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

```kotlin
// Simplest way to declare class
class Deposit {}

// example usage of 'constructor' keyword, and making 'instance'.
// Declare class Person1
class Person1 constructor(val firstName: String, val lastName: String, val age: Int?) {
    /* code */
}
// Make Instance of Person1
val person1 = Person1("Alex", "Smith", 29) // No need to say 'new' keyword!

// example usage of 'init', 'require', secondary constructor
// Declare class Person2
class Person2( val firstName: String, val lastName: String, val age: Int? ) { // 'constructor' keyword is optional for primary constructor.
    // Initialize
    init {
        // Check input values via standard library function 'require'
        require(firstName.trim().length > 0) { "Invalid firstName argument." }
        require(lastName.trim().length > 0) { "Invalid lastName argument." }
        if (age != null) {
            require(age >= 0 && age < 150) { "Invalid age argument." }
        }
    }
    // secondary constructor
    constructor(firstName: String, lastName: String) : this(firstName, lastName, null) {}
}
// Make Instance of Person2
val person3 = Person2("Inha", "Woo") // age will be null

// Nested Class
class Outer {
    private var privateInt = 3
    class staticNestedClass { // Equivalent to static nested class in Java
        init {
            // println("Outer's static nested class : ${privateInt}") // ERROR! Cannot access private member.
        }
    }
    inner class innerNestedClass { // Equivalent to non-static(inner) nested class in Java
        init {
            println("Outer's inner nested class : ${privateInt}") // OK
        }
    }
}

// this@label
class A {
    var myVal = 1
    inner class B {
        var myVal = 2
        init {
            println("Field <myVal> from B: " + this.myVal) // 2
            println("Field <myVal> from B: " + this@B.myVal) // 2
            println("Field <myVal> from A: " + this@A.myVal) // 1
        }
    }
}

// enum class
enum class Day { MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY, SATURDAY, SUNDAY }
enum class Planet(val mass: Double, val radius: Double) {
    MERCURY(3.303e+23, 2.4397e6),
    VENUS(4.869e+24, 6.0518e6),
    EARTH(5.976e+24, 6.37814e6),
    MARS(6.421e+23, 3.3972e6),
    JUPITER(1.9e+27, 7.1492e7),
    SATURN(5.688e+26, 6.0268e7),
    URANUS(8.686e+25, 2.5559e7),
    NEPTUNE(1.024e+26, 2.4746e7)
}

// Singleton class ('Object' in Kotlin)
object myButton {
    private var count = 0
    fun press() {
```

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

```
75          println("Calling myButton.press() : ${++count}")
76      }
77  }
78
79  fun main() {
80      val tmpOuter = Outer()
81      val tmpInner = tmpOuter.innerNestedClass() // Outer's inner nested class : 3
82
83      val tmpA = A()
84      val tmpB = tmpA.B() // A.B.init() is called in here
85      println("${tmpB.myVal}") // 2
86
87      val venus = Planet.valueOf("VENUS")
88      println("VENUS : mass(${venus.mass}), radius(${venus.radius})") // VENUS : mass(4.869E24), radius(
    6051800.0)
89
90      val mySingletonButton = myButton // Not myButton()
91      mySingletonButton.press() // 1
92      mySingletonButton.press() // 2
93      mySingletonButton.press() // 3
94      mySingletonButton.press() // 4
95  }
96
97
98
```