INF-253 Lenguajes de Programación
Tarea 3: Java

11 de mayo de 2022

1. OOLand

Patode Hule al darse cuenta que su obsesión por los punteros no lo estaban dejando dormir, decidió irse a probar suerte en otra ciudad y en su viaje se encontró con OOLand. En esta tierra conoce a una persona llamada Cla Sehere Dada. Esta persona le contó de su empresa de programación en la cual hacen proyectos utilizando el paradigma de la orientación a objetos. Patode emocionado le pidió una oportunidad para ir a trabajar allá, una oportunidad que consiguió. Entonces Cla lo unió a su más reciente proyecto: Javation V. Patode no tiene mucha confianza en sí mismo para programar así que les pide ayuda a ustedes para realizar su trabajo.

2. Javation V

El juego Javation V consiste en un juego solitario de construcción y recursos, en el cual durante un turno el jugador puede crear Personas, Edificios y Atracciones. Al terminar el turno dependiendo de lo que tiene creado el jugador ganará recursos que le permitirán construir más cosas.

2.1. Jugador

Un Jugador consiste en una clase que tiene los siguientes atributos:

- **nombre**: el nombre del Jugador
- **javalares**: la cantidad de javalares del Jugador
- **hierro**: la cantidad de hierro del Jugador
- **trigo**: la cantidad de trigo del Jugador
- **tecnología**: la cantidad de tecnología del Jugador
- **personas**: lista de las Personas creadas por el Jugador
- **edificios**: lista de los Edificios creadas por el Jugador
- **ferias**: lista de las Ferias creadas por el Jugador
- **museos**: lista de los Museos creados por el Jugador
- **javapatos**: lista de los Javapatos creados por el Jugador

Esta clase deberá implementar los siguientes métodos:

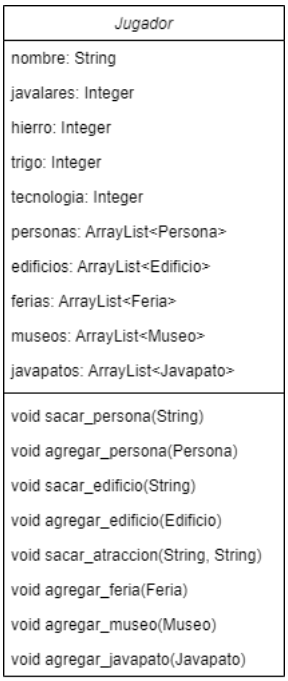
- **sacar_persona**: recibe el nombre de una Persona y debe sacarla de la lista de Personas
- **agregar_persona**: recibe una Persona y la agrega a la lista de Personas
- **sacar_edificio**: recibe el nombre de un Edificio y debe sacarla de la lista de Edificios
- **agregar_edificio**: recibe un Edificio y lo agrego a la lista de Edificios

- **sacar_atraccion**: recibe el nombre de una Atracción y de que clase corresponde ese nombre. Debe sacarla de la lista de Atracciones correspondientes
- **agregar_feria**: recibe una Feria y la agrega a la lista de Ferias
- **agregar_museo**: recibe un Museo y lo agrega a la lista de Museos
- **agregar_javapato**: recibe un Javapato y lo agrega a la lista de Javapatos

Al crear una instancia de esta clase deberá tener los siguientes valores iniciales:

- **nombre**: ingresado por consola
- **javalares**: 30
- **hierro**: 15
- **trigo**: 10
- **tecnología**: 6
- **personas**: lista vacía
- **edificios**: lista vacía
- **ferias**: lista vacía
- **museos**: lista vacía
- **javapatos**: lista vacía

Se entrega el diagrama de clase de lo desrito en esta sección.



2.2. Personas

Una Persona es una clase abstracta con los siguientes atributos:

- **nombre**: el nombre del Persona
- **edad**: la edad de la Persona
- **nivel**: el nivel del Persona
- **productividad**: la productividad de la Persona

Esta clase deberá tener implementado el siguiente método:

- **envejecer**: deberá aumentar la edad de la Persona en 1

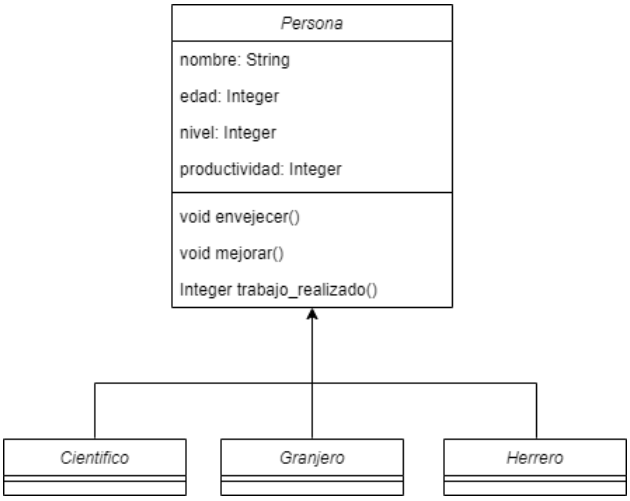
Todas las clases que hereden a Persona tendrán que partir con un **nombre** dado por consola y su **nivel** en 1. Además cuenta con los siguientes métodos abstractos que tendrá que ser implementada en cada clase hijo:

- **mejorar**: deberá aumentar el nivel en 1 y la capacidad dependiendo de la clase que lo implemente
- **trabajo_realizado**: deberá retornar el valor de su trabajo dependiendo de la clase que lo implemente

Esta clase es padre de otras tres clases: Granjero, Científico y Herrero.

- Granjero
  - Atributos:
    - **edad** inicial: 15
    - **productividad** inicial: 8
  - **mejorar**: al subir de nivel deberá aumentar su **productividad** en 2
  - **trabajo_realizado**: deberá retornar $productividad \cdot 2 + \frac{nivel}{2}$
- Científico
  - Atributos:
    - **edad** inicial: 24
    - **productividad** inicial: 10
  - **mejorar**: al subir de nivel deberá aumentar su **productividad** en 1
  - **trabajo_realizado**: deberá retornar $productividad \cdot 3 + nivel$
- Herrero
  - Atributos:
    - **edad** inicial: 18
    - **productividad** inicial: 9
  - **mejorar**: al subir de nivel deberá aumentar su **productividad** en 3
  - **trabajo_realizado**: deberá retornar $productividad + nivel \cdot 2$

Se entrega el diagrama de clase de lo desrito en esta sección.



2.3. Edificios

Un Edificio es una clase abstracta con los siguientes atributos:

- **nombre**: el nombre del Edificio
- **nivel**: el nivel del Edificio
- **capacidad**: la cantidad de Personas que puedan trabajar en el Edificio
- **personas**: una lista de Personas que trabaja en el Edificio

Esta clase deberá implementar los siguientes dos métodos:

- **sacar_persona**: recibe el nombre de una Persona y debe sacarla de la lista de Personas
- **agregar_persona**: recibe una Persona y la agrega a la lista de Personas

Todas las clases que hereden a Edificio tendrán que partir con un **nombre** dado por consola, su **nivel** en 1 y la lista de personas vacía. Además cuenta con los siguientes métodos abstractos que tendrá que ser implementada en cada clase hijo:

- **producir**: retorna la producción realizada por el edificio dependiendo de la clase que lo implemente, la cual esta representará como un arreglo de tamaño 4, donde cada posición será la producción realizada de cada recurso, las posiciones SIEMPRE corresponderán a lo siguiente:
  - Posición 0: javalares
  - Posición 1: hierro
  - Posición 2: trigo
  - Posición 3: tecnología
- **mejorar**: deberá aumentar el nivel en 1 y la capacidad dependiendo de la clase que lo implemente

Esta clase es padre de otras tres clases: Granero, Laboratorio, Herrería y Zona Común.

- Granero
  - Atributos:
    - **capacidad** inicial: 10
  - **producir**: la producción será la siguiente dependiendo de los siguientes casos:
    - Si la más de la mitad de las Personas son de tipo Granjero, entonces la producción de cada recurso realizada será:
      - javalares:
      $$\frac{\sum_{i=0}^{personas.size()-1} p[i].trabajo\_realizado()}{personas.size()}$$
      - trigo:
      $$\frac{\sum_{i=0}^{personas.size()-1} p[i].trabajo\_realizado()}{\frac{personas.size()}{2}}$$
      - tecnología: por cada dos personas tipo Científico se suma 1 a la tecnología producida
      - hierro: 0
    - En cualquier otro caso, la producción de cada recurso realizada será:
      - javalares: $-2 \cdot personas.size()$
      - hierro, trigo y tecnología: 0
  - **mejorar**: al subir de nivel deberá aumentar su **capacidad** en 3

- Laboratorio
  - Atributos:
    - **capacidad** inicial: 5
  - **producir**: la producción será la siguiente dependiendo de los siguientes casos:
    - Si la más de tres cuartos de las personas son de tipo Científico, entonces la producción de cada recurso realizada será:
      - ◇ javalares:

$$\frac{\sum_{i=0}^{personas.size()-1} es\_cientifico(p[i]) \cdot p[i].trabajo\_realizado()}{\frac{personas.size()}{cantidad\_personas\_no\_cientificas}}$$

      - ◇ tecnología:

$$\sum_{i=0}^{personas.size()-1} es\_cientifico(p[i]) \cdot p[i].trabajo\_realizado()$$

      - ◇ trigo y hierro: 0
    - En cualquier otro caso, la producción de cada recurso realizada será:
      - ◇ javalares: $-4 \cdot personas.size()$
      - ◇ hierro, trigo y tecnología: 0
  - **mejorar**: al subir de nivel deberá aumentar su **capacidad** en 2
- Herrería
  - Atributos:
    - **capacidad** inicial: 8
  - **producir**: la producción será la siguiente:
    - javalares:

$$\frac{\sum_{i=0}^{personas.size()-1} es\_herrero(p[i]) \cdot p[i].trabajo\_realizado()}{personas.size()}$$

    - hierro:

$$\frac{\sum_{i=0}^{personas.size()-1} es\_herrero(p[i]) \cdot p[i].trabajo\_realizado()}{cantidad\_personas\_no\_herreras}$$

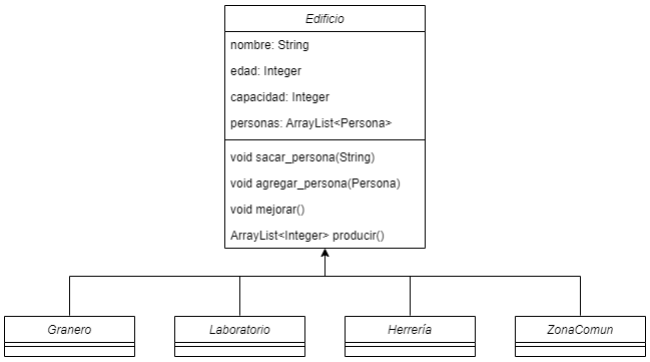
    - tecnología:

$$\frac{\sum_{i=0}^{personas.size()-1} es\_cientifico(p[i]) \cdot p[i].trabajo\_realizado() \cdot \frac{1}{3}}{cantidad\_personas\_cientificas}$$

    - trigo: 0
  - **mejorar**: al subir de nivel deberá aumentar su **capacidad** en 4
- Zona Común
  - Atributos:
    - **capacidad** inicial: 6
  - **producir**: la producción será la siguiente:
    - javalares: $personas.size()$
    - trigo, hierro y tecnología: 0
  - **mejorar**: al subir de nivel deberá aumentar su **capacidad** en 4

NOTA: las variables $es\_granjero$, $es\_cientifico$ y $es\_herrero$ vale 1 si la Persona es del tipo correspondiente y 0 si es que no

Se entrega el diagrama de clase de lo desrito en esta sección.



2.4. Atracciones

Una Atracción es una interfaz la cual tendrá el método **visitar**, donde este recibirá una lista de Personas y retornará una cantidad de javalares dependiendo de la clase que lo implemente.

Esta clase será implementada por estas tres clase: Feria, Museo, Javapato. Cada clase tendrá como atributo **nombre** que será un nombre dado por consola.

- Feria: el método **visitar** deberá retornar la siguiente cantidad de javalares:

$$2 \cdot cantidad\_personas\_granjeras + 2 \cdot \frac{cantidad\_personas\_cientificas}{2} + 2 \cdot \frac{cantidad\_personas\_herreras}{3}$$

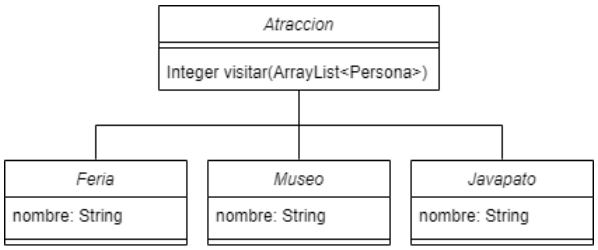
- Museo: el método **visitar** deberá retornar la siguiente cantidad de javalares:

$$4 \cdot \frac{cantidad\_personas\_granjeras}{4} + 4 \cdot cantidad\_personas\_cientificas + 4 \cdot \frac{cantidad\_personas\_herreras}{2}$$

- Javapato: el método **visitar** deberá retornar la siguiente cantidad de javalares:

$$6 \cdot \frac{cantidad\_personas\_granjeras}{2} + 6 \cdot \frac{cantidad\_personas\_cientificas}{4} + 6 \cdot \frac{cantidad\_personas\_herreras}{2}$$

Se entrega el diagrama de clase de lo desrito en esta sección.



2.5. Javation

La clase Javation será la que ejecutará el programa (clase main). Al iniciar el programa este deberá pedir por consola el nombre del Jugador y inicializar una instancia Jugador. Luego deberá comenzar el juego. Este juego es por turno, donde en cada turno el Jugador puede hacer las siguientes acciones:

- Crear Personas, Edificios y Atracciones, pero cada creación tendrá un coste asociado:

Clase	Javalares	Hierro	Trigo	Tecnología
Granjero	5	0	0	0
Científico	8	0	0	0
Herrero	6	0	0	0
Granero	10	5	5	0
Laboratorio	30	10	0	6
Herrería	20	20	0	0
Zona Común	15	3	3	3
Feria	50	25	25	10
Museo	50	30	3	30
Javapato	50	40	0	20

  NOTA: Al crear una Persona, Edificio o Atracción deberá agregarse a la lista correspondiente del Jugador.
- Mejorar las Personas y Edificios, donde cada mejora tendrá el siguiente coste asociado:

Clase	Javalares	Hierro	Trigo	Tecnología
Granjero	15	0	0	0
Científico	15	0	0	0
Granjero	15	0	0	0
Granero	20	10	15	0
Laboratorio	20	15	0	10
Herrería	20	20	0	0
Zona Común	15	10	10	10

- Puede mover una Persona particular a un Edificio en particular, esto significa sacar la persona de la lista del Jugador y agregarla a la lista de Personas del Edificio
- Puede sacar una Persona particular de un Edificio en particular, esto significa sacar la persona de la lista del Edificio y agregarla a la lista de Personas del Jugador
- Mostrar todos los nombres de Personas que aún no estén asignadas a un Edificio
- Mostrar todos los nombres de Edificios construidos, incluyendo a las personas dentro de cada Edificio
- Mostrar todos los nombres de Atracciones construidas
- Terminar el juego

Luego, el Jugador puede decidir terminar su turno y al hacer esto sucede lo siguiente:

- Todos los Edificios construidos por el Jugador deben utilizar su método producir y sumar esos recursos a los atributos correspondientes del jugador
- Todos las Atracciones construidas por el Jugador deben utilizar su método visita, recibiendo una lista de personas correspondiente a todas las personas creadas en el mundo y sumar los javalares generados a los javalares del Jugador
- Todas las Personas que hayan sido creadas deberán utilizar su método envejecer y en caso que superan los 30 años deben ser eliminadas del programa
- Mostrar a través de la salida estándar, todos los nombres de las Personas que hayan fallecido este turno
- Si el Jugador se queda con javalares negativos, el juego se termina

3. Datos de Vital Importancia

- Si alguna división no da exacta se debe redondear hacia abajo

- Se puede asumir que ningún nombre de Persona, Edificio o Atracción se repetirá
- Por cada acción, situación o cosa que sucede se debe mostrar a través de la salida estándar **TODO** lo que está pasando durante la ejecución del programa de manera **CLARA**. Por ejemplo, al usar el método producir se debería mostrar el nombre de la persona que hace uso de trabajo realizado y el valor de este
- Cada atributo debe tener un método para obtenerlo (getter) y un método para modificarlo (setter). Esto quiere decir que todos los atributos a usar deberán ser privados
- Se pueden agregar atributos, métodos y clases según lo estime conveniente, pero aquello que está presente en el enunciado debe ir sí o sí

4. A entregar

- Javation.java
- Jugador.java
- Persona.java, Granjero.java, Cientifico.java y Herrero.java
- Edificio.java, Granero.java, Laboratorio.java, Herrería.java y ZonaComun.java
- Atraccion.java, Feria.java, Museo.java y Javapato.java
- makefile
- readme.txt

5. Sobre Entrega

- Se deberá ocupar el compilador javac, con versión 13.0.7 o mayor
- El código debe venir ordenado
- Los casos de borde pueden ser manejados según lo considere adecuado, indicando claramente su acercamiento en el README y cuidando de no contradecir el resto del enunciado
- Se darán puntos por creatividad :D, máximo 10 puntos extras, si su nota supera el 100 quedará como 100.
- Cada método DEBE llevar arriba un comentario que indique nombre de la función, parámetros que recibe, una breve descripción de lo hace y lo que retorna, los getters y setters son los únicos que pueden ir agrupados, indicando un comentario que diga 'getter y setters de los atributos de la clase'
- Debe estar presente el archivo **makefile** para que se efectúe la revisión, este debe compilar **TODOS** los archivos. Si su código viene sin makefile (o el makefile no funciona), no se revisará su tarea
- El trabajo es individual
- La entrega debe realizarse en tar.gz y debe llevar el nombre: Tarea3LP_RolIntegrante.tar.gz
- El archivo README.txt debe contener nombre y rol del alumno e instrucciones detalladas para la compilación, utilización de su programa y cualquier detalle extra que se relevante. SE DESCONTARÁ SI FALTA ALGO DE LO SEÑALADO
- La entrega será vía aula y el plazo máximo de entrega es hasta el **03 de junio a las 23:55 hora aula**
- Por cada día de atraso se descontarán 20 puntos
- Las copias serán evaluadas con nota 0 y se informarán a las respectivas autoridades

6. Calificación

- Jugador (12 pts)

- No realiza la clase correctamente: omite el constructor, no programa ninguno de los métodos o no se pueden ejecutar. (0 pts)
- Realiza un constructor funcional para la clase, permitiendo inicializar los valores según como es descrito en la tarea. (MAX 3 pts)
- Implementa los métodos sacar y agregar para UN objeto (persona, edificio o atracción) realizando correctamente la funcionalidad descrita en el enunciado. (MAX 6 pts)
- Implementa los métodos sacar y agregar para DOS objetos (persona, edificio o atracción) realizando correctamente la funcionalidad descrita en el enunciado. (MAX 9 pts)
- Implementa los métodos sacar y agregar para TRES objetos (persona, edificio o atracción) realizando correctamente la funcionalidad descrita en el enunciado. (MAX 12 pts)

- Persona (4 pts)
  - No realiza la clase abstracta correctamente: no permite que las clases que hereden de esta implemente sus métodos, no sigue lo descrito en la tarea. (0 pts)
  - Realiza la clase correctamente: escribiendo los atributos que esta tiene y los métodos que las clases que hereden de esta deberían implementar. (MAX 3 pts)
  - Realiza la clase correctamente: escribiendo los atributos que esta tiene y los métodos que las clases que hereden de esta deberían implementar. Agregando la implementación del método envejecer correctamente. (MAX 4 pts)
- Científico, Granjero y Herrero (6 pts c/u)
  - Las clases no heredan de la clase padre, perdiendo el sentido en lo descrito de la tarea. Además no implementa correctamente la clase. (0 pts)
  - Realiza un constructor funcional para la clase, permitiendo inicializar los valores según como es descrito en la tarea. (MAX 2 pts)
  - Implementa correctamente UNO de los dos métodos descritos en la tarea (mejorar, trabajo_realizado) realizando correctamente la funcionalidad descrita en el enunciado. (MAX 4 pts)
  - Implementa correctamente UNO de los dos métodos descritos en la tarea (mejorar, trabajo_realizado) realizando correctamente la funcionalidad descrita en el enunciado. (MAX 6 pts)
- Edificio (5 pts)
  - No realiza la clase abstracta correctamente: no permite que las clases que hereden de esta implemente sus métodos, no sigue lo descrito en la tarea. (0 pts)
  - Realiza la clase correctamente: escribiendo los atributos que esta tiene y los métodos que las clases que hereden de esta deberían implementar. (MAX 3 pts)
  - Realiza correctamente la clase abstracta, implementando correctamente los métodos sacar_persona y agregar_persona descritos en la tarea. (MAX 5 pts)
- Granero, Laboratorio, Herrería y Zona Común (7 pts c/u)
  - Las clases no heredan de la clase padre, perdiendo el sentido en lo descrito de la tarea. Además no implementa correctamente la clase. (0 pts)
  - Realiza un constructor funcional para la clase, permitiendo inicializar los valores según como es descrito en la tarea. (MAX 2 pts)

- Implementa correctamente UNO de los dos métodos descritos en la tarea (producir, mejorar) realizando correctamente la funcionalidad descrita en el enunciado. (MAX 4.5 pts)
- Implementa correctamente UNO de los dos métodos descritos en la tarea (producir, mejorar) realizando correctamente la funcionalidad descrita en el enunciado. (MAX 7 pts)

- Atracción (3 pts)
  - No realiza la interfaz correctamente: no permite que las clases que hereden de esta implemente sus métodos, no sigue lo descrito en la tarea. (0 pts)
  - Realiza la interfaz correctamente: escribiendo los atributos que esta tiene y los métodos que las clases que implementen esta deberían implementar. (MAX 3 pts)
- Feria, Museo y Javapato (5 pts c/u)
  - Las clases no implementen la interfaz, perdiendo el sentido en lo descrito de la tarea. Además no implementa correctamente la clase. (0 pts)
  - Realiza un constructor funcional para la clase, permitiendo inicializar los valores según como es descrito en la tarea. (MAX 2 pts)
  - Implementa correctamente UNO de los dos métodos descritos en la tarea (producir, mejorar) realizando correctamente la funcionalidad descrita en el enunciado. (MAX 5 pts)
- Javation (22 pts), se divide en tres cosas:
  - Correcta inicialización (2 pts)
    - El programa no se ejecuta, no permitiendo la opción de ingresar un nombre al jugador. (0 pts)
    - El programa se ejecuta, permitiendo la opción de ingresar un nombre al jugador. (MAX 2 pts)
  - Turno del jugador (10 pts)
    - El programa no ejecuta la parte del turno del jugador. No se puede elegir ninguna de las opciones pedidas. (0 pts)
    - Por cada opción bien realizada y funcional de las descritas en la parte del turno del jugador se otorgará una cantidad máxima de 1.25 pts.
  - Post turno del jugador (10 pts)
    - El programa no ejecuta la parte del turno del jugador. No se puede elegir ninguna de las opciones pedidas. (0 pts)
    - Por cada opción bien realizada y funcional de las descritas en la parte del después del turno del jugador se otorgará una cantidad máxima de 2 pts.
- Descuentos:
  - Warning (-5 pts c/u, max -30 pts)
  - getter o setter faltante (-5 pts c/u)
  - No compila (-100pts)
  - Reglas de entrega (max -30 pts)