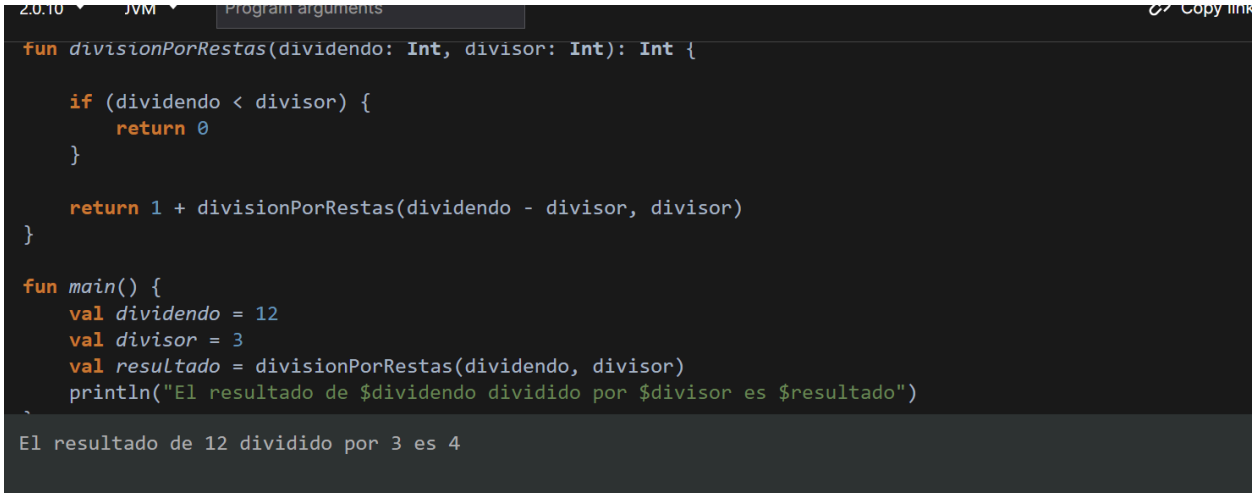
Integrantes: Sandyslein Correa Bueno, Anamaria Torres Peña

1.




```
fun divisionPorRestas(dividendo: Int, divisor: Int): Int {

    if (dividendo < divisor) {
        return 0
    }


    return 1 + divisionPorRestas(dividendo - divisor, divisor)
}

fun main() {
    val dividendo = 12
    val divisor = 3
    val resultado = divisionPorRestas(dividendo, divisor)
```

```
    println("El resultado de $dividendo dividido por $divisor es $resultado")
}
```

2.

```
data class Empleado(
    val nombreCompleto: String,
    val documentoIdentidad: String,
    val sexo: Char,
    val correoElectronico: String,
    val salario: Double,
    val dependencia: Dependencia,
    val anoIngreso: Int,
    val cargo: Cargo,
    val subordinados: MutableList<Empleado> = mutableListOf()
)

data class Cliente(
    val nombreCompleto: String,
    val documentoIdentidad: String,
    val sexo: Char,
    val correoElectronico: String,
    val direccion: String,
    val telefono: String
)
```

```
data class Cargo(
    val nombre: String,
    val nivelJerarquico: Int
)

data class Empresa(
    val razonSocial: String,
    val nit: String,
    val direccion: String,
    val empleados: MutableList<Empleado>,
    val clientes: MutableList<Cliente> = mutableListOf()
)

enum class Dependencia {
    VENTAS,
    RECURSOS_HUMANOS,
    GERENCIA,
    OPERATIVO
}

class SistemaGestion(private val empresa: Empresa) {

    // CRUD para Empleados
```

```
    fun agregarEmpleado(empleado: Empleado) {
        empresa.empleados.add(empleado)
    }

    fun eliminarEmpleado(documentoIdentidad: String) {
        empresa.empleados.removeIf { it.documentoIdentidad == documentoIdentidad }
    }

    fun actualizarEmpleado(documentoIdentidad: String, nuevoEmpleado: Empleado) {
        val index = empresa.empleados.indexOfFirst { it.documentoIdentidad ==
documentoIdentidad }
        if (index != -1) {
            empresa.empleados[index] = nuevoEmpleado
        }
    }

    fun obtenerEmpleado(documentoIdentidad: String): Empleado? {
        return empresa.empleados.find { it.documentoIdentidad == documentoIdentidad }
    }

    fun listarEmpleados(): MutableList<Empleado> {
        return empresa.empleados
    }

    // CRUD para Clientes
```

```
    fun agregarCliente(cliente: Cliente) {
        empresa.clientes.add(cliente)
    }

    fun eliminarCliente(documentoIdentidad: String) {
        empresa.clientes.removeIf { it.documentoIdentidad == documentoIdentidad }
    }

    fun actualizarCliente(documentoIdentidad: String, nuevoCliente: Cliente) {
        val index = empresa.clientes.indexOfFirst { it.documentoIdentidad ==
documentoIdentidad }
        if (index != -1) {
            empresa.clientes[index] = nuevoCliente
        }
    }

    fun obtenerCliente(documentoIdentidad: String): Cliente? {
        return empresa.clientes.find { it.documentoIdentidad == documentoIdentidad }
    }



    fun obtenerNominaTotal(): Double {
        return empresa.empleados.sumOf { it.salario }
    }
```

```
fun obtenerNominaPorDependencia(): Map<Dependencia, Double> {
    return empresa.empleados.groupBy { it.dependencia }
        .mapValues { entry -> entry.value.sumOf { it.salario } }
}


fun obtenerPorcentajeClientesPorSexo(): Map<Char, Double> {
    val totalClientes = empresa.clientes.size
    return empresa.clientes.groupBy { it.sexo }
        .mapValues { entry -> (entry.value.size.toDouble() / totalClientes) * 100 }
}


fun obtenerCantidadEmpleadosPorCargo(): Map<String, Int> {
    return empresa.empleados.groupBy { it.cargo.nombre }
        .mapValues { entry -> entry.value.size }
}


fun obtenerEmpleadoConMasTiempo(): Empleado? {
    return empresa.empleados.minByOrNull { it.anoIngreso }
}
```

```
    fun obtenerDependenciaEmpleadoConMasTiempo(): Dependencia? {
        val empleado = obtenerEmpleadoConMasTiempo()
        return empleado?.dependencia
    }

}
fun main() {
    // Creación de objetos de prueba
    val cargoGerente = Cargo("Gerente", 1)
    val empleado1 = Empleado(
        nombreCompleto = "Juan Pérez",
        documentoIdentidad = "12345678",
        sexo = 'M',
        correoElectronico = "juan.perez@empresa.com",
        salario = 5000.0,
        dependencia = Dependencia.GERENCIA,
        anoIngreso = 2010,
        cargo = cargoGerente
    )

    val empresa = Empresa(
        razonSocial = "Empresa S.A.",
        nit = "900123456",
```

```
    direccion = "Calle 123 #45-67",
    empleados = mutableListOf(empleado1)
)

val sistemaGestion = SistemaGestion(empresa)

// Agregar más empleados
sistemaGestion.agregarEmpleado(
    Empleado(
        nombreCompleto = "María López",
        documentoIdentidad = "87654321",
        sexo = 'F',
        correoElectronico = "maria.lopez@empresa.com",
        salario = 4500.0,
        dependencia = Dependencia.VENTAS,
        anoIngreso = 2015,
        cargo = Cargo("Vendedor", 2)
    )
)

sistemaGestion.agregarEmpleado(
    Empleado(
        nombreCompleto = "Carlos González",
        documentoIdentidad = "11223344",
```

```
            sexo = 'M',
            correoElectronico = "carlos.gonzalez@empresa.com",
            salario = 3000.0,
            dependencia = Dependencia.RECURSOS_HUMANOS,
            anoIngreso = 2020,
            cargo = Cargo("Asistente", 3)
        )
    )

    println("Nómina total de la empresa: ${sistemaGestion.obtenerNominaTotal()}")
    println("Nómina por dependencia:
${sistemaGestion.obtenerNominaPorDependencia()}")

    val empleadoMasAntiguo = sistemaGestion.obtenerEmpleadoConMasTiempo()
    println("Empleado con más tiempo en la empresa:
${empleadoMasAntiguo?.nombreCompleto}")
    println("Dependencia del empleado con más tiempo:
${sistemaGestion.obtenerDependenciaEmpleadoConMasTiempo()}")

    sistemaGestion.agregarCliente(
        Cliente(
            nombreCompleto = "Laura Martínez",
            documentoIdentidad = "22334455",
            sexo = 'F',
            correoElectronico = "laura.martinez@cliente.com",
```

```
            direccion = "Avenida 1 #10-20",
            telefono = "3001234567"
        )
    )

    sistemaGestion.agregarCliente(
        Cliente(
            nombreCompleto = "Pedro Sánchez",
            documentoIdentidad = "33445566",
            sexo = 'M',
            correoElectronico = "pedro.sanchez@cliente.com",
            direccion = "Carrera 2 #20-30",
            telefono = "3101234567"
        )
    )

    println(sistemaGestion.listarEmpleados())

    sistemaGestion.eliminarEmpleado("11223344")

    println(sistemaGestion.listarEmpleados())

    println("Porcentaje de clientes por sexo:
${sistemaGestion.obtenerPorcentajeClientesPorSexo()}")
    println("Cantidad de empleados por cargo:
```

${sistemaGestion.obtenerCantidadEmpleadosPorCargo()}")

}

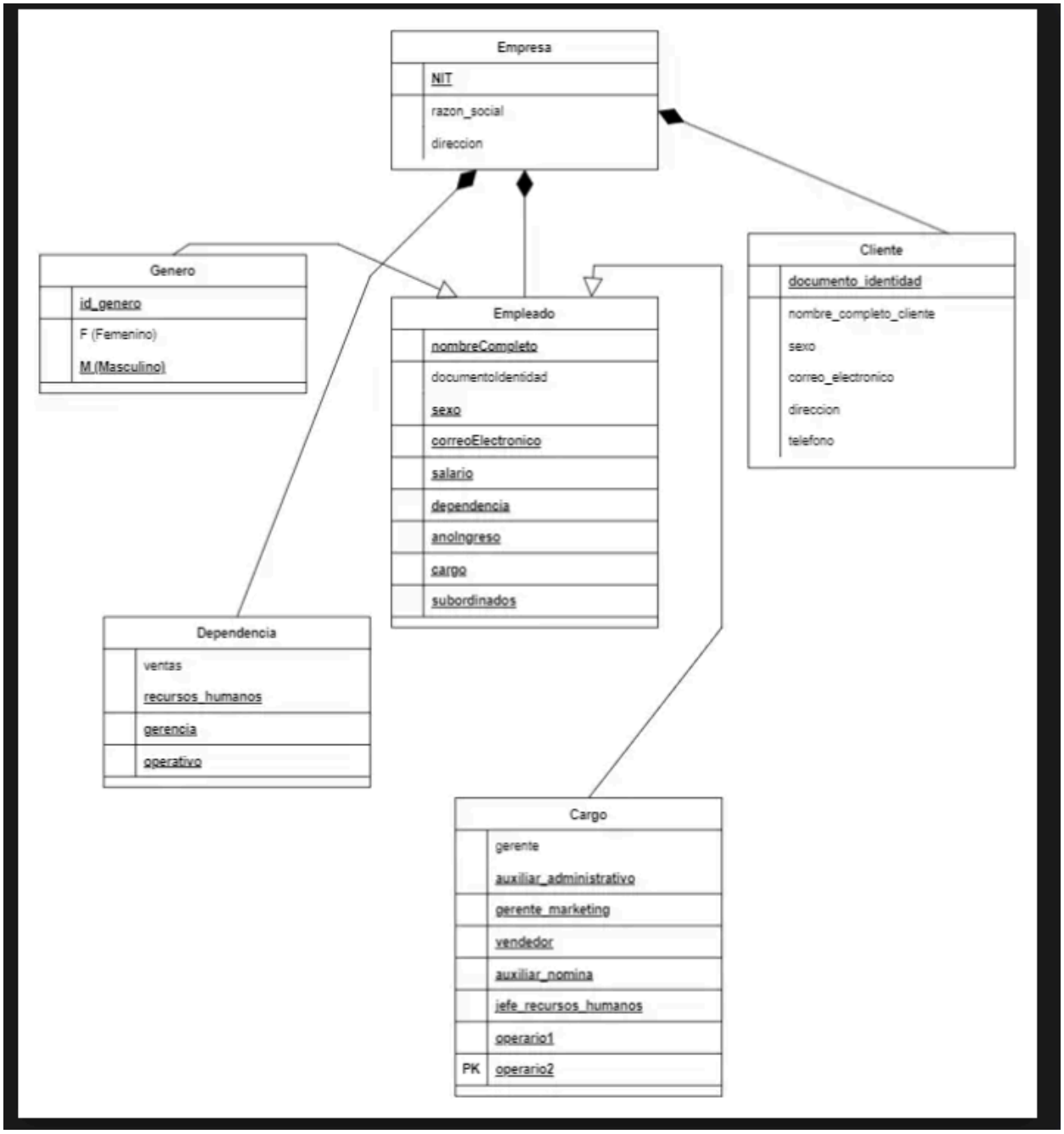
```
    )

    sistemaGestion.agregarCliente(
        Cliente(
            nombreCompleto = "Pedro Sánchez",
            documentoIdentidad = "33445566",
            sexo = 'M',
            correoElectronico = "pedro.sanchez@cliente.com",
            direccion = "Carrera 2 #20-30",
            telefono = "3101234567"
```

```
Nómina total de la empresa: 12500.0
Nómina por dependencia: {GERENCIA=5000.0, VENTAS=4500.0, RECURSOS_HUMANOS=3000.0}
Empleado con más tiempo en la empresa: Juan Pérez
Dependencia del empleado con más tiempo: GERENCIA
Porcentaje de clientes por sexo: {F=50.0, M=50.0}
Cantidad de empleados por cargo: {Gerente=1, Vendedor=1, Asistente=1}
```

Diagrama

https://drive.google.com/file/d/1FVsihRkUZIv2Mo1rolRzrodU92l7PsiP/view?usp=sharing


Empresa
NIT
razon_social
direccion
Genero
id_genero
F (Femenino)
M (Masculino)
Empleado
nombreCompleto
documentoIdentidad
sexo
correoElectronico
salario
dependencia
anoIngreso
cargo
subordinados
Cliente
documento_identidad
nombre_completo_cliente
sexo
correo_electronico
direccion
telefono
Dependencia
ventas
recursos_humanos
gerencia
operativo
Cargo
gerente
auxiliar_administrativo
gerente_marketing
vendedor
auxiliar_nomina
jefe_recursos_humanos
operario1
PK
operario2