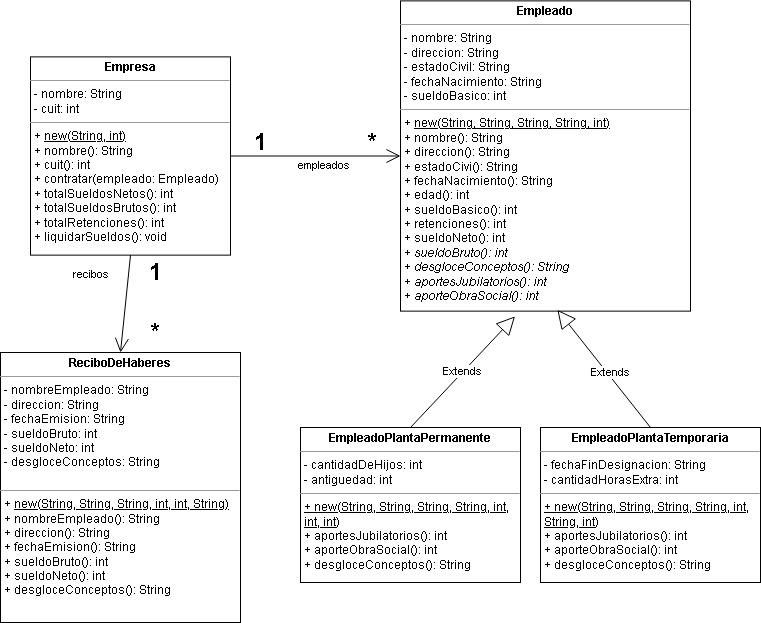
Ejercicio 1)

a)



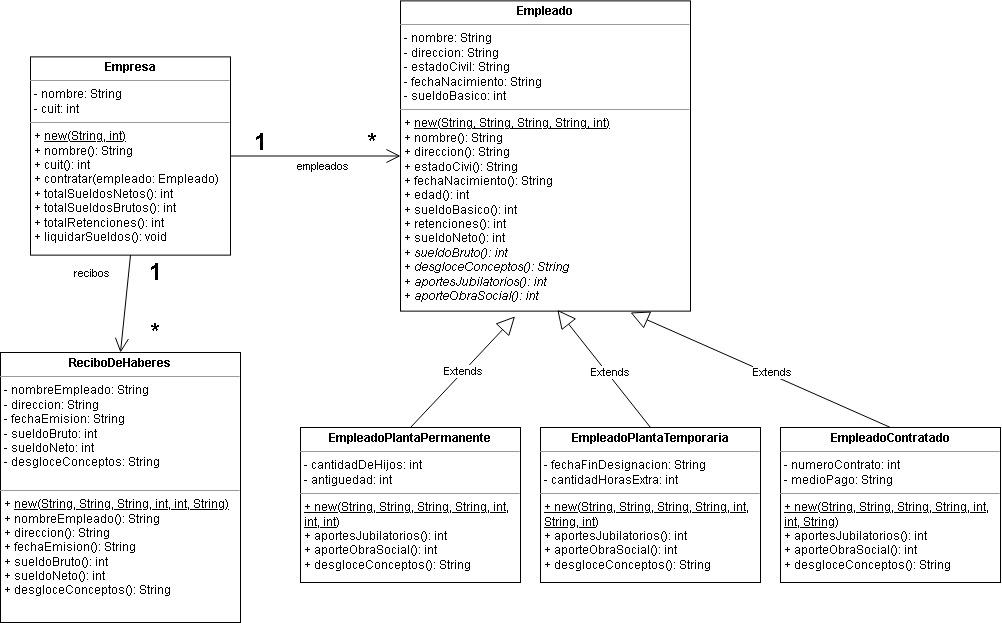
| import java.time.LocalDate  import java.time.format.DateTimeFormatter  class Empresa(  private val nombre: String,  private val cuit: Int,  ) {  private val empleados = *mutableListOf*<Empleado>()  private val recibos = *mutableListOf*<ReciboDeHaberes>()  fun nombre(): String = nombre  fun cuit(): Int = cuit  fun contratar(empleado: Empleado) {  empleados.add(empleado)  }  fun totalSueldosNetos(): Int {  return empleados.*sumOf* **{** empleado **->** empleado.sueldoNeto() **}**  }  fun totalSueldosBrutos(): Int {  return empleados.*sumOf* **{** empleado **->** empleado.sueldoBruto() **}**  }  fun totalRetenciones(): Int {  return empleados.*sumOf* **{** empleado **->** empleado.retenciones() **}**  }  fun liquidarSueldos() {  empleados.*forEach* **{** empleado **->**  recibos.add(reciboDe(empleado))  **}**  }  private fun reciboDe(empleado: Empleado): ReciboDeHaberes {  val date = LocalDate.now()  date.format(DateTimeFormatter.ofPattern("dd-MM-yyyy"))  val recibo = ReciboDeHaberes(  empleado.nombre(),  empleado.direccion(),  date.toString(),  empleado.sueldoBruto(),  empleado.sueldoNeto(),  empleado.desgloceConceptos()  )  return recibo  }  }  abstract class Empleado(  private val nombre: String,  private val direccion: String,  private val estadoCivil: String,  private val fechaNacimiento: String,  private val sueldoBasico: Int  ) {  fun nombre(): String = nombre  fun direccion(): String = direccion  fun edad(): Int {  val now = LocalDate.now()  val formatter = DateTimeFormatter.ofPattern("dd-MM-yyyy")  val anioActual = now.*year*  val anioNacimiento = LocalDate.parse(fechaNacimiento, formatter).*year*  return anioActual - anioNacimiento  }  fun estadoCivil(): String = estadoCivil  fun fechaNacimiento(): String = fechaNacimiento  fun sueldoBasico() = sueldoBasico  fun sueldoNeto(): Int = sueldoBruto() - retenciones()  fun retenciones(): Int = aportesObraSocial() + aportesJubilatorios()  fun desgloceConceptos(): String =  """EMPLEADO = $nombre  |APORTES OBRA SOCIAL = -${aportesObraSocial()}  |APORTES JUBILATORIOS = -${aportesJubilatorios()}  |${desgloceConceptosSueldoBruto()}""".*trimMargin*("|")  internal abstract fun desgloceConceptosSueldoBruto(): String  internal abstract fun sueldoBruto(): Int  internal abstract fun aportesObraSocial(): Int  internal abstract fun aportesJubilatorios(): Int  }  class EmpleadoPlantaPermanente(  nombre: String,  direccion: String,  estadoCivil: String,  fechaNacimiento: String,  sueldoBasico: Int,  private val cantidadHijos: Int,  private val antiguedad: Int  ) : Empleado(nombre, direccion, estadoCivil, fechaNacimiento, sueldoBasico) {  override fun sueldoBruto() = sueldoBasico() + sueldoFamiliar() + plusAntiguedad()  override fun aportesObraSocial(): Int = (sueldoBruto() / 10) + plusHijosObraSocial()  override fun aportesJubilatorios(): Int = (sueldoBruto() \* 15) / 100  override fun desgloceConceptosSueldoBruto(): String =  """SUELDO BÁSICO = ${sueldoBasico()}  |SUELDO FAMILIAR = ${sueldoFamiliar()}  |ANTIGUEDAD = ${plusAntiguedad()}""".*trimMargin*("|")  private fun plusAntiguedad() = antiguedad \* 50  private fun plusHijosObraSocial() = cantidadHijos \* 20  private fun sueldoFamiliar(): Int = asignacionHijos() + asignacionConyuge()  private fun asignacionHijos(): Int = 150 \* cantidadHijos  private fun asignacionConyuge(): Int = if (estadoCivil() == "casado") 100 else 0  }  class EmpleadoPlantaTemporaria(  nombre: String,  direccion: String,  estadoCivil: String,  fechaNacimiento: String,  sueldoBasico: Int,  private val fechaFinDesignacion: String,  private val cantidadHorasExtra: Int  ) : Empleado(nombre, direccion, estadoCivil, fechaNacimiento, sueldoBasico) {  override fun sueldoBruto(): Int = sueldoBasico() + horasExtrasPago()  override fun aportesObraSocial(): Int = sueldoBruto() / 10 + extraSiTieneMasDe50()  override fun aportesJubilatorios(): Int = sueldoBruto() / 10 + aporteHorasExtra()  override fun desgloceConceptosSueldoBruto(): String =  """SUELDO BÁSICO = ${sueldoBasico()}  |HORAS EXTRA = ${horasExtrasPago()}""".*trimMargin*("|")  fun cantidadHorasExtra() = cantidadHorasExtra  private fun aporteHorasExtra(): Int = cantidadHorasExtra \* 5  private fun horasExtrasPago(): Int = cantidadHorasExtra \* 40  private fun extraSiTieneMasDe50(): Int = if (edad() > 50) 25 else 0  }  class ReciboDeHaberes(  private val nombreEmpleado: String,  private val direccion: String,  private val fechaEmision: String,  private val sueldoBruto: Int,  private val sueldoNeto: Int,  private val desgloceConceptos: String  ) {  fun nombreEmpleado(): String = nombreEmpleado  fun direccion(): String = direccion  fun fechaEmision(): String = fechaEmision  fun sueldoBruto(): Int = sueldoBruto  fun sueldoNeto(): Int = sueldoNeto  fun desgloceConceptos(): String = desgloceConceptos  } |
| --- |

c)

| fun main() {  val empresa = Empresa("UNQEmpresa", 123456789)  val empleadoPepe = EmpleadoPlantaPermanente(  "Pepe",  "Av 1234 Quilmes",  "soltero",  "13-08-2000",  45756,  3,  5  )  val empleadoJuan = EmpleadoPlantaTemporaria(  "Juan",  "Av 4321 Quilmes",  "casado",  "13-04-2000",  2356,  "01-09-2023",  4  )  empresa.contratar(empleadoPepe)  empresa.contratar(empleadoJuan)  // calculo sueldos netos  empresa.totalSueldosNetos()  // liquidacion de sueldos  empresa.liquidarSueldos()  } |
| --- |

EJERCICIO 2:

a)



b)

class EmpleadoContratado(

nombre: String,

direccion: String,

estadoCivil: String,

fechaNacimiento: String,

sueldoBasico: Int,

private val numeroContrato: Int,

private val medioPago: String

) : Empleado(nombre, direccion, estadoCivil, fechaNacimiento, sueldoBasico) {

override fun sueldoBruto() = sueldoBasico()

override fun aportesObraSocial(): Int = 0

override fun aportesJubilatorios(): Int = 0

override fun retenciones() = gastosAdministrativosContractuales()

override fun desgloceConceptosSueldoBruto(): String =

"""SUELDO BÁSICO = ${sueldoBasico()}

GASTOS ADMINISTRATIVOS CONTRACTUALES = ${gastosAdministrativosContractuales()}""".*trimMargin*("|")

private fun gastosAdministrativosContractuales(): Int = 50

}

c)

No, ya que la clase empresa maneja la clase “Empleado”, y esta es una implementación que mantiene el polimorfismo, por lo que no cambia nada para ella.

d)

fun main() {

val empresa = Empresa("UNQEmpresa", 123456789)

val empleadoPepe = EmpleadoPlantaPermanente(

"Pepe",

"Av 1234 Quilmes",

"soltero",

"13-08-2000",

45756,

3,

5

)

val empleadoJuan = EmpleadoPlantaTemporaria(

"Juan",

"Av 4321 Quilmes",

"casado",

"13-04-2000",

2356,

"01-09-2023",

4

)

val empleadoMiguel = EmpleadoContratado(

"Miguel",

"Av 1345 Quilmes",

"soltero",

"13-04-2000",

5000,

1245634,

"tarjeta"

)

empresa.contratar(empleadoPepe)

empresa.contratar(empleadoJuan)

empresa.contratar(empleadoMiguel)

// calculo sueldos netos

empresa.totalSueldosNetos()

// liquidacion de sueldos

empresa.liquidarSueldos()

}