)
```

Data class

```
class Person(  
    val navn: String,  
    val alder: Int  
)
```

```
val person1 = Person(  
    navn = "Patrick",  
    alder = 27  
)
```

```
val person2 = Person(  
    navn = "Patrick",  
    alder = 27  
)
```

Data class

```
class Person(  
    val navn: String,  
    val alder: Int  
)
```

```
val person1 = Person(  
    navn = "Patrick",  
    alder = 27  
)
```



```
val person2 = Person(  
    navn = "Patrick",  
    alder = 27  
)
```


Data class

```
class Person(  
    val navn: String,  
    val alder: Int  
) {  
    override fun equals(other: Any?): Boolean {  
        if (this === other) return true  
        if (other !is Person) return false  
  
        if (navn != other.navn) return false  
        if (alder != other.alder) return false  
  
        return true  
    }  
}
```

```
val person1 = Person(  
    navn = "Patrick",  
    alder = 27  
)
```

```
val person2 = Person(  
    navn = "Patrick",  
    alder = 27  
)
```

Data class

```
class Person(  
    val navn: String,  
    val alder: Int  
)
```

Data class

```
data class Person(  
    val navn: String,  
    val alder: Int  
)
```

Data class

```
data class Person(  
    val navn: String,  
    val alder: Int  
) {  
  
    override fun equals(other: Any?): Boolean {  
        if (this === other) return true  
        if (other !is Person) return false  
  
        if (navn != other.navn) return false  
        if (alder != other.alder) return false  
  
        return true  
    }  
  
    override fun hashCode(): Int {  
        var result = navn.hashCode()  
        result = 31 * result + alder  
        return result  
    }  
}
```

Data class

```
class Person(  
    val navn: String,  
    val alder: Int  
)
```

```
val person1 = Person(  
    navn = "Patrick",  
    alder = 27  
)
```

```
val person2 = Person(  
    navn = "Patrick",  
    alder = 27  
)
```

Data class

```
data class Person(  
    val navn: String,  
    val alder: Int  
)
```

```
val person1 = Person(  
    navn = "Patrick",  
    alder = 27  
)
```

```
val person2 = Person(  
    navn = "Patrick",  
    alder = 27  
)
```

Data class

```
data class Person(  
    val navn: String,  
    val alder: Int  
)
```

```
val person1 = Person(  
    navn = "Patrick",  
    alder = 27  
)
```



```
val person2 = Person(  
    navn = "Patrick",  
    alder = 27  
)
```

Argumenter

```
class Person(  
    val navn: String,  
    val alder: Int  
)
```


Argumenter

```
class Person(  
    val navn: String = "Anonym",  
    val alder: Int = 42  
)
```

Argumenter

```
class Person(  
    val navn: String = "Anonym",  
    val alder: Int = 42  
)
```

```
Person(navn = "Patrick", alder = 27) // Person(navn = "Patrick", alder = 27)
```

```
Person(navn = "Patrick") // Person(navn = "Patrick", alder = 42)
```

```
Person() // Person(navn = "Anonym", alder = 42)
```

Argumenter

```
class Person(  
    val navn: String = "Anonym",  
    val alder: Int = 42  
)  
  
Person(navn = "Patrick", alder = 27) // Person(navn = "Patrick", alder = 27)  
  
Person(navn = "Patrick") // Person(navn = "Patrick", alder = 42)  
  
Person() // Person(navn = "Anonym", alder = 42)  
  
fun greet(navn: String = "world") = println("Hello, $navn!")  
  
greet("Patrick") // Hello, Patrick!  
  
greet() // Hello, world!
```

Lister I kotlin

```
val numbers: MutableList<Int> = mutableListOf(1, 2, 3)
numbers.add(4) // [1, 2, 3, 4]
numbers.clear() // []
```

Lister I kotlin

```
val numbers: MutableList<Int> = mutableListOf(1, 2, 3)
numbers.add(4) // [1, 2, 3, 4]
numbers.clear() // []
```

```
val numbers: List<Int> = listOf(1, 2, 3)
numbers.add(4) // ikke lov
```

Lambda-funksjoner

Lambda-funksjoner

Variabler

```
int tallEn = 0
```

```
var tallTo: Int = 0
```

```
var tallTo = 0
```

```
tallTo = 1 // == 1
```

```
tallTo += 1 // == 2
```

```
tallTo ++ // == 3
```

```
val tallTre = 0
```

```
tallTre = 1 // ikke lov
```

```
val tallFire = tallTre + 1 // == 1
```

Lambda-funksjoner

Lambda-funksjoner

```
val patrick = Person(  
    navn = "Patrick",  
    alder = 27  
)  
  
val eldrePatrick = patrick.copy(  
    alder = patrick.alder + 1  
)  
// == Person(navn = "Patrick", alder = 28)
```

```
data class Person(  
    val navn: String,  
    val alder: Int  
)
```

Lambda-funksjoner

```
data class Person(  
    val navn: String,  
    val alder: Int  
)
```

```
fun bliEttAarEldre(person: Person) = person.copy(alder = person.alder + 1)
```

```
val patrick = Person(  
    navn = "Patrick",  
    alder = 27  
)
```

```
val eldrePatrick = patrick.copy(  
    alder = patrick.alder + 1  
)  
// == Person(navn = "Patrick", alder = 28)
```

Lambda-funksjoner

```
data class Person(  
    val navn: String,  
    val alder: Int  
)
```

```
fun bliEttAarEldre(person: Person) = person.copy(alder = person.alder + 1)
```

```
val patrick = Person(  
    navn = "Patrick",  
    alder = 27  
)
```

```
val eldrePatrick = bliEttAarEldre(patrick)  
// == Person(navn = "Patrick", alder = 28)
```

Lambda-funksjoner

```
data class Person(  
    val navn: String,  
    val alder: Int  
)
```

```
fun bliEttAarEldre(person: Person) = person.copy(alder = person.alder + 1)
```

```
val patrick = Person(navn = "Patrick", alder = 27)
```

```
val sondre = Person(navn = "Sondre", alder = 28)
```

```
val frikk = Person(navn = "Frikk", alder = 29)
```

```
val ungePersoner = listOf(patrick, sondre, frikk)
```

Lambda-funksjoner

```
data class Person(  
    val navn: String,  
    val alder: Int  
)
```

```
fun bliEttAarEldre(person: Person) = person.copy(alder = person.alder + 1)
```

```
val patrick = Person(navn = "Patrick", alder = 27)
```

```
val sondre = Person(navn = "Sondre", alder = 28)
```

```
val frikk = Person(navn = "Frikk", alder = 29)
```

```
val ungePersoner = listOf(patrick, sondre, frikk)
```

```
val eldrePersoner = ungePersoner.map { person -> bliEttAarEldre(person)}
```

Lambda-funksjoner

```
data class Person(  
    val navn: String,  
    val alder: Int  
)
```

```
fun bliEttAarEldre(person: Person) = person.copy(alder = person.alder + 1)
```

```
val patrick = Person(navn = "Patrick", alder = 27)
```

```
val sondre = Person(navn = "Sondre", alder = 28)
```

```
val frikk = Person(navn = "Frikk", alder = 29)
```

```
val ungePersoner = listOf(patrick, sondre, frikk)
```

```
val eldrePersoner = ungePersoner.map { person -> bliEttAarEldre(person)}
```

```
val eldrePersoner = ungePersoner.map { bliEttAarEldre(it)}
```

Lambda-funksjoner

```
data class Person(  
    val navn: String,  
    val alder: Int  
)
```

```
fun bliEttAarEldre(person: Person) = person.copy(alder = person.alder + 1)
```

```
val patrick = Person(navn = "Patrick", alder = 27)
```

```
val sondre = Person(navn = "Sondre", alder = 28)
```

```
val frikk = Person(navn = "Frikk", alder = 29)
```

```
val ungePersoner = listOf(patrick, sondre, frikk)
```

```
val eldrePersonerOver28 = ungePersoner
```

Lambda-funksjoner

```
data class Person(  
    val navn: String,  
    val alder: Int  
)
```

```
fun bliEttAarEldre(person: Person) = person.copy(alder = person.alder + 1)
```

```
val patrick = Person(navn = "Patrick", alder = 27)
```

```
val sondre = Person(navn = "Sondre", alder = 28)
```

```
val frikk = Person(navn = "Frikk", alder = 29)
```

```
val ungePersoner = listOf(patrick, sondre, frikk)
```

```
val eldrePersonerOver28 = ungePersoner  
    .map { bliEttAarEldre(it)}
```


Lambda-funksjoner

```
data class Person(  
    val navn: String,  
    val alder: Int  
)
```

```
fun bliEttAarEldre(person: Person) = person.copy(alder = person.alder + 1)  
fun erPersonenEldreEnn(person: Person, alder: Int) = person.alder > alder  
  
val patrick = Person(navn = "Patrick", alder = 27)  
val sondre = Person(navn = "Sondre", alder = 28)  
val frikk = Person(navn = "Frikk", alder = 29)  
  
val ungePersoner = listOf(patrick, sondre, frikk)  
  
val eldrePersonerOver28 = ungePersoner  
    .map { bliEttAarEldre(it)}
```

Lambda-funksjoner

```
data class Person(  
    val navn: String,  
    val alder: Int  
)
```

```
fun bliEttAarEldre(person: Person) = person.copy(alder = person.alder + 1)  
fun erPersonenEldreEnn(person: Person, alder: Int) = person.alder > alder
```

```
val patrick = Person(navn = "Patrick", alder = 27)  
val sondre = Person(navn = "Sondre", alder = 28)  
val frikk = Person(navn = "Frikk", alder = 29)
```

```
val ungePersoner = listOf(patrick, sondre, frikk)
```

```
val eldrePersonerOver28 = ungePersoner  
    .map { bliEttAarEldre(it) }  
    .filter { erPersonenEldreEnn(it, 28) }
```

Extension functions

```
fun String.shout(): String {  
    return this.uppercase()  
}
```

```
"hello".shout() // "HELLO!"
```

Extension functions

```
fun String.shout(): String {  
    return this.uppercase()  
}
```

```
"hello".shout() // "HELLO!"
```

```
val strenger = listOf("hei", "på", "deg")
```

```
val skrikendeStrenger = strenger.map { it.shout() } // listOf("HEI", "PÅ", "DEG")
```

Extension functions

```
fun String.shout(): String {  
    return this.uppercase()  
}
```

```
"hello".shout() // "HELLO!"
```

```
val strenger = listOf("hei", "på", "deg")
```

```
val skrikendeStrenger = strenger.map { it.shout() } // listOf("HEI", "PÅ", "DEG")
```

```
fun List<String>.somSkrikendeStrenger() = this.map { it.uppercase() }
```

```
val skrikendeStrengerListe = strenger.somSkrikendeStrenger()  
// listOf("HEI", "PÅ", "DEG")
```

Lambda-funksjoner ++

```
data class Person(  
    val navn: String,  
    val alder: Int  
)
```

```
fun bliEttAarEldre(person: Person) = person.copy(alder = person.alder + 1)  
fun erPersonenEldreEnn(person: Person, alder: Int) = person.alder > alder
```

```
val patrick = Person(navn = "Patrick", alder = 27)  
val sondre = Person(navn = "Sondre", alder = 28)  
val frikk = Person(navn = "Frikk", alder = 29)
```

```
val ungePersoner = listOf(patrick, sondre, frikk)
```

```
val eldrePersonerOver28 = ungePersoner  
    .map { bliEttAarEldre(it) }  
    .filter { erPersonenEldreEnn(it, 28) }
```

Lambda-funksjoner ++

```
fun bliEttAarEldre(person: Person) = person.copy(alder = person.alder + 1)
fun erPersonenEldreEnn(person: Person, alder: Int) = person.alder > alder
```

Lambda-funksjoner ++

```
fun bliEttAarEldre(person: Person) = person.copy(alder = person.alder + 1)
fun List<Person>.gjorAlleEttAarEldre() = this.map { person ->
    person.copy(alder = person.alder + 1)
}
```

```
fun erPersonenEldreEnn(person: Person, alder: Int) = person.alder > alder
fun List<Person>.filtrerForAlleOver(minimumsAlder: Int) = this.filter { person ->
    person.alder > minimumsAlder
}
```


Lambda-funksjoner ++

```
fun List<Person>.gjorAlleEttAarEldre() = this.map { person ->
    person.copy(alder = person.alder + 1)
}
fun List<Person>.filtrerForAlleOver(minimumsAlder: Int) = this.filter { person ->
    person.alder > minimumsAlder
}
```

Lambda-funksjoner ++

```
fun List<Person>.gjorAlleEttAarEldre() = this.map { person ->
    person.copy(alder = person.alder + 1)
}
fun List<Person>.filtrerForAlleOver(minimumsAlder: Int) = this.filter { person ->
    person.alder > minimumsAlder
}

val ungePersoner = listOf(patrick, sondre, frikk)

val eldrePersonerOver28 = ungePersoner
```

Lambda-funksjoner ++

```
fun List<Person>.gjorAlleEttAarEldre() = this.map { person ->
    person.copy(alder = person.alder + 1)
}

fun List<Person>.filtrerForAlleOver(minimumsAlder: Int) = this.filter { person ->
    person.alder > minimumsAlder
}

val ungePersoner = listOf(patrick, sondre, frikk)

val eldrePersonerOver28 = ungePersoner
    .map { bliEttAarEldre(it)}
    .filter { erPersonenEldreEnn(it, 28) }
```

Lambda-funksjoner ++

```
fun List<Person>.gjorAlleEttAarEldre() = this.map { person ->
    person.copy(alder = person.alder + 1)
}

fun List<Person>.filtrerForAlleOver(minimumsAlder: Int) = this.filter { person ->
    person.alder > minimumsAlder
}

val ungePersoner = listOf(patrick, sondre, frikk)

val eldrePersonerOver28 = ungePersoner
    .gjorAlleEttAarEldre()
    .filter { erPersonenEldreEnn(it, 28) }
```

Lambda-funksjoner ++

```
fun List<Person>.gjorAlleEttAarEldre() = this.map { person ->
    person.copy(alder = person.alder + 1)
}

fun List<Person>.filtrerForAlleOver(minimumsAlder: Int) = this.filter { person ->
    person.alder > minimumsAlder
}

val ungePersoner = listOf(patrick, sondre, frikk)

val eldrePersonerOver28 = ungePersoner
    .gjorAlleEttAarEldre()
    .filtrerForAlleOver(28)
```

Bekk
B k
B

Lynkurs overstått!

Spørsmål?

Tid for progging!

<https://github.com/bekk/kotlin-workshop-bootcamp-2025>

1. Klon repoet
2. Åpne i IntelliJ
3. Åpne fila README.md
4. Følg instruksene der!

(Slides ligger i mappa docs)