Academia Java - Noviembre 2022

jhonpaco@frba.utn.edu.ar Cambiar de cuenta

Borrador guardado

Ejercicios

Resuelva los siguientes ejercicios en los campos de texto provistos.
Recuerde que no es necesario resolver todos los ejercicios.

Consideraciones

1. Si bien el enunciado hace referencia a Java, se puede resolver con cualquier otro lenguaje de programación orientado a objetos ACLARANDO QUÉ LENGUAJE FUE USADO. En el caso de lenguajes multiparadigma, utilizar solo los mecanismos orientados a objetos.
2. Se recomienda que los ejercicios sean resueltos en un editor local para evitar perdida de datos por fallas de Internet. No es necesario que sea una IDE, se puede resolver en un editor de texto simple.
3. Es importante la legibilidad del código, será leído por un humano.
4. Solo implemente lo pedido en los enunciados. Asuma que el resto del código existe.

IMPORTANTE: Utilice el lenguaje orientado a objetos con el que se sienta más cómodo. El objetivo de la evaluación no es evaluar los conocimientos de un lenguaje en particular, sino el conocimiento del paradigma de programación orientada a objetos.

¿Qué lenguaje de programación utilizarás para resolver los ejercicios?

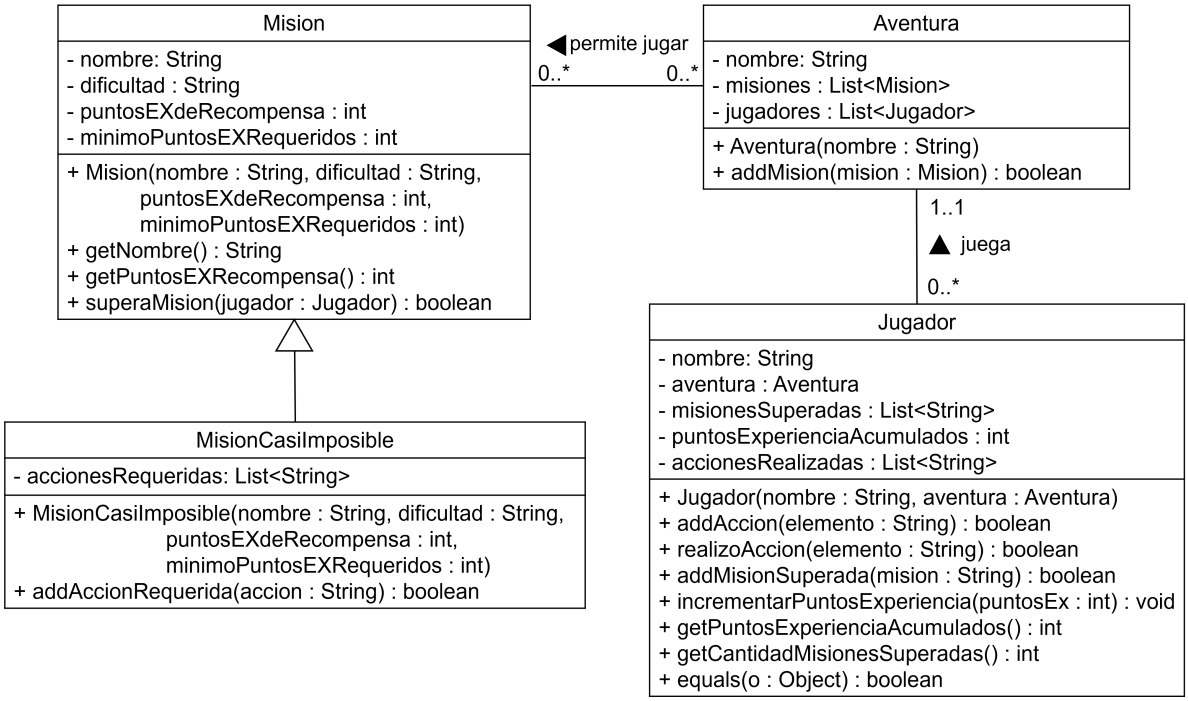
- ○ C#
- ○ C++
- ◉ Java
- ○ JavaScript
- ○ Kotlin
- ○ Python
- ○ Otro: _____

Borrar selección

Narrativa

Se tiene un sistema que modela un juego. Hasta el momento, el sistema administra jugadores y misiones. Cada jugador conoce su nivel de experiencia, acciones que realizó y misiones que superó. Por su lado, las misiones registran el nombre, dificultad, puntos de experiencia de recompensa y puntos de experiencia requeridos para realizar la misión. Además las misión es responsable de saber si un jugador supera la misión o no, basado solamente en los puntos de experiencia.
El administrador de juego desea agregar otro tipo de misiones, llamadas misiones casi imposibles, donde además de los puntos de experiencia, se desea realizar otros controles adicionales sobre el jugador. El diseñador modificó el diagrama y nos dejó instrucciones para implementar esta nueva característica del sistema.

Diagrama



NOTA

Si bien el enunciado hace referencia a Java, se puede resolver con cualquier otro lenguaje de programación orientado a objetos ACLARANDO QUÉ LENGUAJE FUE USADO. En el caso de lenguajes multiparadigma, utilizar solo los mecanismos orientados a objetos.

Ejercicio 1

Tenemos un diagrama de clases preliminar de un sistema y hay que implementar la clase **`MisionCasiImplosible`** de acuerdo con el diagrama. El resto de las clases serán implementadas por otros desarrolladores.

`addAccionRequerida(elemento: String): boolean` solo agrega una acción requerida si esta no se encuentra en la lista de acciones requeridas. Tener en cuenta que el método debe retornar **`true`** si la acción pudo ser agregada y **`false`** en caso contrario.

Una vez que se haya implementado la clase, se ejecutará el siguiente código, el cual no debe tener

errores de ejecución:

```
MisionCasiImposible m1 = new MisionCasiImposible("Mision CI", "difícil", 100, 200);
m1.addAccionRequerida("llegar al castillo");
```

Implemente la solución al Ejercicio 1 en el siguiente espacio

```
 public MisionCasiImposible(String nombre, String dificultad, int puntosEXdeRecompensa,
int minimoPuntosEXrequeridos){
      this.accionesRequeridas=new ArrayList<String>();
      this.nombre=nombre;
      this.dificultad=nombre;
      this.puntosEXdeRecompensa=puntosEXdeRecompensa;
      this.minimoPuntosEXrequeridos=minimoPuntosEXrequeridos;
   }

   public boolean addAccionRequerida(String string) {
      return this.accionesRequeridas.add(string);
   }
```

En este espacio podés dejar comentarios sobre el Ejercicio 1.

para el constructor , se setea los atributos en la clase MisionCasiImposible

en el metodo addAccionRequerida() el metodo add() es de la clase List y justo devuelve un valor booleano al agregar un elemento, si pudo o no pudo agregar un elemento

Ejercicio 2

En principio, para superar una **Mision**, el Jugador tiene que haber acumulado más puntos de experiencia que el **minimoPuntosEXRequeridos** requerido por la misión. Para ello, definieron el siguiente método en la clase **Mision**:

```
public boolean superaMision(Jugador jugador){
	return minimoPuntosEXRequeridos < jugador.getPuntosExperienciaAcumulados();
}
```

Ahora, para superar una **MisionCasiImposible** se requiere cumplir más condiciones. En este caso, el **Jugador** no solo debe haber acumulado más puntos de experiencia que el mínimo requerido por la misión, sino que también debe haber realizado todas las acciones requeridas por la misión (representadas por **accionesRequeridas: List<String>**). En caso de que la segunda condición no se cumpla, para superar la **MisionCasiImposible** la cantidad de acciones que le faltan cumplir al jugador debe ser menor a la cantidad de misiones superadas por el **Jugador.**

Tenga en cuenta el siguiente método de la clase **Jugador**:

```
public boolean realizoAccion(String elemento){
	return accionesRealizadas.contains(elemento);
}
```

Implemente los métodos necesarios para incorporar la modificación solicitada. Indique a qué clases pertenecen los métodos implementados.

Implemente los métodos necesarios para resolver el Ejercicio 2. Indique a qué clases pertenecen los métodos implementados.

```
   @Override
   public boolean superaMision(Jugador jugador) {
      boolean condicionPuntos = super.superaMision(jugador);
      if(condicionPuntos==false) return false;

      boolean
completoAccionesRequeridas=this.accionesRequeridas.containsAll(jugador.getAccionesRe
alizadas(;
      if(completoAccionesRequeridas==false){
         int cantidadDeAccionesPorCumplir=this.accionesRequeridas.size()-
jugador.getAccionesRealizadas().size();
         int cantidadDeMisionesSuperadasPorElJugador=jugador.getMisionesSuperadas();

completoAccionesRequeridas=cantidadDeAccionesPorCumplir<cantidadDeMisionesSupera
dasPorElJugador;
      }
     return  completoAccionesRequeridas;
   }
```

En este espacio podés dejar comentarios sobre el Ejercicio 2.

como se agrega mas condiciones al metodo, se sobreescribe (por eso el override), luego se comprueba si el Jugador completo las misiones imposibles () sino pasa termina el programa (devolviendo false), sino se comprueba la cantidad de acciones faltantes con la cantidad de misiones hechas

Ejercicio 3a

Una vez implementados las **Misiones** y **MisionesCasiImposibles**, se quiere implementar la funcionalidad correspondiente a agregar un **Jugador** a una **Aventura**. Para ello quieren implementar el método **public boolean agregarJugador(jugador: Jugador): boolean**. Para poder agregar un **Jugador** a la **Aventura**, este no debe haber sido previamente agregado.

Implementar esa funcionalidad teniendo en cuenta:

```
Si el jugador ya existía:
       Retornar false

Si el jugador pudo ser agregado:
       Por cada misión de la aventura:
           Si el jugador superaMision:
               Agregarle al jugador el nombre de la misión superada.
               Si la misión pudo ser agregada (addMisionSuperada
               retornó true), incrementar los puntos de experiencia del
               jugador de acuerdo a los puntos retornados por la misión.
       Retornar true
```

Implemente los métodos necesarios para resolver el Ejercicio 3a. Indique a qué clases pertenecen los métodos implementados.

```
public boolean agregarJugador(Jugador jugador){
   boolean yaExistia= jugadores.contains(jugador);
   if(yaExistia) return false;
   misiones.stream().allMatch(mision -> {
      boolean misionSuperada=jugador.addMisionSuperada(mision.getNombre());
      if(misionSuperada){
         jugador.incrementarPuntosDeExperiencia(mision.getPuntosEXdeRecompensa());
      }
      return mision.superaMision(jugador) && misionSuperada;
   });
}
```

Ejercicio 3b

Describa que fue necesario tener en cuenta para que el código desarrollado en el punto 3a pueda soportar tanto misiones y misiones casi imposible. ¿Cómo se determina que lógica utilizar para saber si se superó la misión?

Respuesta ejercicio 3b

utilice funcion lambda, mas precisamente el metodo "all() o allSatisfy()" pero en java, que seria allMatch(), que la itero para cada mision, pude haber usado polimorfirmo para que la clase Jugador tambien entienda el metodo o la condicion superaMision() pero decidi hacerlo en la clase Aventura

En este espacio podés dejar comentarios sobre el Ejercicio 3.

Tu respuesta

Ejercicio 4

En el sistema encontramos el siguiente método en el que no fueron muy claros con los nombres de métodos y variables.

```
public List<String> metodoSinNombre(){
      List<String> variableSinNombre = new ArrayList<>();
      for(Mision m : misiones){
            boolean booleanSinNombre = false;
            for(Jugador j : jugadores)
                  if(m.superaMision(j))
                        booleanSinNombre = true;
            if(!booleanSinNombre)
                  variableSinNombre.add(m.getNombre());
      }
      return variableSinNombre;
}
```

```
public static void main(String[] args) {

	Mision m1 = new Mision("buscando la olla del duende","facil",100,0);
	Mision m2 = new Mision("lobo está?","intermedio",200,100);

	Mision m5 = new MisionCasiImposible("san jorge un poroto","dificil",500,300);
	((MisionCasiImposible)m5).addAccionRequerida(“derrotar 10 dragones”)

	Juego j1 = new Juego("D&D");
	j1.addMision(m1);
	j1.addMision(m5);
	j1.addMision(m2);

	Aventura player1 = new Aventura("Juan",j1);

	Jugador player2 = new Jugador("Sebastián",j1);
	player2.addAccion("derrotar 10 dragones");

	j1.addJugador(player1);
	j1.addJugador(player2);

	System.out.println(j1.metodoSinNombre());

}
```

¿Cuál es la salida de ejecutar dicho código? NOTA: Java imprime las listas de string con el siguiente formato: [String1, String2, String3]

[" buscando la olla del duende "," lobo está ? " ,"san jorge un poroto "]

Documente la funcionalidad del metodoSinNombre

lista los todos nombres de las misiones que NO pudieron ser agregadas

En este espacio podés dejar comentarios sobre el Ejercicio 4

En este espacio podés dejar comentarios sobre el Ejercicio 4.

se pudo jhaber usado una funcion lambda dentro de otra funcion lambda

Ejercicio 5: Un poco de algoritmia

Dada una lista de enteros y un entero objetivo se debe retornar la lista de todos pares, sin repetir, de índices tal que la suma de los enteros en la lista con esos índices sea igual al valor objetivo.

Por ejemplo, dado:

```
L = [2, 6, 8 , 3, 5, 0, 2, 6]
T = 8
```

el método debe devolver:

```
[(0, 1), (0, 7), (1, 6), (2, 5), (3, 4), (6, 7)]
```

El primer elemento `(0,1)` es porque `L[0] + L[1] = 2 + 6 = 8`.
El segundo elemento `(0,7)` es porque `L[0] + L[7] = 2 + 6 = 8`.
El tercer elemento `(1,6)` es porque `L[1] + L[6] = 6 + 2 = 8`.
El cuarto elemento `(2,5)` es porque `L[2] + L[5] = 8 + 0 = 8`.
...

Nótese que el par `(1,0)` también suma 8, pero cómo se encuentra el par `(0,1)` se considera repetido.

Implemente el código necesario para resolver el Ejercicio 5.

```
L.forEach( (Integer unNumero) -> {
   indiceListaHija=0;
   L.forEach(otroNumero -> {
      if(unNumero+otroNumero==T && indiceListaHija!=indiceListaPadre){
         ArrayList<Integer> listaAux= new ArrayList<Integer>();
         listaAux.add(unNumero);
         listaAux.add(otroNumero);
         listaDeListas.add(listaAux);
         indiceListaHija++;
      }
   indiceListaPadre++;
   });
});
```

En este espacio podés dejar comentarios sobre el Ejercicio 5.

la variable [(0 , 1) , (0 , 7) , (1 , 6) , (2 , 5) , (3 , 4) , (6 , 7)]
esta almacenada en listaDeListas

En este espacio podés dejar comentarios sobre la evaluación

sugeriria hacer esta evaluacion en algun compilador online, y editable, y que funciones al momento de probar
tambien si se puede hacer uso de las funciones basicas de lambdas
como any, all, sort, fold, foldr etc (esta ultima funcion de foldr puede ser muy util para el ejercicio 5, pero me tardare tiempo en aclarar los parametros y como se usa la funcion)

seria mejor y mas rapida la evaliuacion si tuvieramos el codigo del diagrama de clases de tres lenguajes almenos y junto las respuestas añadimos el diagrama de clases resultante o final (mas que todo para ver los metodos agregados y observar mejor la responsabilidad de cada objeto) en algun software online

Envíame una copia de mis respuestas.

Atrás Enviar Borrar formulario