

Pengenalan Pemrograman Kotlin



Pra-Praktikum

Tujuan :

- Mahasiswa mengenal apa itu Kotlin.
- Mahasiswa mengetahui perbedaan Kotlin dan Java.
- Mahasiswa mengetahui langkah-langkah menginstalasi IDE untuk membuat program Kotlin

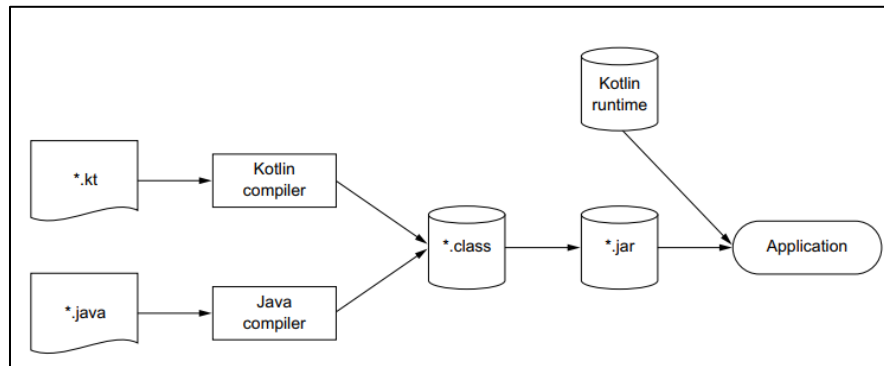
Output :

- Dasar Kotlin.
- Perbedaan pemrograman Kotlin dan Java.
- Melakukan instalasi IDE untuk membuat program Kotlin.

Dasar Kotlin

Kotlin adalah salah satu bahasa pemrograman yang dikenalkan oleh sebuah perusahaan IT yang terkemuka yaitu JetBrains. JetBrains adalah sebuah perusahaan yang mengembangkan sejumlah IDE yang populer seperti Android Studio IDE, IntelliJ IDEA dan masih banyak lagi. Kotlin berjalan pada Java Virtual Machine (JVM) seperti halnya bahasa pemrograman Java. Kotlin adalah bahasa pemrograman sumber terbuka (*open source*) yang mengkombinasikan konsep OOP (Object Oriented Programming) dan fitur fungsional kedalam sebuah platform yang unik.

Kotlin juga terpengaruh dari beberapa gaya bahasa pemrograman seperti Java, Swift, Scala, Groovy, Gosu dan lain-lain. Walaupun Kotlin memiliki sintak yang berbeda dengan Java sejatinya ketika dikompilasi, kode Kotlin sangat kompatibel dengan Java, hal ini berarti sebagian besar pustaka dukungan Java juga mendukung Kotlin.



Gambar : Arsitektur Kotlin

(Sumber : Jamerov. D dan Isakov. S, 2017)

File Kotlin berekstensi .kt di kompilasi menggunakan kompiler Kotlin untuk dikonversi menjadi *byte code* yang dapat dijalankan di JVM, sama halnya dengan Java yang melakukan konversi file .class menjadi *byte code* untuk dijalankan di JVM. Ketika Kotlin akan menargetkan JavaScript, kompiler kotlin akan merubah file .kt kedalam file ES5.1 dan akan menghasilkan kode yang kompatibel dengan JavaScript.

Sejak pertama kali diluncurkan Kotlin telah mengesankan para programmer sebagai bahasa pemrograman yang ringkas, mudah dibaca, fleksibel, *powerfull*, *statically typed* (tipe dari setiap ekspresi didalam program diketahui pada saat kompilasi), aman (memastikan kode tetap bersih dan mencegah *error*), pragmatis(sesuatu yang menjadi mudah untuk kita), memiliki penulisan sintak yang bersahabat serta sangat mendukung kode Java.

Perbedaan Kotlin dan Java

Berikut adalah beberapa perbedaan dasar antara Kotlin dan Java:

Kotlin Miliki	Java Miliki
Ekspresi <i>lambda</i> dan <i>inline functions</i>	Pengecekan eksepsi
Properties (<i>automatic implemented getter and setter</i>)	Tipe data primitve(<i>int, float, char, short, long, double, byte, boolean</i>)
Null-safety	Static member (<i>private static int</i>)

String template	Non-private attribute
Primary constructor	Genericity wildcard
Automatic data types detection	Ternary operator (?:)
Range expressions (1 ... 10)	
Operator overriding (ObjA + ObjB)	
Companion objects	

Instalasi IntelliJ IDEA

Berikut adalah langkah-langkah untuk instalasi IntelliJ IDEA :

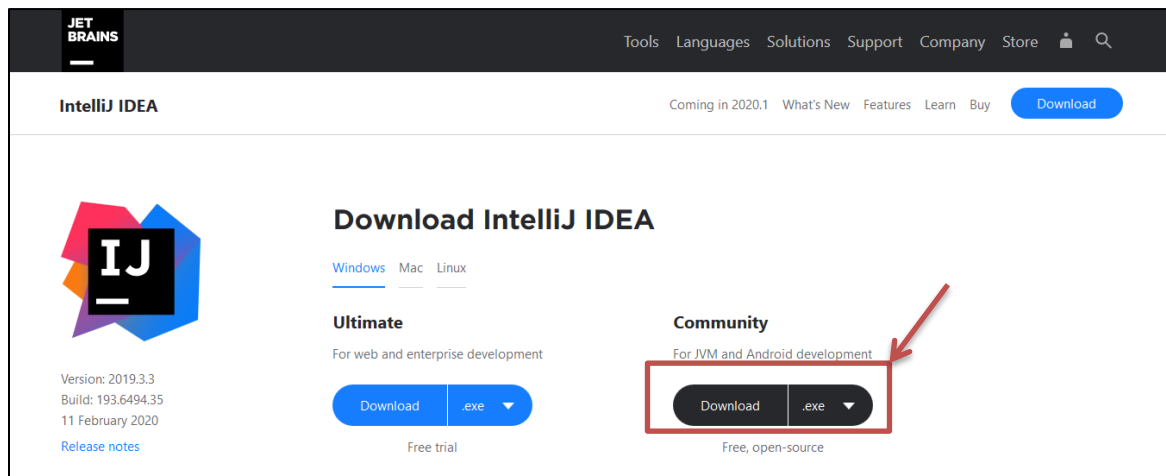
1. Sebelum melakukan instalasi IntelliJ IDEA silahkan download dan install terlebih dahulu Java Standard Edition Development Kit (JDK™) di alamat :

<https://www.oracle.com/java/technologies/javase-jdk8-downloads.html>

Catatan : Silahkan download dan install versi terbaru dari JDK.

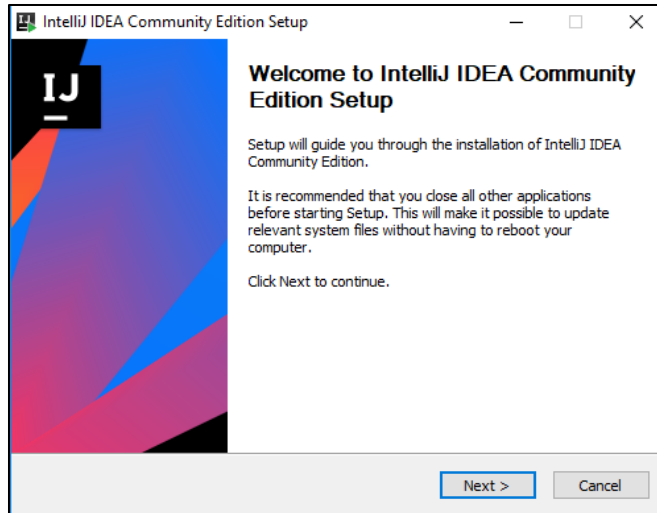
2. Bukalah alamat website resmi IntelliJ di alamat berikut :

<https://www.jetbrains.com/idea/download>

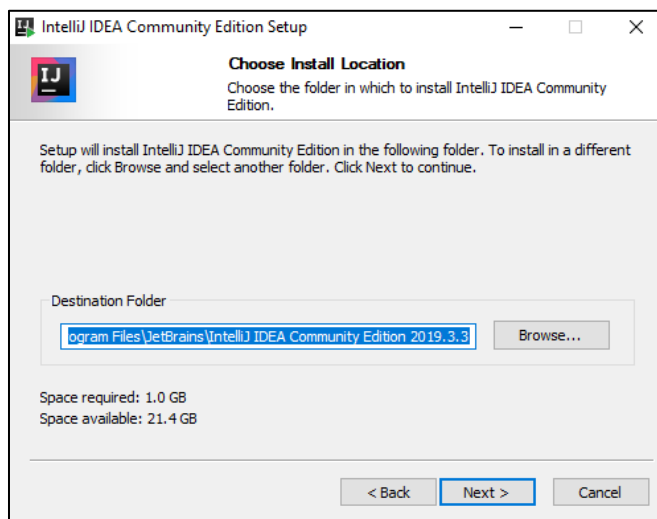


Terdapat 2 versi IntelliJ IDEA, Ultimate adalah versi berbayar yang menyediakan banyak fitur kepada pengembang dan versi Community yang merupakan versi gratis. Silahkan download IntelliJ IDEA Community Edition (versi: 2019.3.3).

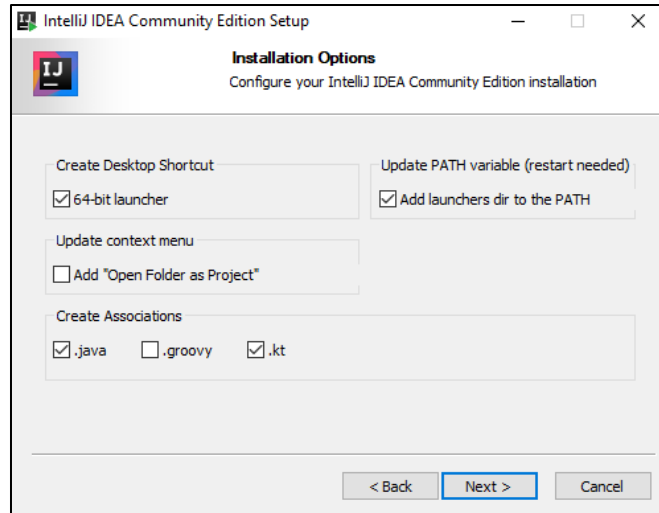
3. Setelah mendownload IntelliJ IDEA, lakukan instalasi IntelliJ IDEA seperti melakukan instalasi perangkat lunak pada umumnya. Klik next untuk melanjutkan instalasi.



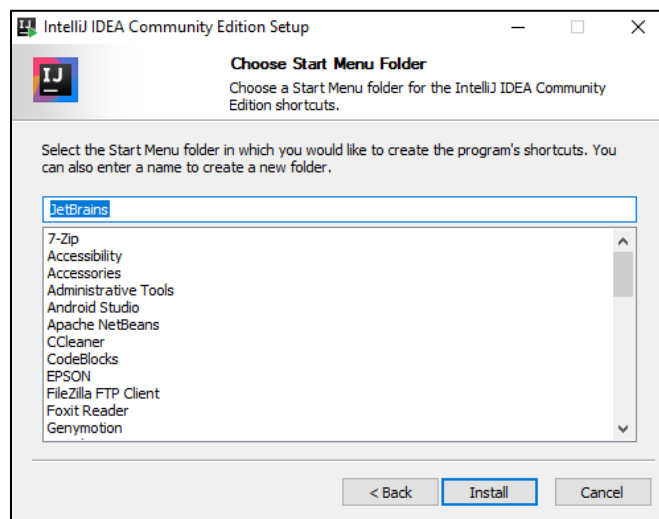
4. Tentukan folder tujuan untuk menampung file instalasi IntelliJ IDEA. Biarkan pengaturan folder tujuan secara default yaitu di “C:\Program Files\ ...”. Klik Next > untuk melanjutkan.



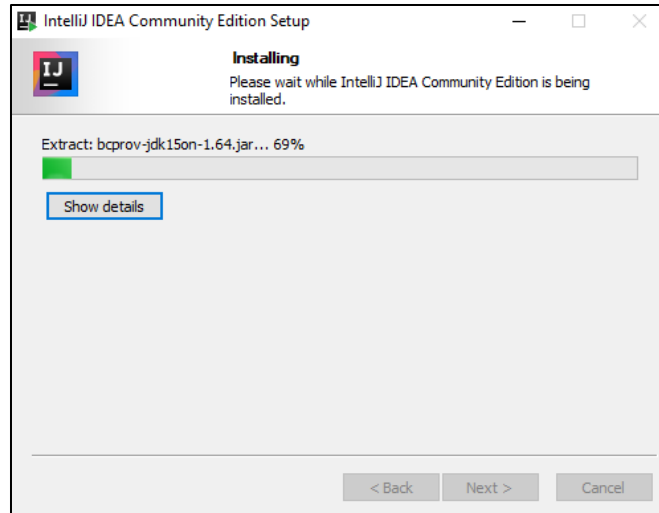
5. Pada opsi instalasi lakukan konfigurasi seperti berikut ini:



6. Biarkan konfigurasi default pada pilihan Start Menu Folder, lanjutkan klik Install untuk memulai instalasi.



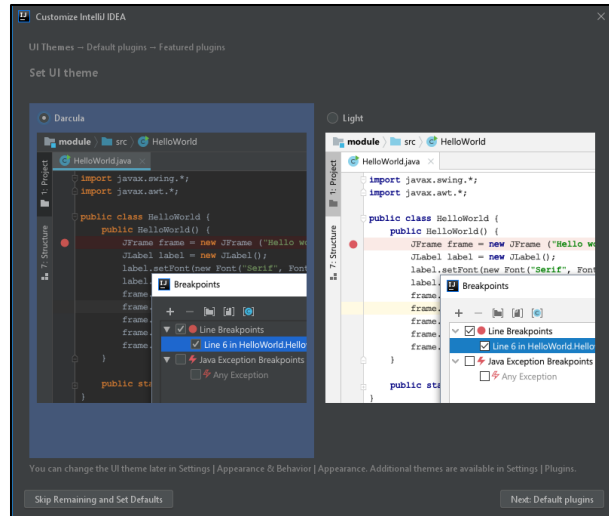
7. Berikut adalah progress menyalin file instalasi.



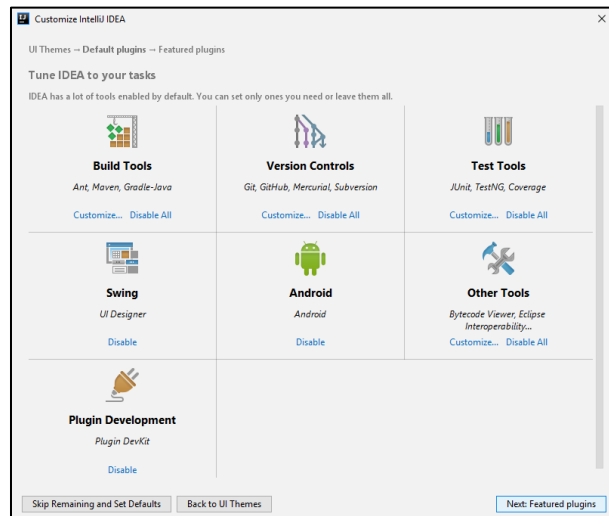
8. Setelah selesai melakukan instalasi jalankan IntelliJ IDEA maka akan tampil form import setting. Jika anda sudah memiliki instalasi folder proyek sebelumnya maka bisa diimport pada pilihan import. Klik next untuk melanjutkan. Berikut adalah tampilan splash screen IntelliJ.



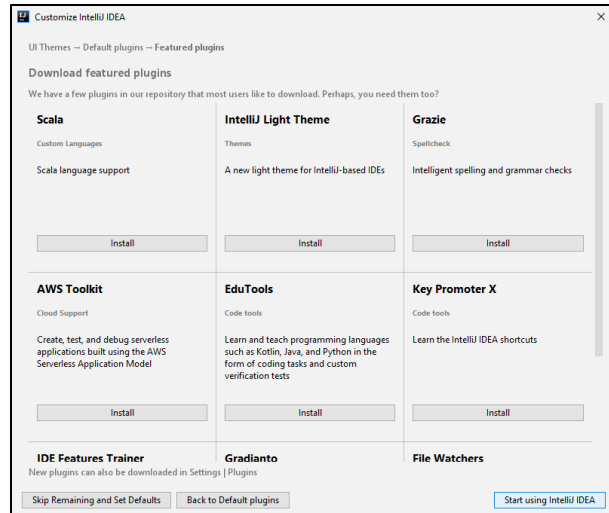
9. Selanjutnya pilihlah UI Theme sesuai dengan keinginan. Pilih Light dan Klik Next Default plugins.



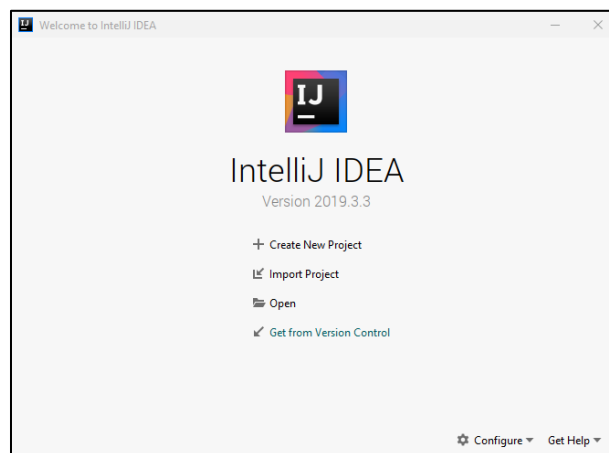
10. Biarkan fitur tools diatur secara default klik Next Featured plugins.



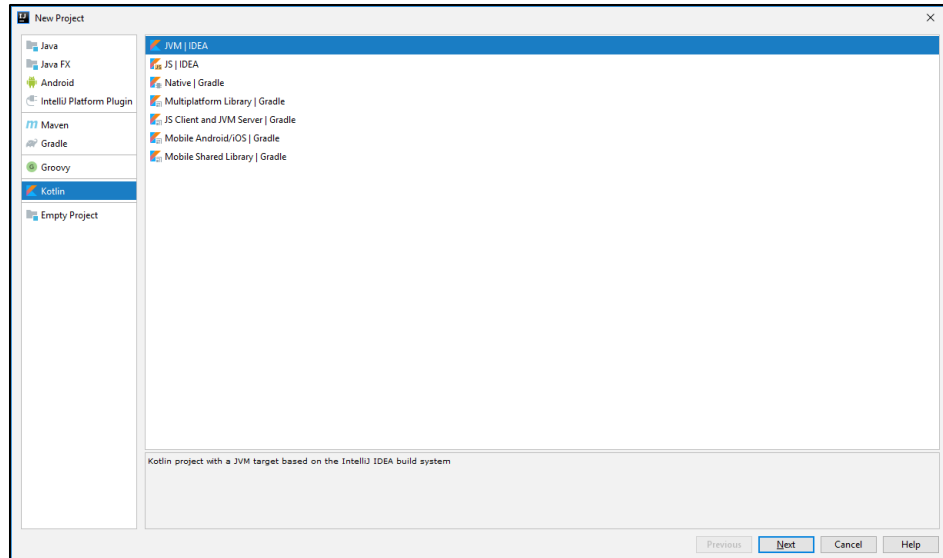
11. Jika ingin mendownload fitur plugin tambahan bisa ditambahkan pada antarmuka berikut, biarkan saja hanya fitur default yang tampil di IDE IntelliJ. Klik Start using IntelliJ IDEA untuk memulai menggunakan IDE.



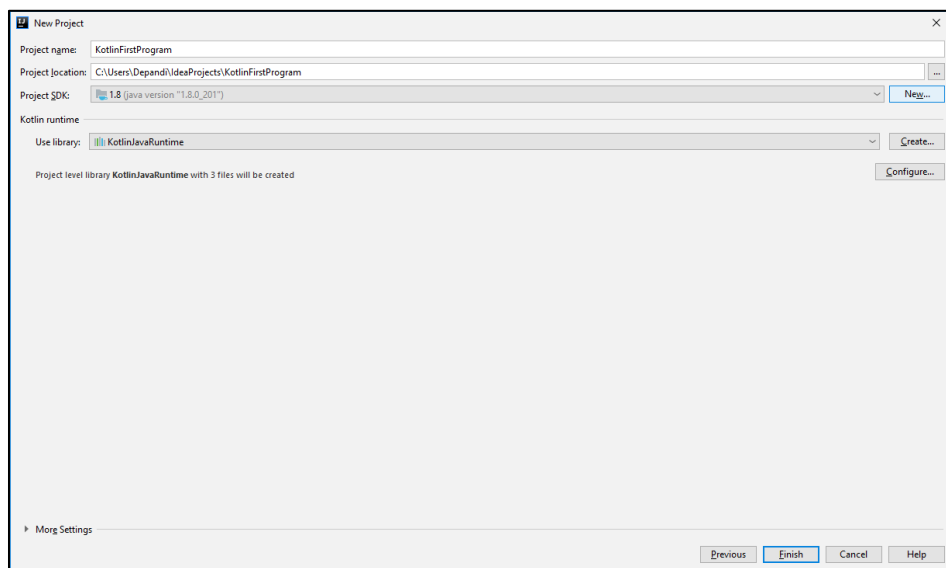
12. Setelah IDE berhasil terinstal silahkan mengujinya dengan membuat sebuah project baru dengan cara klik Create New Project.



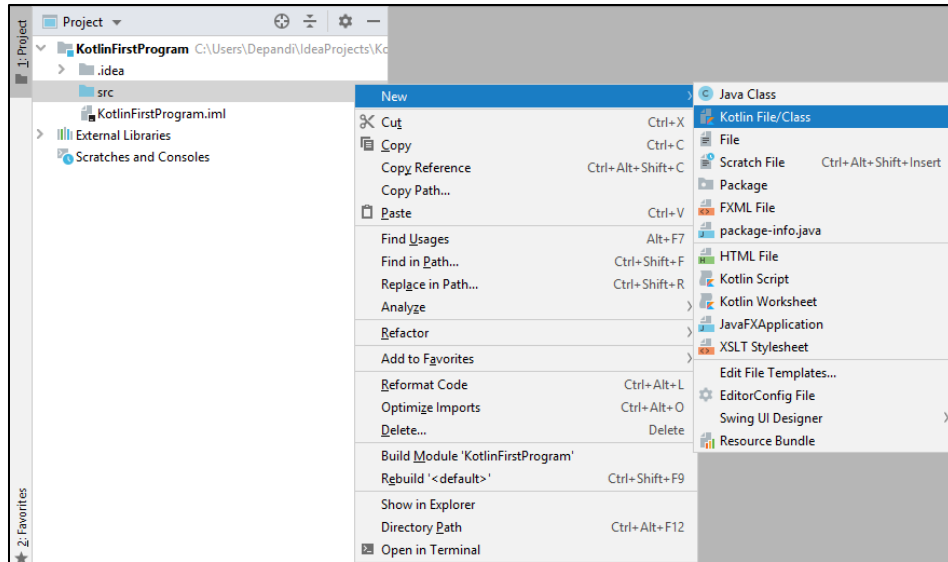
13. Pilihlah Kotlin pada pilihan jenis project side bar dan JVM | IDEA sebagai build system. Klik Next untuk melanjutkan.



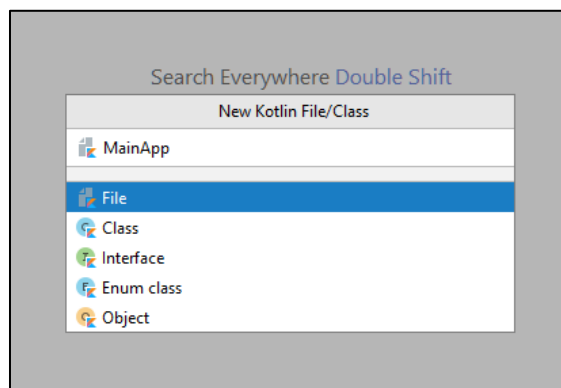
14. Masukkan nama project “KotlinFirstProgram” dan klik Finish untuk mengakhiri pengaturan pembuatan project.



15. Langkah berikutnya membuat file MainApp pada folder src project. Klik kanan pada folder src → New → Kotlin File/Class



16. Pada kotak dialog pilih File dan masukkan MainApp pada input teks yang disediakan tekan Enter untuk melanjutkan



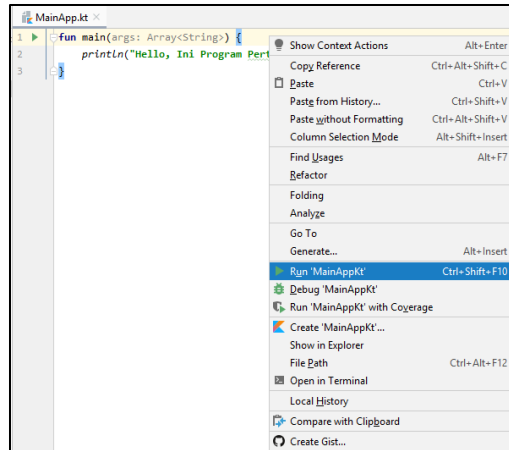
17. Masukkan sintaks kode program berikut pada file MainApp.

```
fun main(args: Array<String>) {
    println("Hello, Ini Program Pertama Saya di Kotlin!")
}
```

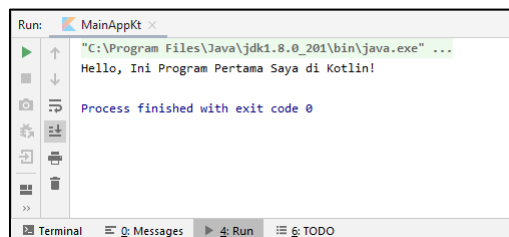
Catatan :

- ✂ **fun** adalah sebuah keyword yang menandakan sebuah fungsi di kotlin
- ✂ **fun main()** adalah fungsi main diaplikasi Kotlin dimana sebagai titik awal ketika program dijalankan.
- ✂ **println()** adalah fungsi dasar untuk mencetak string ke layar.

18. Jalankan program dengan cara klik kanan pada kode editor MainApp dan klik pilihan Run “MainAppKt”.



19. Tunggu proses kompilasi selesai dan perhatikan hasilnya, maka akan tampak seperti berikut.



20. Sampai disini anda sudah berhasil melakukan instalasi IntelliJ IDEA dan membuat proyek pertama Kotlin.

Tugas Pra-Praktikum :

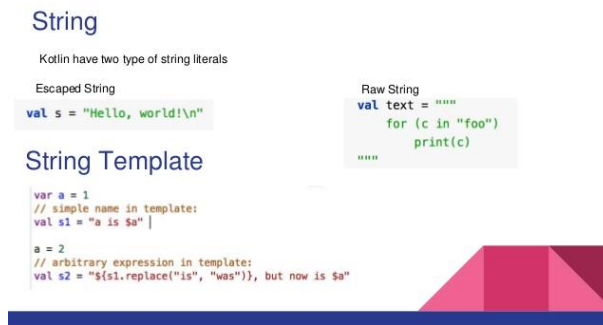
- Cobalah untuk mencetak kalimat berikut di layar output : “Selamat belajar Kotlin”

Daftar Referensi

<https://www.oracle.com/java/>

<https://www.jetbrains.com/idea/>

Dasar Pemrograman Kotlin



Praktikum 1

Tujuan :

Mahasiswa mengenal dan mengimplementasikan :

- Elemen Program (Literal, Expression and Statement, Keyword, Whitespaces, Operator, Blok, Komentar, Variabel, Var dan Val)
- Tipe Data Number (Byte, Short, Int, dan Long)
- Tipe Data Floating Point Number (Float dan Double)
- Constants Literal dan Underscore Literal dalam tipe data Numeric
- Representasi Numeric
- Konversi Eksplisit dalam tipe data Numeric
- Tipe Data Character (Char)
- Tipe Data Boolean (true or false)
- Tipe Data String
- String Templates
- Fungsi Anggota pada String
- Penerapan String
- Perintah Masukkan di Kotlin
- Program Kalkulator Sederhana

Output :

- Contoh program praktikum.

Elemen Program Kotlin

Literal

Kotlin menyediakan beberapa literal untuk tipe dasar (Numbers, Character, Boolean, String). Literal di Kotlin akan memberikan tipe data secara otomatis setelah program dikompilasi sesuai dengan nilai yang cocok untuk diberikan pada suatu variabel.

Contoh literal di Kotlin:

```
fun main(args: Array<String>) {  
    var intLiteral = 5  
    var doubleLiteral = .02  
    var stringLiteral = "Hello"  
    var charLiteral = '1'  
    var boolLiteral = true  
    println("intLiteral = " + intLiteral)  
    println("doubleLiteral = " + doubleLiteral)  
    println("stringLiteral = " + stringLiteral)  
    println("charLiteral = " + charLiteral)  
    println("boolLiteral = " + boolLiteral)  
}
```

Output:

```
"C:\Program Files\Java\jdk1.8.0_201\bin\java.exe" ...  
intLiteral = 5  
doubleLiteral = 0.02  
stringLiteral = Hello  
charLiteral = 1  
boolLiteral = true  
  
Process finished with exit code 0
```

Expression and Statement

Ekspresi adalah kombinasi dari operator, fungsi, nilai literal, variabel atau konstanta, dan selalu merubah nilai.

Contoh : var x = 20 % 4

Kita tidak bisa melewati operasi assignment sebagai argument didalam loop statement seperti while :

Contoh : while ((rem = a % b) != 0) { }

Operasi assignment dalam perulangan while `rem = a % b` tidak sah digunakan dalam Kotlin sebab kotlin menganggap hal tersebut sebagai statement di Kotlin berbeda dengan Java menganggap hal tersebut sebagai ekspresi.

Keyword

Keyword adalah istilah cadangan yang memiliki arti khusus bagi compiler, dan juga tidak dapat digunakan sebagai pengenalan (*identifier*) untuk apapun elemen program seperti nama kelas, variabel, fungsi dan interfaces.

Kotlin memiliki hard, soft dan modifier keyword. Hard keyword selalu diterjemahkan sebagai keyword dan tidak dapat digunakan sebagai *identifier*. Contohnya adalah : break, class, continue, do, else, false, while, this, throw, try, super, dan when.

Soft keyword dikenali sebagai kata-kata cadangan, dalam konteks tertentu mereka dapat dipakai, selain itu ia juga dapat digunakan sebagai regular identifier. Contohnya adalah : file, finally, get, import, receiver, set, constructor, delegate, get, by, dan where.

Modifier keyword dikenali sebagai kata cadangan didalam daftar modifier, selain itu ia juga dapat digunakan sebagai modifier. Contohnya adalah : abstract, actual, annotation, companion, enum, final, infix, inline, lateinit, operator, dan open.

Whitespaces

Seperti Java, Kotlin juga merupakan bahasa pemrograman yang ditokenisasi. Whitespace tidak signifikan mempengaruhi program dan dengan aman dapat diacuhkan. Kita dapat menggunakan gaya penulisan yang terlihat baik seperti berikut:

```
fun main(args: Array<String>) {  
    println("Hello, world!")  
}
```

dan juga dapat ditulis seperti berikut ini:

```
fun main(args: Array<String>) { println("Hello, world!") }
```

Kompiler tetap dapat menjalankan kedua program diatas. Tetapi menurut kemudahan cara membacanya maka kode program pertama lebih baik dan mudah dibaca dibandingkan yang kedua.

Operator

Seperti halnya Java, Kotlin juga mendukung beberapa operator untuk melakukan operasi tertentu.

Jenis Operator atau Simbol	Operator atau Simbol
Arithmetic operator	+, -, *, /, %
Equal symbol	=

Assignment operator	+=, -=, *=, /=, %=
Logical operator	&&, , !, 'and', 'or', 'not'
Equality operator	==, !=
Referential operator (memeriksa referensi 2 objek)	===, !==
Comparison operator	<, >, <=, >=
Index access operator (digunakan untuk mengakses elemen array dan koleksi)	[], [,]

Blok dan Komentar

Blok digunakan untuk menulis ikatan sebuah statement yang dikelompokkan bersama menjadi satu bagian. Contoh:

```
fun main(args: Array<String>) {
    println("Hello, Ini Program Pertama Saya di Kotlin!")
}
```

Blok ditandai dengan tanda buka dan tutup kurung kurawal.

Comment adalah baris komentar pada program yang tidak akan dieksekusi oleh compiler.

Contoh:

```
fun main(args: Array<String>) {
    println("Hello, Ini Program Pertama Saya di Kotlin!")
    //Contoh komentar satu baris
    /*
    Contoh komentar multi baris
    */
    /**
     * Kotlin Doc --> baris ini untuk mendokumentasikan kode program
     */
}
```

Variabel

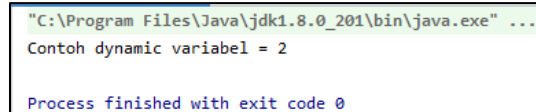
Variabel adalah sebuah tempat/wadah didalam memori komputer untuk menyimpan data seperti data nama pengguna, kata sandi, data waktu, jumlah makanan dan lain-lain sebagainya. Tempat yang digunakan oleh suatu variabel memiliki ukuran yang berbeda-beda didalam memori komputer tergantung dari jenis tipe data yang digunakan pada sebuah variabel.

Terdapat 2 *type system* dalam variabel yaitu statis dan dinamis. Tipe system yang dinamis tidak memerlukan adanya deklarasi tipe data pada sebuah variabel, compiler secara otomatis akan

mengenali tipe data dari sebuah variabel berdasarkan nilai yang diberikan kepadanya. Contoh kode program:

```
fun main(args: Array<String>) {  
    var dynamicVar = 2  
    println("Contoh dynamic variabel = " + dynamicVar)  
}
```

Output:

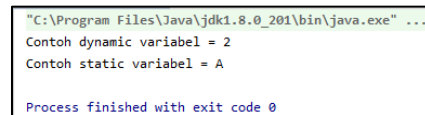


```
"C:\Program Files\Java\jdk1.8.0_201\bin\java.exe" ...  
Contoh dynamic variabel = 2  
  
Process finished with exit code 0
```

Berdasarkan contoh diatas maka secara otomatis compiler Kotlin akan mengenali bahwa variabel `dynamicVar` bertipe `Int`, hal ini identik dengan gaya bahasa pemrograman PHP dan Ruby. Sebaliknya tipe system yang statis mengharuskan kita mendeklarasikan jenis tipe data tertentu dalam sebuah variabel dan tipe data tersebut tidak bisa diubah selama program dieksekusi. Contoh kode program:

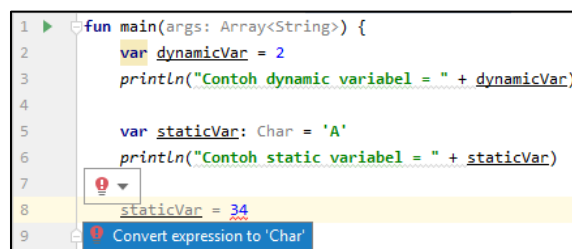
```
fun main(args: Array<String>) {  
    var dynamicVar = 2  
    println("Contoh dynamic variabel = " + dynamicVar)  
  
    var staticVar: Char = 'A'  
    println("Contoh static variabel = " + staticVar)  
}
```

Output:



```
"C:\Program Files\Java\jdk1.8.0_201\bin\java.exe" ...  
Contoh dynamic variabel = 2  
Contoh static variabel = A  
  
Process finished with exit code 0
```

Berdasarkan contoh kode diatas dapat diketahui bahwa variabel `staticVar` memiliki tipe data `Char`. Ketika kita memberikan nilai 34 pada `staticVar` maka compiler akan memberikan pesan error.



```
1 fun main(args: Array<String>) {  
2     var dynamicVar = 2  
3     println("Contoh dynamic variabel = " + dynamicVar)  
4  
5     var staticVar: Char = 'A'  
6     println("Contoh static variabel = " + staticVar)  
7  
8     staticVar = 34  
9 }
```

Convert expression to 'Char'

Var vs Val

Kata kunci `var` di Kotlin digunakan untuk mendeklarasikan variabel yang nilainya dapat berubah sepanjang program dieksekusi. Sedangkan kata kunci `val` digunakan untuk mendeklarasikan variabel konstan yang nilainya tidak akan berubah selama program dieksekusi. Variabel dengan kata kunci `val` hanya bisa diberikan nilai sekali saja ketika program dijalankan, untuk *assigned value* yang kedua dan selanjutnya maka variabel dengan kata kunci `val` tersebut akan dianggap suatu kesalahan oleh compiler.

Contoh kode program menggunakan `var` variabel:

```
fun main(args: Array<String>) {  
    var varVariabel: Int  
    varVariabel = 56 //assigning first value to varVariabel  
    println("Pemberian nilai pertama pada varVariabel = "+varVariabel)  
    varVariabel = 78 //assigning second value to varVariabel  
    println("Pemberian nilai kedua pada varVariabel = "+varVariabel)  
}
```

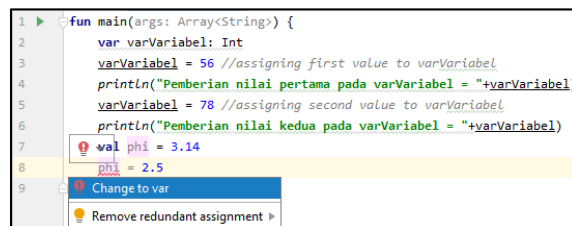
Output:

```
"C:\Program Files\Java\jdk1.8.0_201\bin\java.exe" ...  
Pemberian nilai pertama pada varVariabel = 56  
Pemberian nilai kedua pada varVariabel = 78  
  
Process finished with exit code 0
```

Contoh kode program menggunakan `val` variabel:

```
fun main(args: Array<String>) {  
    var varVariabel: Int  
    varVariabel = 56 //assigning first value to varVariabel  
    println("Pemberian nilai pertama pada varVariabel = "+varVariabel)  
    varVariabel = 78 //assigning second value to varVariabel  
    println("Pemberian nilai kedua pada varVariabel = "+varVariabel)  
    val phi = 3.14  
    phi = 2.5  
}
```

Output:



```
1 fun main(args: Array<String>) {  
2     var varVariabel: Int  
3     varVariabel = 56 //assigning first value to varVariabel  
4     println("Pemberian nilai pertama pada varVariabel = "+varVariabel)  
5     varVariabel = 78 //assigning second value to varVariabel  
6     println("Pemberian nilai kedua pada varVariabel = "+varVariabel)  
7     val phi = 3.14  
8     phi = 2.5  
9 }
```

Ketika kita merubah nilai variabel `val phi` untuk kedua kalinya maka akan terdapat pesan kesalahan agar kita menggunakan `var` variabel.

Tipe Data

Tipe data di Kotlin terdiri dari Byte, Short, Int, dan Long untuk bilangan bulat (Number), Double dan Float untuk bilangan desimal atau real (Floating-point), Char untuk karakter, Boolean untuk kondisional (true/false), String untuk array karakter, Array yang merupakan koleksi data dengan tipe data yang homogen, Collection dan Range.

Number (Byte, Short, Int, Long, Float, Double)

Berikut adalah spesifikasi 4 tipe data bilangan bulat (Number) di Kotlin :

Data Type	Size (bits)	Min Value	Max Value
Byte	8	-128	127
Short	16	-32768	32767
Int	32	-2,147,483,648 (-2^{31})	2,147,483,647 ($2^{31} - 1$)
Long	64	-9,223,372,036,854,775,808 (-2^{63})	9,223,372,036,854,775,807 ($2^{63} - 1$)

(Sumber : <https://kotlinlang.org/docs/reference/basic-types.html>)

Contoh kode program menggunakan tipe data bilangan bulat:

```
fun main(args: Array<String>) {  
    val minByte: Byte = -128  
    val maxByte: Byte = 127 // 8bit  
    val minShort: Short = -32768  
    val maxShort: Short = 32767 // 16bit  
    val minInt: Int = -2147483648  
    val maxInt: Int = 2147483647 // 32bit  
    val minLong: Long = -9223372036854775807  
    val maxLong: Long = 9223372036854775807 // 64bit  
  
    println("minByte:" + minByte)  
    println("maxByte:" + maxByte)  
    println("minShort:" + minShort)  
    println("maxShort:" + maxShort)  
    println("minInt:" + minInt)  
    println("maxInt:" + maxInt)  
    println("minLong:" + minLong)  
    println("maxLong:" + maxLong)  
}
```

Output:

```
"C:\Program Files\Java\jdk1.8.0_201\bin\java.exe" ...
minByte:-128
maxByte:127
minShort:-32768
maxShort:32767
minInt:-2147483648
maxInt:2147483647
minLong:-9223372036854775807
maxLong:9223372036854775807
```

Cobalah untuk mengubah nilai maxByte menjadi 128 perhatikan apa yang terjadi. Cobalah untuk menganalisa kode program.

```
1 fun main(args: Array<String>) {
2     val minByte: Byte = -128
3     val maxByte: Byte = 128 // 8bit
4     Change type of 'maxByte' to 'Int' 2768
5     Convert expression to 'Byte' 767 // 16bit
```

Berikut adalah spesifikasi 2 tipe data bilangan desimal (*floating-point*) di Kotlin :

Data Type	Size (bits)	Decimal digits
Float	32	6-7
Double	64	15-16

(Sumber : <https://kotlinlang.org/docs/reference/basic-types.html>)

Contoh kode program menggunakan tipe data bilangan decimal (*floating-point*):

```
fun main(args: Array<String>) {
    val maxFloat: Float = 9.123456789f
    val maxDouble: Double = 9.123456789

    println("maxFloat:" + maxFloat)
    println("maxDouble:" + maxDouble)
}
```

Output:

```
"C:\Program Files\Java\jdk1.8.0_201\bin\java.exe" ...
maxFloat:9.123457
maxDouble:9.123456789
Process finished with exit code 0
```

Cobalah untuk memperhatikan hasil presisi digit yang terekam oleh variabel maxFloat yaitu hanya 6 digit (kelebihan digit akan dibulatkan) sedangkan untuk var maxDouble presisi digit mencapai 10 digit. Untuk memberikan nilai sebagai float maka harus memberikan akhiran f atau F pada akhir literal.

Constants Literal

Contoh kode program menggunakan literal konstan:

```

fun main(args: Array<String>) {
    val floatLiteral : Float = 178.95F //Literal Float
    println("Contoh Literal Float : " + floatLiteral)
    val oneBillion: Long = 1000000000L //Literal Long
    println("Contoh Literal Long : " + oneBillion)
    val valHeksa = 0x0F //Literal heksadesimal diawali dengan 0x
    println("Contoh Literal Heksadesimal : " + valHeksa)
    val valBinary = 0b00001011 //Literal binary diawali dengan 0b
    println("Contoh Literal Binary : " + valBinary)
    val notasiKonvensional = 123.5e10
    println("Contoh Literal Konvensional Floating Point Number : " + notasiKonvensional)
}

```

Output:

```

"C:\Program Files\Java\jdk1.8.0_201\bin\java.exe" ...
Contoh Literal Float : 178.95
Contoh Literal Long : 1000000000
Contoh Literal Heksadesimal : 15
Contoh Literal Binary : 11
Contoh Literal Konvensional Floating Point Number : 1.235E12

Process finished with exit code 0

```

Underscore dalam Literal

Selain memiliki literal konstan Kotlin juga dapat menggunakan *underscore* dalam literal, untuk memberikan kemudahan pada pengembang dalam membaca dan menulis kode program.

```

fun main(args: Array<String>) {
    val oneMillion = 1_000_000
    val creditCardNumber = 1234_5678_9012_3456L
    val socialSecurityNumber = 999_99_9999L
    val hexBytes = 0xFF_EC_DE_5E
    val bytes = 0b11010010_01101001_10010100_10010010
    println("oneMillion = " + oneMillion)
    println("creditCardNumber = " + creditCardNumber)
    println("socialSecurityNumber = " + socialSecurityNumber)
    println("hexBytes = " + hexBytes)
    println("bytes = " + bytes)
}

```

Output:

```

"C:\Program Files\Java\jdk1.8.0_201\bin\java.exe" ...
oneMillion = 1000000
creditCardNumber = 1234567890123456
socialSecurityNumber = 999999999
hexBytes = 4293713502
bytes = 3530134674

Process finished with exit code 0

```

Representasi

Untuk literal number penunjukan nilai (*boxing*) pada sebuah variabel tidak akan berpengaruh pada identitas suatu variabel. Perhatikan kode program berikut:

```
fun main(args: Array<String>) {
    val a: Int = 10000
    println("a apakah identik dengan a atau a === a : " + (a === a))
    val boxedA: Int? = a
    val anotherBoxedA: Int? = a
    println("boxedA apakah identik dengan anotherBoxedA atau boxedA === anotherBoxedA : " + (boxedA === anotherBoxedA))
    println("boxedA apakah memiliki nilai yang sama dengan anotherBoxedA atau boxedA == anotherBoxedA : " + (boxedA == anotherBoxedA))
}
```

Output:

```
"C:\Program Files\Java\jdk1.8.0_201\bin\java.exe" ...
a apakah identik dengan a atau a === a : true
boxedA apakah identik dengan anotherBoxedA atau boxedA === anotherBoxedA : false
boxedA apakah memiliki nilai yang sama dengan anotherBoxedA atau boxedA == anotherBoxedA : true
Process finished with exit code 0
```

Konversi Eksplisit

Setiap Number di Kotlin mendukung konversi secara eksplisit seperti :

- toByte(): konversi ke Byte
- toShort():konversi ke Short
- toInt():konversi ke Int
- toLong():konversi ke Long
- toFloat():konversi ke Float
- toDouble():konversi ke Double
- toChar():konversi ke Char

Berikut contoh kode penerapan konversi secara eksplisit:

```
fun main(args: Array<String>) {
    val valInt: Int = 1
    val valSum: Long = 3L + valInt //konversi implisit
    println("Konversi variabel valInt secara implisit : " + valSum)
    val valLong: Long = valInt.toLong() // konversi eksplisit
    println("Konversi variabel valInt secara eksplisit : " + valLong)
}
```

Output:

```
"C:\Program Files\Java\jdk1.8.0_201\bin\java.exe" ...
Konversi variabel valInt secara implisit : 4
Konversi variabel valInt secara eksplisit : 1

Process finished with exit code 0
```

Character (Char)

Karakter di Kotlin ditandai dengan memberikan single quotes pada nilai sebuah variabel. Karakter di Kotlin tidak bisa diperlakukan secara langsung sebagai Number.

Contoh :

```
fun main(args: Array<String>) {
    var charA : Char = 'A'
    println(charA == 65)
}
```

Output :

```
1 fun main(args: Array<String>) {
2     var charA : Char = 'A'
3     println(charA == 65)
4 }
```

Operator '==' cannot be applied to 'Char' and 'Int'

Seperti di Java kita juga bisa menggunakan escape karakter seperti `\t`, `\b`, `\n`, `\r`, `\'`, `\"`, `\\`, dan `\$`. Jika diperlukan kita juga bisa melakukan encode apapun karakter lainnya dengan menggunakan Unicode sintak contohnya (`\uFF00`).

Character di Kotlin adalah sebagai objek yang memiliki fungsi anggota seperti `isLowerCase()`, `isDigit()`, `toUpperCase()`, `toLowerCase()`, `toString()` dan lainnya.

Contoh kode program penggunaan fungsi anggota dari Character:

```
fun main(args: Array<String>) {
    var charA : Char = 'A'
    //println(charA == 65)
    println(charA.isUpperCase())
    println(charA.isLowerCase())
    println(charA.isDigit())
    println(charA.toLowerCase())
    val strA: String = charA.toString()
    println("Kini charA sudah menjadi String = "+strA)
}
```

Output:

```
"C:\Program Files\Java\jdk1.8.0_201\bin\java.exe" ...
true
false
false
a
Kini charA sudah menjadi String = A

Process finished with exit code 0
```

Penerapan Konversi Eksplisit pada Char

Berikut program untuk melakukan konversi sebuah karakter kedalam nilai ASCII dan sebaliknya.

```
fun main(args: Array<String>) {  
    //ASCII value  
    var c: Char // character  
    var i: Int // ordinal (ASCII) value of the character  
    // conversion from text to ASCII value  
    c = 'a'  
    i = c.toInt()  
    println("The character $c was converted to its ASCII value of $i")  
  
    // conversion from an ASCII value to text  
    i = 98  
    c = i.toChar()  
    println("The ASCII value of $i was converted to its textual value of $c")  
}
```

Output:

```
"C:\Program Files\Java\jdk1.8.0_201\bin\java.exe" ...  
The character a was converted to its ASCII value of 97  
The ASCII value of 98 was converted to its textual value of b  
  
Process finished with exit code 0
```

Boolean (true or false)

Tipe data Boolean memiliki nilai true atau false. Nilai 0 dan 1 tidak berlaku untuk menunjuk nilai pada sebuah variabel bertipe Boolean.

```
fun main(args: Array<String>) {  
    var varBool : Boolean = true  
    if(varBool)  
        println("varBool bernilai " + varBool)  
}
```

Output:

```
"C:\Program Files\Java\jdk1.8.0_201\bin\java.exe" ...  
varBool bernilai true  
  
Process finished with exit code 0
```

Tambahkan kode berikut dalam program dan perhatikan hasilnya:

```
fun main(args: Array<String>) {  
    var varBool : Boolean = true  
    if(varBool)  
        println("varBool bernilai " + varBool)  
    var varInt : Int = 1
```

```

    if(varInt)
        println("varInt bernilai " + varInt)
}

```

Buatlah analisa anda atas pada baris kode if(varInt).

String (Array of Character)

String merupakan array dari karakter dan bersifat *immutable* (bersifat statis). Elemen suatu String dapat diakses dengan operasi indeks array contoh s[i] atau dengan menggunakan for-loop.

Contoh kode program mengakses elemen String:

```

fun main(args: Array<String>) {
    val myName: String = "Depandi Enda"
    for(chr in myName){
        print(chr)
    }
    print('\n')
}

```

Output:

```

"C:\Program Files\Java\jdk1.8.0_201\bin\java.exe" ...
Depandi Enda

Process finished with exit code 0

```

Kita bisa menggabungkan String dengan tipe data lain dengan menggunakan operator +, sehingga tipe data yang digabung otomatis akan melebur kedalam String.

```

fun main(args: Array<String>) {
    val s = "abc" + 1
    println(s + "def")
}

```

Output :

```

"C:\Program Files\Java\jdk1.8.0_201\bin\java.exe" ...
abc1def

Process finished with exit code 0

```

Untuk dapat menyimpan data String yang memuat keywords Kotlin bisa dengan menggunakan raw String yang dipisahkan dengan triple quotes (""").

Contoh kode program raw String:

```

fun main(args: Array<String>) {
    var text = """
        for (c in "foo")
            print(c)
    """
    println(text)
}

```



```
}
```

Output :

```
"C:\Program Files\Java\jdk1.8.0_201\bin\java.exe" ...  
  
    for (c in "foo")  
        print(c)  
  
Process finished with exit code 0
```

Kita juga dapat menghapus spasi yang tidak diperlukan dalam raw String seperti contoh kode program berikut:

```
fun main(args: Array<String>) {  
    var text = ""  
        |Tell me and I forget.  
        |Teach me and I remember.  
        |Involve me and I learn.  
        |(Benjamin Franklin)  
    """.trimMargin()  
    println(text)  
}
```

Output:

```
"C:\Program Files\Java\jdk1.8.0_201\bin\java.exe" ...  
Tell me and I forget.  
Teach me and I remember.  
Involve me and I learn.  
(Benjamin Franklin)  
  
Process finished with exit code 0
```

String Templates

Template ekspresi String diawali dengan tanda dolar (\$) diikuti oleh nama variabel yang akan digabungkan kedalam suatu String. Perhatikan kode program berikut:

```
fun main(args: Array<String>) {  
    val a = 7  
    val b = 8  
    val c = a + b  
    val s = "When we add $a and $b, we get $c"  
    println(s)  
    println("When we add $a and $b, we get ${a + b}")  
    val price = ""  
    ${'$'}9.99  
    ""  
    println(price)  
}
```

Output:

```
"C:\Program Files\Java\jdk1.8.0_201\bin\java.exe" ...  
When we add 7 and 8, we get 15  
When we add 7 and 8, we get 15  
  
$9.99  
  
Process finished with exit code 0
```

Diperlukan tanda kurung kurawal untuk melakukan pemrosesan pada saat menggabungkan variabel kedalam template String. Karakter dolar ('\$') tidak bisa dicetak menggunakan escape character oleh karena itu gunakan template String untuk mencetaknya.

Fungsi Anggota pada String

Seperti Java String juga merupakan Objek yang memiliki fungsi anggota. Berikut adalah contoh kode program untuk menggunakan fungsi anggota pada String.

```
fun main(args: Array<String>) {  
    val s = "Rhinopotamus"  
    println(s.startsWith("rhin"))  
    println(s.endsWith("tamus"))  
    println(s.contains("pot"))  
    println(s.contains("lol"))  
  
    //toUpperCase() and toLowerCase()  
    var config = "Fullscreen shaDows autosave"  
    config = config.toLowerCase()  
    println("Will the game run in fullscreen?")  
    println(config.contains("fullscreen"))  
    println("Will shadows be turned on?")  
    println(config.contains("shadows"))  
    println("Will sound be turned off?")  
    println(config.contains("nosound"))  
    println("Would the player like to use autosave?")  
    println(config.contains("autosave"))  
  
    //replace()  
    var strJava = "Java is the best!"  
    strJava = strJava.replace("Java", "Kotlin")  
    println(s)  
    println("$strJava is ${strJava.length} characters long.")  
  
    //substring()  
    println("I would not banish all of these Internets.".substring(2, 7))  
  
    //compareTo()  
    println("alpha".compareTo("bravo"))  
}
```

Output:

```
"C:\Program Files\Java\jdk1.8.0_201\bin\java.exe" ...
false
true
true
false
Will the game run in fullscreen?
true
Will shadows be turned on?
true
Will sound be turned off?
false
Would the player like to use autosave?
true
Rhinopotamus
Kotlin is the best! is 19 characters long.
would
-1

Process finished with exit code 0
```

Penerapan String

Berikut program untuk mencari huruf konsonan, huruf vocal dan karakter lainnya(selain huruf konsonan dan vocal) dalam sebuah String.

```
fun main(args: Array<String>) {
    // Character occurrence in a sentence analysis
    // the string that we want to analyze
    var s = "A programmer gets stuck in the shower because the instructions on the shampoo were: Lather,
Wash, and Repeat."
    println(s)
    s = s.toLowerCase()

    // counters initialization
    var vowelCount = 0
    var consonantCount = 0

    // definition of character groups
    val vowels = "aeiouy"
    val consonants = "bcdfghjklmnpqrstvwxyz"

    // main loop
    for (c in s) {
        if (vowels.contains(c)) {
            vowelCount++
        } else if (consonants.contains(c)) {
            consonantCount++
        }
    }
}
```

```

println("Vowels: $vowelCount")
println("Consonants: $consonantCount")
println("Other characters: ${s.length - (vowelCount + consonantCount)}")
}

```

Output:

```

"C:\Program Files\Java\jdk1.8.0_201\bin\java.exe" ...
A programmer gets stuck in the shower because the instructions on the shampoo were: Lather, Wash, and Repeat.
Vowels: 33
Consonants: 55
Other characters: 21

Process finished with exit code 0

```

Program Caesar Chiper

Program ini mengenkripsi sebuah input text String yang diberikan kedalam karakter yang dinaikan atau diturunkan.

```

fun main(args: Array<String>) {
    //The Caesar cipher
    // variable initialization
    val s = "blackholesarewheregoddividedbyzero"
    println("Original message: $s")
    var message = ""
    var shift = 1

    // Loop iterating over characters
    for (c in s) {
        var i = c.toInt()
        i += shift
        if (i > 'z'.toInt()) {
            i -= 26
        }
        val char = i.toChar()
        message += char
    }

    // printing
    println("Encrypted message: $message")
}

```

Output:

```

"C:\Program Files\Java\jdk1.8.0_201\bin\java.exe" ...
Original message: blackholesarewheregoddividedbyzero
Encrypted message: cmbdlipmftbsfxifsfhpeejwjefeczafsp

Process finished with exit code 0

```

Program Sandi Morse

Program ini membaca atau mengkonversi sebuah input text String berbentuk sandi Morse kedalam plain teks.

```
fun main(args: Array<String>) {
    // split() and joinToString()
    // Morse code decoder
    // the string which we want to decode
    val s = ".. -.-. - ... --- -.-. .. .- -.-."
    println("The original message: $s")

    // the string with the decoded message
    var message = ""

    // array definitions
    val alphabetChars = "abcdefghijklmnopqrstuvwxyz"
    val morseChars = arrayOf(".-", "-...", "-.-.", "-..", ".", "-.-.", "--", "...", "..", ".---", "-.-",
    ".-..", "--", "-.", "---", ".---", "--.-", ".-.", "...", "-", ".-.", "....", "-.-.", "-.-.", "-.-.", "-.-.",
    "-..")

    // splitting the string into Morse characters
    val characters = s.split(" ")

    // iterating over Morse characters
    for (morseChar in characters) {
        val index = morseChars.indexOf(morseChar)
        // character was found
        if (index != -1) {
            message += alphabetChars[index]
        }
    }
    println("The decoded message: $message")
}
```

Output:

```
"C:\Program Files\Java\jdk1.8.0_201\bin\java.exe" ...
The original message: .. -.-. - ... --- -.-. .. .- -.-.
The decoded message: ictsocial
Process finished with exit code 0
```

Perintah Masukan di Kotlin

Perintah masukan di Kotlin yaitu sebuah fungsi `readLine()!!`. Secara default perintah masukan di Kotlin akan mengembalikan nilai String ke dalam variabel penampung.

```
fun main(args: Array<String>) {
```

```
//Parrot program
println("Hi I'm Lora, the virtual parrot, and i love to repeat!");
println("Type something in: ");
var input: String
input = readLine()!!
var output: String
output = input + ", " + input + "!"
println(output)
}
```

Output:

```
"C:\Program Files\Java\jdk1.8.0_201\bin\java.exe" ...
Hi I'm Lora, the virtual parrot, and i love to repeat!
Type something in:
bla
bla, bla!

Process finished with exit code 0
```

Untuk merubah nilai String yang dikembalikan oleh fungsi `readLine()!!` kedalam nilai yang diinginkan bisa dilakukan dengan menggunakan konversi secara eksplisit seperti contoh program berikut:

```
fun main(args: Array<String>) {
    //Doubler program parsing
    println("Enter a number and I'll double it: ")
    val input = readLine()!!
    var a = input.toInt() // eventually Double
    a = a * 2
    println(a)

    println("Enter a number, and I'll square it:")
    val number = readLine()!!.toInt()
    val square = number * number
    println("Result: " + square)
}
```

Output:

```
"C:\Program Files\Java\jdk1.8.0_201\bin\java.exe" ...
Enter a number and I'll double it:
5
10
Enter a number, and I'll square it:
5
Result: 25

Process finished with exit code 0
```

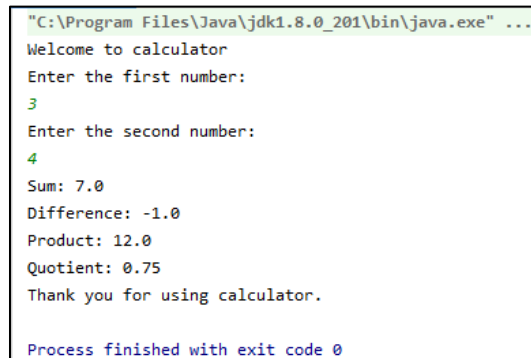
Program Kalkulator Sederhana

Program ini menerima 2 masukan bilangan dan akan melakukan operasi tambah, kurang, kali dan bagi dari kedua bilangan yang dimasukkan.

Kode Program:

```
fun main(args: Array<String>) {  
    //Simple calculator  
    println("Welcome to calculator")  
    println("Enter the first number:")  
    val a = readLine()!!.toDouble()  
    println("Enter the second number:")  
    val b = readLine()!!.toDouble()  
    val sum = a + b  
    val difference = a - b  
    val product = a * b  
    val quotient = a / b  
    println("Sum: ${sum}")  
    println("Difference: ${difference}")  
    println("Product: ${product}")  
    println("Quotient: ${quotient}")  
    println("Thank you for using calculator.")  
}
```

Output:



```
"C:\Program Files\Java\jdk1.8.0_201\bin\java.exe" ...  
Welcome to calculator  
Enter the first number:  
3  
Enter the second number:  
4  
Sum: 7.0  
Difference: -1.0  
Product: 12.0  
Quotient: 0.75  
Thank you for using calculator.  
Process finished with exit code 0
```

Tugas Praktikum

- Tuliskan pemahaman anda mengenai contoh koding program praktikum 1 dan screen shoot hasilnya.
- Buatlah laporan resmi praktikum 1.

Daftar Referensi

Ted Hagos, *Learn Android Studio 3 with Kotlin (Efficient Android App Development)*, Apress, Manila, 2018

Dmitry Jemerov and Svetlana Isakova, *Kotlin in Action*, Manning Publication Co., Shelter Island, 2017

<https://www.tutorialspoint.com/kotlin>

<https://www.ict.social/kotlin/basics>

<https://kotlinlang.org/docs/reference/>

** 😊 Selamat Mengerjakan 😊 ***