

LAPORAN PRAKTIKUM PENGEMBANGAN APLIKASI BERGERAK

Pengenalan Kotlin - Week 2



Disusun oleh:

Nama : Muhammad Ryan Fikri Fakhrezi

Nim : L0122114

Kelas : C

PROGRAM STUDI INFORMATIKA

FAKULTAS TEKNOLOGI INFORMASI DAN SAINS DATA

UNIVERSITAS SEBELAS MARET

2024

1. Screenshot Source Code

```
index.kt

//abstract class
abstract class EncryptorDecryptor {
    abstract fun encrypt(text:String):String
    abstract fun decrypt(text:String):String
}

//inheritance
class caesarCipher(key: Int = 23): EncryptorDecryptor(){

    //collection (Map)
    private val alphabet_encrypt = mutableMapOf<Char, Char>()
    private val number_encrypt = mutableMapOf<Int, Int>()
    private val alphabet_decrypt = mutableMapOf<Char, Char>()
    private val number_decrypt = mutableMapOf<Int, Int>()

    init{
        //lambda expressions
        (65..90).forEach { num →
            alphabet_encrypt[num.toChar()] = (((num % 65) + key) % 26 + 65).toChar()
            alphabet_decrypt[num.toChar()] = (((num % 65) - key + 26) % 26 + 65).toChar()
        }
        (48..57).forEach { num →
            number_encrypt[num % 48] = (((num % 48) + key) % 10 + 48) % 48
            number_decrypt[num % 48] = (((num % 48) - key + 10) % 10 + 48) % 48
        }
    }

    override fun encrypt(text:String):String{
        var encryptedString : String = ""

        // Control flow (for loop)
        for(letter in text){
            if(letter.isUpperCase()){
                encryptedString += alphabet_encrypt[letter]
            }else if(letter.isLowerCase()){
                encryptedString += alphabet_encrypt[letter.uppercaseChar()]?.lowercaseChar()
            }else if(letter.isDigit()){
                encryptedString += number_encrypt[letter.digitToInt()]
            }else{
                encryptedString += letter
            }
        }

        return encryptedString
    }

    override fun decrypt(text:String):String{
        var decryptedString : String = ""

        // Control flow (for loop)
        for(letter in text){
            if(letter.isUpperCase()){
                decryptedString += alphabet_decrypt[letter]
            }else if(letter.isLowerCase()){
                decryptedString += alphabet_decrypt[letter.uppercaseChar()]?.lowercaseChar()
            }else if(letter.isDigit()){
                decryptedString += number_decrypt[letter.digitToInt()]
            }else{
                decryptedString += letter
            }
        }

        return decryptedString
    }
}
```

```

fun main(){
    println("THIS IS SIMPLE TEXT ENCRYPTOR")
    var key : Int?
    do{
        print("Enter the encryption key : ")
        key = readLine()!!.toIntOrNull()
        if(key == null) println("Not a valid number, try again")
    }while(key == null) // Control flow (do-while loop)

    var encryptor : caesarCipher = caesarCipher(key)

    var choice : Int?
    while(true){
        println("\n1. Encrypt\n2. Decrypt\n3. Quit")
        do{
            print("What do you want to do : ")
            choice = readLine()!!.toIntOrNull()
            if(choice == null) println("Not a valid number, try again")
        }while(choice == null) // Control flow (do-while loop)

        //Control flow (when wxpression)
        when(choice){
            1 → {
                print("Enter the text to be encrypted : ")
                var encryptedText = encryptor.encrypt(readLine()!!)
                println("Encrypted text : ${encryptedText}")
            }
            2 → {
                print("Enter the text to be decrypted : ")
                var decryptedText = encryptor.decrypt(readLine()!!)
                println("Decrypted text : ${decryptedText}")
            }
            3 → break
        }
    }
}

```

Source code ini merupakan source code yang digunakan untuk praktikum minggu 2 saya.
Source code ini mencakup materi berikut

A. Control flow

```

// Control flow (for loop)
for(letter in text){
    if(letter.isUpperCase()){
        encryptedString += alphabet_encrypt[letter]
    }else if(letter.isLowerCase()){
        encryptedString += alphabet_encrypt[letter.uppercaseChar()]?.lowercaseChar()
    }else if(letter.isDigit()){
        encryptedString += number_encrypt[letter.digitToInt()]
    }else{
        encryptedString += letter
    }
}

var key : Int?
do{
    print("Enter the encryption key : ")
    key = readLine()!!.toIntOrNull()
    if(key == null) println("Not a valid number, try again")
}while(key == null) // Control flow (do-while loop)

```

```

var choice : Int?
while(true){
    println("\n1. Encrypt\n2. Decrypt\n3. Quit")
    do{
        print("What do you want to do : ")
        choice = readLine()!!.toIntOrNull()
        if(choice == null) println("Not a valid number, try again")
    }while(choice==null) // Control flow (do-while loop)

    //Control flow (when wxpression)
    when(choice){
        1 → {
            print("Enter the text to be encrypted : ")
            var encryptedText = encryptor.encrypt(readLine()!!)
            println("Encrypted text : ${encryptedText}")
        }
        2 → {
            print("Enter the text to be decrypted : ")
            var decryptedText = encryptor.decrypt(readLine()!!)
            println("Decrypted text : ${decryptedText}")
        }
        3 → break
    }
}

```

Control Flow merupakan sebuah cara seorang pemrogram mengatur alur dari sebuah program berdasarkan kondisi yang terjadi saat program tersebut berjalan. Disini terdapat banyak sekali penggunaan control flow, seperti while loop, for loop, ataupun when expression.

B. Inheritance

```

//inheritance
class caesarCipher(key: Int = 23): EncryptorDecryptor(){

    //collection (Map)
    private val alphabet_encrypt = mutableMapOf<Char, Char>()
    private val number_encrypt = mutableMapOf<Int, Int>()
    private val alphabet_decrypt = mutableMapOf<Char, Char>()
    private val number_decrypt = mutableMapOf<Int, Int>()

    init{
        //lambda expressions
        (65..90).forEach { num →
            alphabet_encrypt[num.toChar()] = (((num % 65) + key) % 26 + 65).toChar()
            alphabet_decrypt[num.toChar()] = (((num % 65) - key + 26) % 26 + 65).toChar()
        }
        (48..57).forEach { num →
            number_encrypt[num % 48] = (((num % 48) + key) % 10 + 48) % 48
            number_decrypt[num % 48] = (((num % 48) - key + 10) % 10 + 48) % 48
        }
    }

    override fun encrypt(text:String):String{
        var encryptedString : String = ""

        // Control flow (for loop)
        for(letter in text){
            if(letter.isUpperCase()){
                encryptedString += alphabet_encrypt[letter]
            }else if(letter.isLowerCase()){
                encryptedString += alphabet_encrypt[letter.toUpperCaseChar()]?.toLowerCaseChar()
            }else if(letter.isDigit()){
                encryptedString += number_encrypt[letter.digitToInt()]
            }else{
                encryptedString += letter
            }
        }

        return encryptedString
    }

    override fun decrypt(text:String):String{
        var decryptedString : String = ""

        // Control flow (for loop)
        for(letter in text){
            if(letter.isUpperCase()){
                decryptedString += alphabet_decrypt[letter]
            }else if(letter.isLowerCase()){
                decryptedString += alphabet_decrypt[letter.toUpperCaseChar()]?.toLowerCaseChar()
            }else if(letter.isDigit()){
                decryptedString += number_decrypt[letter.digitToInt()]
            }else{
                decryptedString += letter
            }
        }

        return decryptedString
    }
}

```

Inheritance atau pewarisan adalah salah satu pilar yang ada dalam 4 pilar OOP. Dengan menerapkan inheritance, kita dapat menghemat kode dan meminimalisir terjadinya boilerplate pada kode yang kita buat. Disini class caesarCipher merupakan hasil penurunan dari abstract class EncryptorDecryptor, dan dilakukan override pada method encrypt dan decrypt.

C. Abstract class

```
//abstract class
abstract class EncryptorDecryptor {
    abstract fun encrypt(text:String):String
    abstract fun decrypt(text:String):String
}

//inheritance
```

Abstract class adalah sebuah kelas yang tidak dapat diinstansiasi secara langsung dan dapat memiliki metode abstrak (abstract method). Metode abstrak adalah metode yang tidak memiliki implementasi di dalam kelas abstrak, tetapi harus diimplementasikan oleh kelas turunannya (subclass). Abstraksi digunakan ketika kita ingin memiliki kelas dasar yang menyediakan kerangka kerja umum, tetapi membiarkan detail implementasi khusus ditentukan oleh kelas turunannya. Disini terdapat abstract class EncryptorDecryptor yang didalamnya terdapat 2 abstract method, yakni encrypt() dan decrypt().

D. Collections

```
//collection (Map)
private val alphabet_encrypt = mutableMapOf<Char, Char>()
private val number_encrypt = mutableMapOf<Int, Int>()
private val alphabet_decrypt = mutableMapOf<Char, Char>()
private val number_decrypt = mutableMapOf<Int, Int>()
```

Collections digunakan untuk menyimpan dan memanipulasi kelompok objek atau data. Terdapat dua jenis collections, yaitu mutable dan immutable. Disini terdapat 4 buah collection bertipe mutable dengan Map sebagai struktur datanya, Map adalah kumpulan pasangan kunci-nilai di mana setiap kunci unik dan memiliki nilai yang terkait. Keempat map ini digunakan untuk memetakan setiap huruf angka dengan hasil enkripsinya.

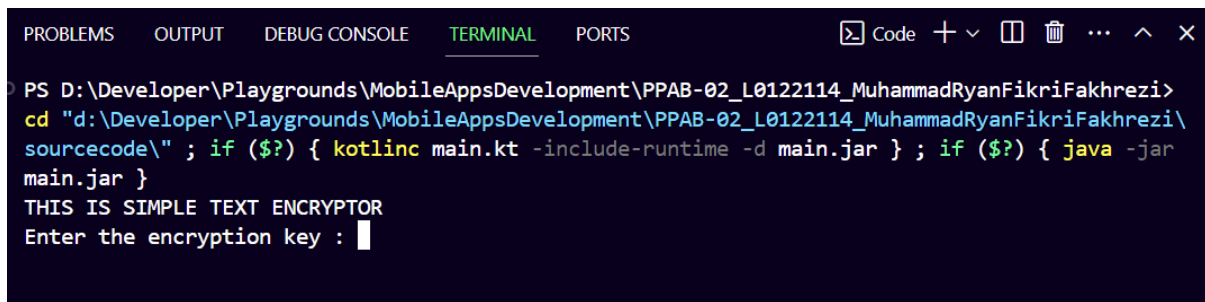
E. Lambda Expression

```
init{
    //lambda expressions
    (65..90).forEach { num →
        alphabet_encrypt[num.toChar()] = (((num % 65) + key) % 26 + 65).toChar()
        alphabet_decrypt[num.toChar()] = (((num % 65) - key + 26) % 26 + 65).toChar()
    }
    (48..57).forEach { num →
        number_encrypt[num % 48] = (((num % 48) + key) % 10 + 48) % 48
        number_decrypt[num % 48] = (((num % 48) - key + 10) % 10 + 48) % 48
    }
}
```

Dalam Kotlin, lambda expression adalah cara untuk membuat fungsi anonim yang dapat ditransmisikan sebagai argumen, disimpan dalam variabel, atau dikembalikan dari fungsi lain. Lambda expression sangat berguna untuk menyederhanakan kode dan mengekspresikan operasi yang singkat dan mudah dibaca. Disini, untuk setiap elemen pada Kumpulan angka dengan range 65-90/48-57 akan dilakukan proses pemasangan pasangan key-value pada map yang nantinya akan digunakan untuk mengenkripsi ataupun mendeskripsi pesan

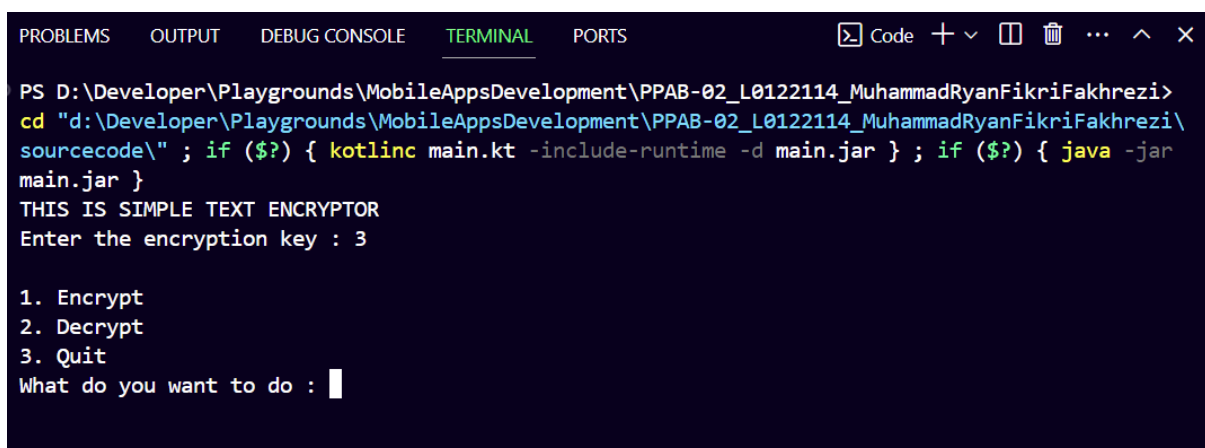
2. Screenshot Terminal

Berikut adalah hasil pada terminal apabila source code dijalankan.



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS D:\Developer\Playgrounds\MobileAppsDevelopment\PPAB-02_L0122114_MuhammadRyanFikriFakhrezi>
cd "d:\Developer\Playgrounds\MobileAppsDevelopment\PPAB-02_L0122114_MuhammadRyanFikriFakhrezi\
sourcecode\" ; if ($?) { kotlinc main.kt -include-runtime -d main.jar } ; if ($?) { java -jar
main.jar }
THIS IS SIMPLE TEXT ENCRYPTOR
Enter the encryption key : 
```


Disini, program meminta interaksi user untuk memasukkan key yang akan digunakan untuk proses enkripsi/dekripsi.



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS D:\Developer\Playgrounds\MobileAppsDevelopment\PPAB-02_L0122114_MuhammadRyanFikriFakhrezi>
cd "d:\Developer\Playgrounds\MobileAppsDevelopment\PPAB-02_L0122114_MuhammadRyanFikriFakhrezi\
sourcecode\" ; if ($?) { kotlinc main.kt -include-runtime -d main.jar } ; if ($?) { java -jar
main.jar }
THIS IS SIMPLE TEXT ENCRYPTOR
Enter the encryption key : 3

1. Encrypt
2. Decrypt
3. Quit
What do you want to do : 
```

Kemudian, User akan disuguhi menu yang terdiri dari 3 opsi: enkripsi, dekripsi, atau quit.



```
1. Encrypt
2. Decrypt
3. Quit
What do you want to do : 1
Enter the text to be encrypted : Ryan Fikri
Encrypted text : Ubdq Ilnul
```

Setelahnya, jika user memilih pilihan 1/2, user akan diminta memasukkan teks yang ingin dienkripsi/dekripsi. Kemudian program akan menampilkan teks hasil enkripsi/dekripsi. Kemudian user akan dikembalikan Kembali ke menu. Loop ini terus dijalankan semestara user tidak memilih opsi ketiga di menu.