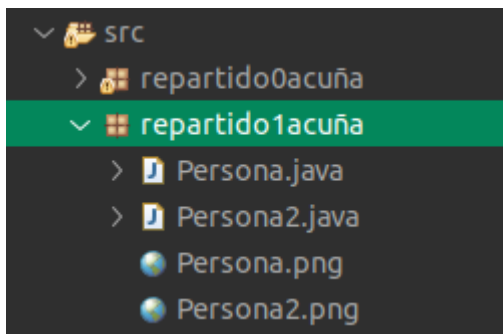


## Repartido 1

*El objetivo es comenzar a programar bajo el paradigma orientado a objetos.*

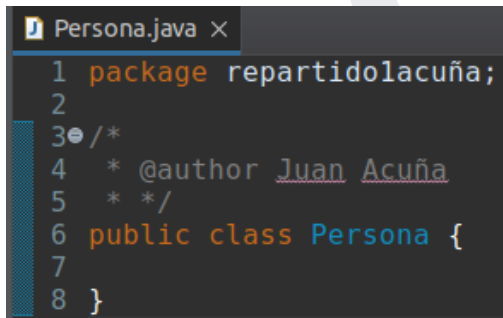
### Forma de entrega

Cada ejercicio deberá quedar guardado como un archivo, nombrándolo de la forma indicada en “producto esperado”. Y todos estos archivos deben estar en el paquete “repartido1apellido”.



En el código se pueden hacer las aclaraciones que se desee por medio de comentarios `//` o por medio de *Javadoc* `/**`

Se pide especialmente usar la tag `@author` para identificarlos mejor.



```
1 package repartido1acuña;
2
3 /**
4  * @author Juan Acuña
5  * */
6 public class Persona {
7
8 }
```

Al finalizar, se entregará el package, comprimido, exportándolo desde Eclipse.

Todos los archivos que sean del tipo imagen, debe ser del formato .png

Ejercicio 01

Debes diseñar una clase que pueda almacenar la información necesaria para representar a una persona. Debe contener los siguientes campos: fecha de nacimiento, cédula de identidad, nombre, apellido. Serás tú quien defina qué tipo de dato llevará cada campo y deberás justificarlo.

Esta clase no tendrá métodos.

Sobre la clase Persona deberás:

A. crear un diagrama de clase en UML que la represente. Producto esperado:

Persona.png

B. codificarla en Java. Producto esperado: Persona.java

Ejercicio 02

Debes crear una clase llamada Persona2 con los mismos campos que Persona1 pero agregando los métodos necesarios para cargar los valores en esos campos y los métodos necesarios para imprimir esos valores en pantalla.

Sobre la clase Persona deberás:

C. crear un diagrama de clase en UML que la represente. Producto esperado:

Persona2.png

D. codificarla en Java. Producto esperado: Persona2.java

Ejercicio 03

Debes codificar la clase representada en este diagrama de clase UML.

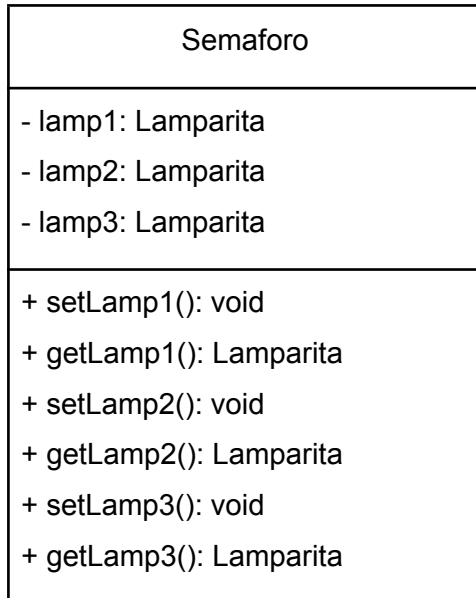
Lamparita
- encendida: boolean - color: String
+ setEncendida(): void + getEncendida(): boolean + setColor(): void + getColor(): String + toString(): String

El método toString() debe crear un String con un mensaje del tipo “Esta lamparita está encendida: *true/false*. Su color es: *color*”. Pero no lo imprime.

Producto esperado: Lamparita.java

Ejercicio 04

Debes codificar la clase representada en este diagrama de clase UML.



Producto esperado: Semaforo.java

Ejercicio 05

- I. Debes crear un array de tamaño 16 cuyo contenido sea del tipo:
  - A. boolean
  - B. int
  - C. Lamparita
- II. Tu programa (tendrá main) debe imprimir el contenido de cada array. No se puede modificar Lamparita, debes usar la misma definida en el Ejercicio 03.

Producto esperado: Ejercicio05.java

Ejercicio 06

Debes codificar la clase representada en este diagrama de clase UML.

CintaLed
- cinta: Lamparita[16]
+ setCinta(): void + getCinta(): Lamparita[ ]

Producto esperado: CintaLed.java

Ejercicio 07

La Java API tiene muchas clases predefinidas, una de ellas es Object. Esta tiene sus propios métodos, los cuales son 11 si nos fijamos en la [API Specification](#). De momento, solo nos interesan: equals(), hashCode() y getClass().

Guiándote con el siguiente diagrama UML, debes crear una clase que use los tres métodos nombrados anteriormente, pero dentro del main.

Ejercicio07
- a: Object - b: Object
+ main(String [ ]): void

De manera más precisa, el programa debe:

- preguntar si a y b son iguales por medio de equals()
- preguntar el hashcode de a y b.
- preguntar de qué clase son a y b.

Nota: se sobreentiende que todo quedará impreso en pantalla de manera MUY prolija.

Producto esperado: Ejercicio07.java

Ejercicio 08

En la unidad curricular Sistemas Operativos 2, pudiste observar que el sistema operativo Linux Ubuntu guarda sus logs en la carpeta /var/log.

Un log cuenta con un elemento fundamental que es la marca temporal, lo cual permite al administrador del sistema saber en qué momento ocurrió cada cosa (error o no).

Esas marcas temporales no aparecen mágicamente, fueron programadas. En Java lo podemos realizar con otra clase predefinida en la Java API: `LocalDate`.

Tomaremos el ejercicio 02 del repartido 0. Haremos una pequeña modificación al código de ese ejercicio para que respete este diagrama:

Ejercicio08
- a: int - b: int
+ main(String [ ]): void + suma(): void + resta(): void + multiplicacion(): void + division(): void + setA(int): void + setB(int): void + getA(int): void + getB(int): void

Los métodos `suma()`, `resta()`, `multiplicacion()` y `division()` imprimen en pantalla el resultado.

Ahora, agregaremos que cada vez que se imprima un resultado, también muestre la marca temporal.

La salida debería ser algo así `"La suma da: " + suma + "Fecha: " + f`

Producto esperado: Ejercicio08.java

Ejercicio 14

Revisa todos los ejercicios anteriores y agrega una explicación de su funcionamiento por medio de *Javadoc*.

### Ejercicio 15

Revisa todos los ejercicios anteriores y asegúrate de que cumplen con el estándar de codificación que definimos para el curso: [Google Java Style Guide](#).

Sobre este estilo recuerda:

- variables: comienza con minúscula. lowerCamelCase.
- clases: comienza con mayúsculas, continúa con minúsculas. UpperCamelCase.
- paquetes: solo minúsculas y números.

**¿TODO PRONTO?** Bien hecho. Ahora, envía tu trabajo.

2024