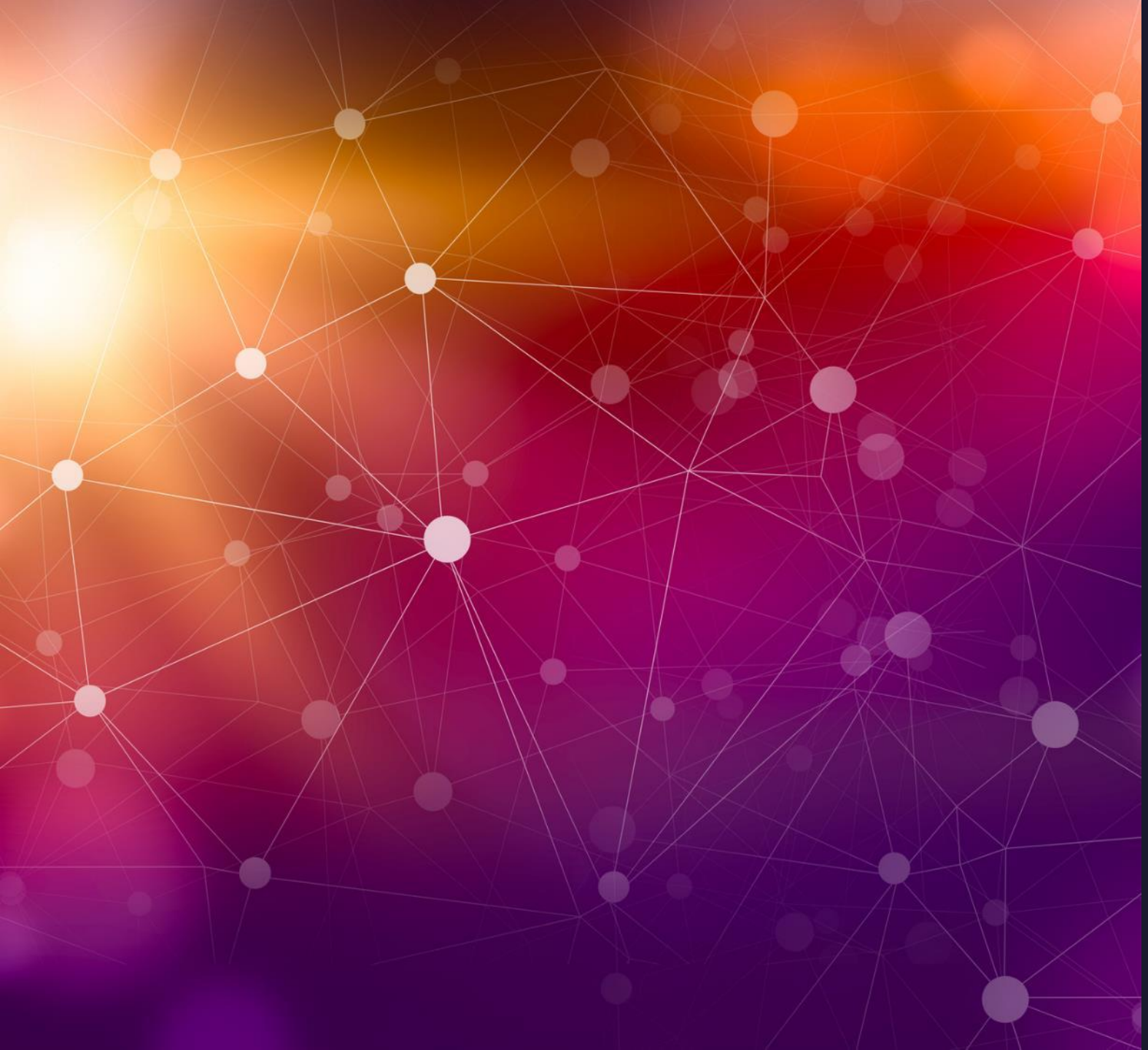

Algunas cosas básicas de Kotlin

Ilse Suárez

Los tipos de datos en kotlin son: Byte, Short, Int, Long, Float, Double, String, Char, Boolean

Variables

Variable mutable, es decir que su valor se puede reasignar varias veces y se determina en tiempo de ejecución.

El tipo de dato puede ser opcional, siempre y cuando su valor no sea nulo.

```
fun main() {
    // Declaración de var
    var tienda: String = "Una libra de arroz, por favor"
    tienda = "Me puede adicionar dos libras de arroz, por favor"
    println(tienda) // Me puede adicionar dos libras de arroz, por
    favor

    // Tipo de dato Boolean
    var declaracion = true
    println(declaracion) // true
}
```

Los tipos de datos en kotlin son: Byte, Short, Int, Long, Float, Double, String, Char, Boolean

Val

- Variable de solo lectura e inmutable. Su valor no puede ser reasignado en un futuro y se determina en tiempo de ejecución.
- El tipo de dato también puede ser opcional, siempre y cuando su valor no sea nulo.

```
fun main() {
    // Declaración de val
    val hola: String = "Bienvenidos a conceptos básicos de Kotlin"
    println(hola) // Bienvenidos a conceptos básicos de Kotlin

    // Error: No se puede declarar un valor nulo
    /* val nullDeclarado: Long = null */
}
```

Const val

- Es una variable inmutable de solo lectura, que su valor no va a hacer cambiado en un futuro. La diferencia con val, es que const val se determina en tiempo de compilación.
- Debe ser una propiedad de nivel superior es decir global o un miembro de una declaración object o companion object
- Una buena práctica en kotlin es definir el nombre de las variables const val en mayúsculas.
- Siempre se debe inicializar su valor.

```
// Variable de nivel global (nivel superior)
const val PI: Double = 3.1416
const val LENGUAJE_OFICIAL: String = "Kotlin"
const val PRIMERA_VOCAL = 'A' // Se puede inferir el tipo de dato

fun main() {
    // Declaración de const val
    println("El valor de PI es: $PI") // El valor de PI es: 3.1416
    println("El lenguaje oficial de android es: $LENGUAJE_OFICIAL")
    // El lenguaje oficial de android es: Kotlin
    println("La primera vocal es: $PRIMERA_VOCAL") // La primera
    vocal es: A
}
```

Modificadores de visibilidad

- Las clases, objetos, interfaces, constructores y funciones, así como las propiedades y sus setters, pueden tener modificadores de visibilidad. Los getters siempre tienen la misma visibilidad que sus propiedades. Las variables, funciones y clases locales no pueden tener modificadores de visibilidad.

- `private:` Solo es visible dentro del archivo en el cual fue declarado y también es visible dentro de la clase contenedora (incluido todos sus miembros).
- `public:` Su implementación esta por defecto, lo que significa que sus declaraciones serán visibles en todas partes.
- `protected:` No está disponible para declaraciones de nivel superior. Es visible en la clase contenedora y subclase de la misma.
- `internal:` Es visible en todas partes del mismo modulo, es decir un conjunto de archivos kotlin compilados juntos.

Getters y setters

- En kotlin los getters y setters se generan automáticamente según el tipo de variable que ha implementado (val o var).

```
const val GETTERS = "GETTER en tiempo de compilación"

class GettersYSetters {
     // Personalización Getter Y Setter
     var lenguaje: String = "Kotlin"
         get() {
              println("Retorna el valor por defecto: $field")
              return field // Palabra reservada field, tiene acceso
              al valor del campo asigando.
         }
         set(nuevoValor) {
              println("Retorna el nuevo valor asignado:
              $nuevoValor")
              field = nuevoValor // Se asigna el nuevo valor
         }

         var miTallaDeZapatos = 40 // Genera automáticamente getter
         y setter

         var privateSetters: String = "abc"
             private set // Setter privado

         // Personalización Getter
         val estadoAnimo: String
             get() = "Super Feliz"

         val lenguajeCienciaDatos = "Python" // Solo genera getter
}
```

- Se pueden personalizar los getters y setters.
- La variable val, por ser inmutable genera getters.
- La variable var, por ser mutable genera getters y setters.
- const val no puede generar getters personalizados.
- La palabra reservada field, tiene acceso al valor del campo asignado.

Null Safety

- En kotlin los objetos por defecto no aceptan valores nulos, para poder asignar un valor nulo tendremos que indicar que ese objeto realmente pueda ser nulo.

- Operador Safe Call (?): En los sistemas de tipos, para poder hacer una llamada segura en valores nulos colocar el siguiente operador llamado Safe Call (?)
- Operador Elvis (?:): Sirve para hacer una comprobación, si una condición no coincide la otra se ejecutará.
- Operador Double Bang (!!): Con este operador se evita la necesidad de chequear nulos, si está completamente seguro de que una variable nunca será nula.

- let: Para realizar una determinada operación solo para valores no nulos, puede utilizar el operador Safe Call (?) junto con let.
- Existen funciones que ayudan a comprobar si hay valores nulos como: filterNotNull(), isNullOrBlank(), isNullOrEmpty(), maxOrNull(), randomOrNull() y muchas más.

```
fun main() {
    var a: String = "kotlin"
    /* a = null */ // Error de compilación
    // Si la variable no es nula, se puede omitir el operador Safe
    Call (llamada segura)
    println(a?.length) // 6

    val b: String? = null
    println(b?.length) // null

    val c = b?.length ?: -1
    println(c) // -1

    //val d = b!!.length
    //println(d) // Error: NullPointerException

    val nullList: List<Int?> = listOf(1, 2, null, 4)
    println(nullList.filterNotNull()) // [1,2,4]
    println(nullList.elementAtOrNull(10)) // null

    val letList: List<String?> = listOf("java", null, "kotlin")
        for (list in letList) {
            list?.let { // Con el operador Safe Call, ignora los
            valores nulos.
                print("$it ") // java kotlin
            }
       }
}
```

Gracias

Ilse Suárez

Programación de dispositivos móviles

Facultad de Ciencias, UNAM

ilse_suarez@ciencias.unam.mx