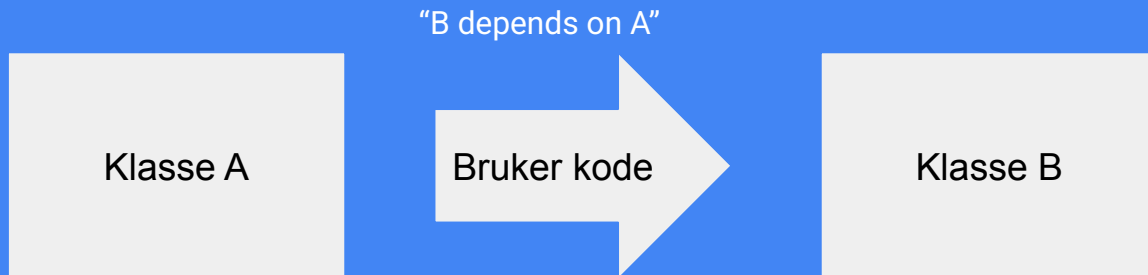


Dependency Injection

Torbjørn
IN2000 V26
(Original: Ånund og Victor V25)



Hva er Dependency Injection?

- Designmønster for å håndtere avhengigheter
 - Mønstre fra 1010:
 - Singleton
 - Observer
 - Listener
 - Factory
- Prosessen ved å gi en klasse sine instansvariabler
- Enkelt, men viktig konsept i apputvikling

Formål

Redusere hardkodede avhengigheter mellom objekter

Motivasjon 100

- Løs kobling -> Gjør koden fleksibel og enkel å vedlikeholde
- Enklere å teste -> Kan raskt sette inn testmoduler og isolere moduler i test
- Gjenbrukbarhet -> Injisere én avhengighet i flere moduler

Rammeverk i Kotlin/Compose for Dependency Injection

Hilt

- Offisielle DI biblioteket for Android bygget på et annet bibliotek som heter Dagger

Hva er en dependency? 🎓

```
class Student{  
    val name: String = "Anund"  
}
```

Injection

```
class Student(  
    val name: String  
) {  
  
}
```

Typing DI - Constructor Injection

```
// Constructor Injection  
class Student(  
    val name: String  
) {  
  
}
```

Typing DI - Method Injection

```
// Method Injection
class Student {
    lateinit var name: String

    fun setName(name: String){
        this.name = name
    }
}
```

Typer DI - Field Injection

```
// Field Injection (HILT)
class StudentHILT @Inject constructor() {
    lateinit var name: String
}

// Field Injection u/ HILT
class Student {
    lateinit var name: String

    // I Main:
    fun main(){
        val student: Student = Student()
        student.name = "Anund"
    }
}
```

Testing... 🤔

- Hvorfor bruke interfacer når vi kun har én implementasjon?

```
interface Database {  
    fun execute(query: Query)  
}
```

Hvorfor er dette hensiktsmessig?

```
class Repository(  
    private val database: Database  
) {  
  
}
```

Bruke lokal db...

```
class Test{  
  
    fun main(){  
        //Bruke RoomDB  
        val database: Database = RoomDB()  
  
        val repository: Repository = Repository(database)  
    }  
}
```

Bytte til produksjon?

```
class Test{  
  
    fun main(){  
        //Bruke AmazonDB  
        val database: Database = AmazonDB()  
  
        val repository: Repository = Repository(database)  
    }  
}
```

Forskjell på manuell DI og DI med HILT

```
@HiltAndroidApp  
class MyApp : Application()
```

```
@AndroidEntryPoint  
class MainActivity : AppCompatActivity() {  
  
    @Inject  
    lateinit var repository: Repository  
  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
    }  
}
```

```
// Annotasjon for HILT  
@Module  
@InstallIn(SingletonComponent::class)  
object DatabaseModule {  
  
    // HILT  
    @Provides  
    @Singleton  
    fun provideDatabase(): Database = RoomDB()  
}
```

Prinsipper

- Klasser skal ikke instansiere egne avhengigheter
- Klasser skal kun bruke objekter som injiseres i de
- Avhengigheter settes inn under kjøretid
- Avhengigheter bør være generelle (Interfacer fremfor implementasjoner)
- IoC (Inversion of Control) - flytt ansvar ut av objektene, bruk eventlisteners, rammeverk osv.

Hva slags type injisering?

Constructor Injection:

- Når objektet ikke kan eksistere uten avhengigheten sin

Setter / Method Injection:

- Når avhengigheten er valgfri eller må instansieres basert på en betingelse
- Eksempler?

Interessant?

Se:

- <https://developer.android.com/training/dependency-injection>