File - C:\Users\pepet\IdeaProjects\Variables\src\main\kotlin\Main.kt

```kotlin
fun main() {

// Declare Variables
    var myInt: Int = 4
    var myUInt: UInt = 4u
    var myLong: Long = 5L
    var myFloat: Float = 4.2f
    var myDouble: Double = 4.3
    var myHexInt: Int = 0x000A
    var myBinInt: Int = 0b0111
    var myChar: Char = 'D'
    var myByte: Byte = 2
    var myShort: Short = 5
    var myString: String = "inha"

// Type Casting
    myInt = myLong.toInt()

// Bit Operator (shifting, and, or, xor)
    var leftShift  = 1 shl 2 // shift left,  0100
    var rightShift = 0b0100.shr(2) // shift right, 0001
    var INT_MAX: UInt = (1 shl 31).toUInt() // shift left, 214748364
    var and = 1 and 0x00001111
    var or  = 1 or  0x00001111
    var xor = 1 xor 0x00001111

// String with Double Quotes, or Triple Double Quotes(No need escape letters)
    var myString1: String = "<Sale>\nPrice : \$100,000"
    var myString2: String = """<Sale>
Price : $100,000"""

// Array Declare
    var myArray = arrayOf(1,2,3)
    var mySquareArray1 = Array(10, {k -> k * k }) // {0,1,4,9,16,...,1024}
    var mySquareArray2 = Array(10, { it * it }) // {0,1,4,9,16,...,1024}

// Array Print
    println(myArray.contentToString()) // "[1, 2, 3]"
    println(myArray.joinToString()) // Only possible when element is primitive, like Array<Int>.

// Range operator '..', 'in', 'downTo', 'rangeTo', 'step', 'reversed'
// Remember : Range is defined by Start, End, Delta(step).
    val aToz = "A".."Z"
    val isCapitalLetter = "c" in aToz // false
    val myDescendingOrder1 = 5.downTo(1) // range 5,4,3,2,1
    val myDescendingOrder2 = 5 downTo 1 // range 5,4,3,2,1
    val myAscendingOrder1 = 5.downTo(9) // range 5,6,7,8,9
    val myAscendingOrder2 = 5 downTo 9 // range 5,6,7,8,9
    val my13579_1 = (1..10).step(2) // range 1,3,5,7,9
    val my13579_2 = 1..10 step 2 // range 1,3,5,7,9
    val my97531_1 = my13579_1.reversed() // range 9,7,5,3,1

// for loops
    // Remember : for ('elem' in 'range')

    // 1. .. operator
    for (i in 1..5){
        print(i)
    }; println() // 12345

    // 2. Array
    for (i in intArrayOf(0,1,0,5,3,1,8,6,4,6,1)){
        print(i)
    }; println() // 01053186461

    // 3. Descending Order
    for (i in 5 downTo 1){
        print(i)
    }; println() // 54321

    // 4. Descending Order, step
    for (i in 9 downTo 0 step 2){
        print(i)
    }; println() // 97531

    // 5. String
    val tmpString: String = "InHa"
    for (i in tmpString){
        print(i.toString()+" ")
    }; println() // I n h a

    // 6. When you need index, use 'indices'.
    val tmpArray = arrayOf(1,2,3)
    for (i in tmpArray.indices){
        println("Index $i : ${tmpArray[i]}")
    }

// class declare(No need to use 'new')
    class Vector2D(var x: Double, var y: Double){
```

File - C:\Users\pepet\IdeaProjects\Variables\src\main\kotlin\Main.kt

```kotlin
        constructor() : this(0.0, 0.0)
        fun biggerValue(): Double = if (x>y) x else y // return statement with one-line if-else!
    }
    var myVec = Vector2D(3.0, 4.0)
    println("${myVec.x} ${myVec.y} ${myVec.biggerValue()}")
    var myVec2 = Vector2D()
    println("${myVec2.x} ${myVec2.y} ${myVec2.biggerValue()}")

// How to print many variables(Use '$' in ""!)
    val tmpInt1 = 1;
    val tmpInt2 = 2;
    val tmpInt3 = 3;
    println("$tmpInt1, $tmpInt2, $tmpInt3") // 123

// Get User Input, and Store in List
    val myList: List<Int>? = readLine()?.split(" ")?.map{it.toInt()}
    // ? : Means that it's nullable.
    // readLine()? : Get user input as ASCII String.
    // split(" ") : Return List<T> that delimiter is " ".
    // map{code} : Apply 'code' in to every element, and change them.
    // it : Name of Variable in Lambda Function.
    // it.toInt() : Means to convert every element into Int.
    // Ex) input : "1 2" -> result : myList = [1,2]

// Get 2 numbers by user, print sum
    print(readln().sumOf{it-' '}-32)
    // readln() : Get user input as ASCII String.
    // sumOf : Function that return Sum, which have Lambda Function as it's argument
    // {it-' '} : Subtract ' ' for every char in String. So, it subtracts ' '(32).
    // -32 : '0' is 48. We have to subtract 16 for each number because we subtracted 32 already. So subtract 32 because there are 2 numbers.
    // Ex) input : "1 2" -> result : print 3.

// Get 2 numbers by user, print sum (2)
    print(readln().split(" ").sumOf{it.toInt()})
    // readln() : Get user input as ASCII String.
    // split(" ") : Return List<T> that delimiter is " ".
    // sumOf : Function that return Sum, which have Lambda Function as it's argument
    // Ex) input : "1 2" -> result : print 3.
}
```