



Esta ejercitación no tiene entrega obligatoria.

## 1. Vector De Ints

El objetivo de esta sección es implementar la clase *VectorDeInts*. El *VectorDeInts* es una implementación sencilla de una secuencia de enteros.

La implementación se basa en guardar los elementos en un arreglo. Cuando el arreglo se llena, lo que debe hacerse es crear un nuevo arreglo más grande, copiar los elementos del antiguo arreglo al nuevo, y reemplazar el arreglo viejo por el nuevo.

**Ejercicio 1.** Implementar las funciones `longitud`, `agregarAtras` y `obtener` de la clase *VectorDeInts*.

- El método `longitud` devuelve la cantidad de elementos en la secuencia.
- El método `agregarAtras`, extiende la secuencia con el elemento recibido por parámetro.
- El método `obtener` devuelve el elemnto en la posición `i`.

**Ejercicio 2.** Implementar el método `quitarAtras` de la clase *VectorDeInts*.

- El método `quitarAtras` elimina el último elemento de la secuencia.

**Ejercicio 3.** Implementar el método `modificarPosicion` de la clase *VectorDeInts*.

- El método `modificarPosicion` guarda el valor recibido por parámetro en la posición indicada del vector.

**Ejercicio 4.** Implementar el método `copiar` de la clase *VectorDeInts*.

- El método `copiar` retorna un nuevo vector igual al original.
- No debe haber *aliasing* entre los dos vectores.
- ¿Cuántas veces se reserva memoria para el arreglo en una invocación del método `copiar`? Dicho de otra forma, ¿cuántas operaciones `new` se realizan?

**Ejercicio 5.** Implementar el constructor por copia `VectorDeInts(VectorDeInts)` de la clase *VectorDeInts*.

- El constructor `VectorDeInts(VectorDeInts)` construye un vector de ints, igual al pasado por parámetro.
- No debe haber *aliasing* entre los dos vectores.
- ¿Cuántas veces se reserva memoria para el arreglo en una invocación del método `copiar`? Dicho de otra forma, ¿cuántas operaciones `new` se realizan?

## 2. Agenda

Vamos a programar un pequeño sistema para manejar recordatorios. Nos interesa que el sistema de recordatorio permita agendar recordatorios para una fecha y una hora dada considerando solo el lapso de un año. El sistema debe imprimir los recordatorios de la fecha actual.

## 2.1. Fecha

En el contexto del ejercicio, una fecha representa un día del año. Permite recuperar el mes y el día.

**Ejercicio 6.** Implementar el constructor `Fecha(int, int)` y los métodos `dia` y `mes` de la clase `Fecha`.

- El constructor `Fecha(int, int)` recibe por parámetro el día y el mes de la fecha a construir.
- El método `mes` devuelve el número de mes de la fecha
- El método `dia` devuelve el día de la fecha.

**Ejercicio 7.** Implementar el método `toString` de la clase `Fecha`.

- El método `toString` debe retornar un `String` con el formato `d/m`. Por ejemplo, `12/6` corresponde al 12 de junio.
- Notar que el método `toString` ya tiene una implementación por defecto y lo que debe hacerse es una sobrecarga (*override*) de la misma.

**Ejercicio 8.** Implementar el método `incrementarDia` de la clase `Fecha`.

- El método `incrementarDia` debe incrementar en un día la fecha (ignorando años bisiestos).
- Pueden aprovechar el método provisto `diasEnMes`.

**Ejercicio 9.** Implementar el método `equals` de la clase `Fecha`.

- El método `equals` debe retornar verdadero si el objeto recibido por parámetro es una fecha y es igual a la fecha original.
- Notar que el método `equals` ya tiene una implementación por defecto y lo que debe hacerse es una sobrecarga (*override*) de la misma.
- Notar que el método recibe como parámetro cualquier objeto, no necesariamente una `Fecha`.

## 2.2. Horario

**Ejercicio 10.** Implementar el constructor `Horario(int, int)` y los métodos `hora` y `minutos` de la clase `Horario`.

- El constructor `Horario(int, int)` recibe por parámetro la hora y los minutos del horario a construir.
- El método `hora` devuelve la hora del horario.
- El método `minutos` devuelve los minutos del horario.

**Ejercicio 11.** Implementar el método `toString` de la clase `Horario`.

- El método `toString` debe retornar un `String` con el formato `h:m`. Por ejemplo, `10:5` corresponde a las 10:05 hs.
- Notar que el método `toString` ya tiene una implementación por defecto y lo que debe hacerse es una sobrecarga (*override*) de la misma.

**Ejercicio 12.** Implementar el método `equals` de la clase `Horario`.

- El método `equals` debe retornar verdadero si el objeto recibido por parámetro es una fecha y es igual a la fecha original.
- Notar que el método `equals` ya tiene una implementación por defecto y lo que debe hacerse es una sobrecarga (*override*) de la misma.
- Notar que el método recibe como parámetro cualquier objeto, no necesariamente una `Horario`.

## 2.3. Recordatorio

**Ejercicio 13.** Implementar el constructor `Recordatorio(String, Fecha, Horario)` y los métodos `mensaje`, `fecha` y `horario` de la clase `Recordatorio`.

- El constructor `Recordatorio(String, Fecha, Horario)` recibe por parámetro el mensaje del recordatorio, la fecha y el horario en que ocurre el evento.
- El método `fecha` devuelve la fecha del recordatorio. No debe haber aliasing entre la fecha devuelta y la instancia del `Recordatorio`.
- El método `horario` devuelve el horario del recordatorio.
- El método `mensaje` devuelve el mensaje del recordatorio.

**Ejercicio 14.** Implementar el método `toString` de la clase `Recordatorio`.

- El método `toString` debe retornar un `String` con el formato `mensaje @ fecha horario`. Por ejemplo, `Cumple Agustin @ 6/12 17:30` corresponde al recordatorio con mensaje “Cumple Agustin”, fecha 6/12 y horario 17:30 hs.
- Notar que el método `toString` ya tiene una implementación por defecto y lo que debe hacerse es una sobrecarga (*override*) de la misma.

## 2.4. Agenda

La Agenda mantiene la fecha actual y el conjunto de recordatorios.

**Ejercicio 15.** Implementar el constructor `Agenda(Fecha)` y el método `fechaActual` la clase `Agenda`.

- El constructor `Agenda(Fecha)` recibe por parámetro la fecha del día de hoy.
- El método `fechaActual` devuelve la fecha del día de hoy según la agenda. No debe haber aliasing entre la fecha devuelta y la instancia de `Agenda`.

**Ejercicio 16.** Implementar el método `agregarRecordatorio` y `toString` la clase `Agenda`.

- El método `agregarRecordatorio` agrega un recordatorio a la agenda.
- El método `toString` debe retornar un `String` que contenga los recordatorios de la fecha actual, con formato como el del ejemplo:

```
10/5
=====
Clase Algo @ 10/5 9:0
Labo Algo @ 10/5 11:0
```

- Notar que se imprime primero la fecha actual, luego un separador de cinco = y finalmente un recordatorio por línea.
- Notar que el método `toString` ya tiene una implementación por defecto y lo que debe hacerse es una sobrecarga (*override*) de la misma.
- En este ejercicio, pueden utilizar la clase `java.util.Vector` de la biblioteca estándar de java.

**Ejercicio 17.** Implementar el método `incrementarDia` de la clase `Agenda`.

- El método `incrementarDia` incrementa en un día la fecha actual de la agenda.