PRACTICA DE LENGUAJE GROOVY

Objetivos:

El objetivo del presente practico es que el alumno pueda resolver distintas situaciones problemáticas desarrollando algoritmos y estrategias de programación mediante el lenguaje dinámico Groovy.

Capacidades a desarrollar según CONFEDI[1]:

- ✓ Ser capaz de acceder a las fuentes de información relativas a las técnicas y herramientas y de comprender las especificaciones de las mismas.
- ✓ Ser capaz de conocer los alcances y limitaciones de las técnicas y herramientas a utilizar y de reconocer los campos de aplicación de cada una de ellas y de aprovechar toda la potencialidad que ofrecen.

Bibliografía: Venkat Subramaniam (2013). *Programming Groovy 2: Dynamic Productivity for the Java Developer*. The Pragmatic Bookshelf

Sitios de Referencias:

- Lenguaje de Programación Groovy – URL: http://groovy-lang.org/

Carácter de elaboración:

El presente trabajo laboratorio deberá ser desarrollado en forma individual o en grupo de hasta tres integrantes.

Forma y fecha de entrega:

Cada grupo deberá crear un repositorio en GitHub que contendrá el trabajo resuelto, la dirección del repositorio la deben enviar a la dirección de correo **oequinteros@tecno.unca.edu.ar**. La fecha de entrega es hasta el 14/09/2018.

[1] CONFEDI. (2014). Competencias en Ingeniería. Mar del Plata. 1a ed. Universidad Fasta. ebook. Mar del Plata.

Actividades a desarrollar

Ejercicio Nº 1

Dada la jerarquía de clases que se muestra en el diagrama de la Figura 1, cuya implementación se muestra en la tabla como sigue, indique qué retornan las siguientes expresiones:

def gerente = new Gerente()
a) gerente.aportes()
b) gerente.calcularSueldo()

Empleado	EmpleadoJerarquico	Gerente

Float sueldoBasico(){
return this.montoBasico()
-
this.aportes()
}
	Float calcularSueldo(){
	return super.sueldoBasico() +
	this.bonoPorCategoria()
	}
		Float aportes(){
		return this.montoBasico() *
		0.05
		}

Float montoBasico(){
return 7000
}
	Float montoBasico(){
	return 8000
	}
		Float montoBasico(){
		return 10000
		}

Float aportes(){
return 100
}
	Float bonoPorCategoria(){
	return 2000
	}	

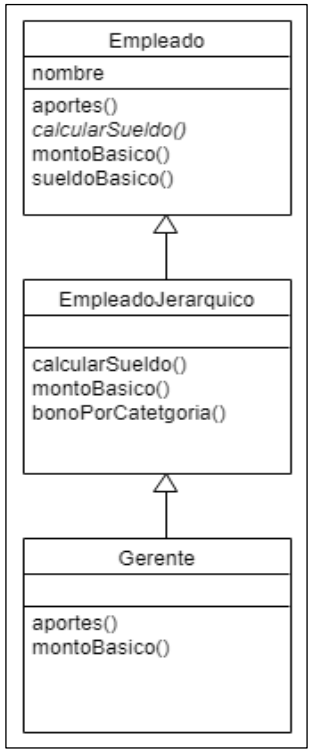

Figura 1. Diagrama de clases e implementación en Groovy

Ejercicio Nº 2

Dado el diagrama de clases UML de la Figura 2, implemente en Groovy las clases Estudiante y Curso. A continuación, se detalla el protocolo de la clase Curso:

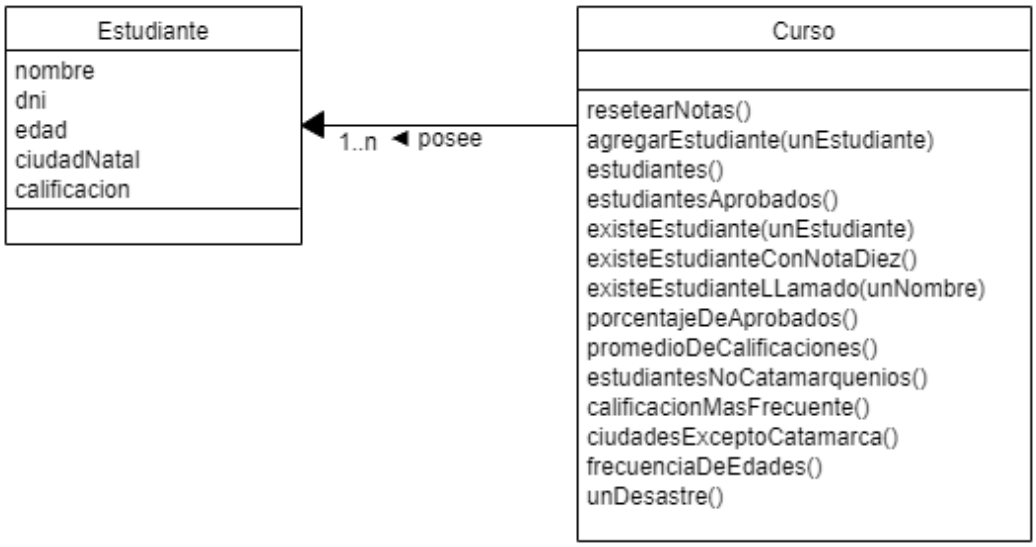

Figura 2. Diagrama de clases e implementación en Groovy

Protocolo Clase Curso

- #resetearNotas(): Pone en cero las calificaciones de todos los estudiantes.
- #agregarEstudiante(unEstudiante): Agrega unEstudiante al curso
- #cantidadDeEstudiantesInscriptos(): Retorna la cantidad de alumnos que se inscribieron al curso
- #estudiantes(): Retorna la colección de estudiantes del curso
- #estudiantesAprobados(): Retorna una colección con todos los estudiantes que aprobaron el curso (calificación superior a 4)
- #existeEstudiante(unEstudiante): Indica si "unEstudiante" se encuentra inscripto en el curso
- #existeEstudianteConNotaDiez(): Determina si algún alumno obtuvo la calificación 10
- #existeEstudianteLlamado(unNombre)(): Indica si el estudiante llamado "unNombre" se encuentra inscripto en el curso
- #porcentajeDeAprobados(): Retorna en porcentaje de estudiantes aprobados
- #promedioDeCalificaciones(): Calcula el promedio de las calificaciones obtenidas por los alumnos
- #estudiantesNoCatamarquenios(): Retorna una colección con todos los estudiantes que no nacieron en Catamarca
- #calificacionMasFrecuente(): Retorna la calificación más frecuente del curso
- #ciudadesExceptoCatamarca(): Retorna una colección, sin repeticiones, conteniendo los nombres de todas las ciudades (excepto Catamarca) donde nacieron los alumnos inscriptos al curso
- #unDesastre(): Retorna verdadero si todos los estudiantes desaprobaron el curso

• #frecuenciaDeEdades(): Retorna un mapa que asocia las edades de los estudiantes con la frecuencia en la que aparecen. Esta información luego será utilizada para generar un histograma (ver Figura 3) cuyo eje x serán las edades y cuyo eje y serán la frecuencia de las ocurrencias

La Figura 3 muestra un ejemplo del histograma que se desea realizar con la información provista por el método #frecuenciaDeEdades.

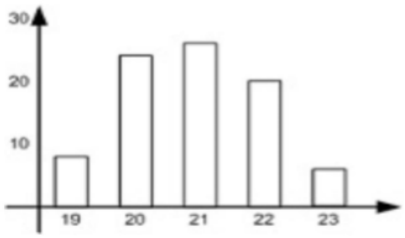

Figura 3. Histograma de edades