(SIMULACIÓN BANDA) PROGRAMACIÓN ORIENTADA A OBJETOS BÁSICA

Angel Maroto Chivite
19/01/2023

Índice

Contenido

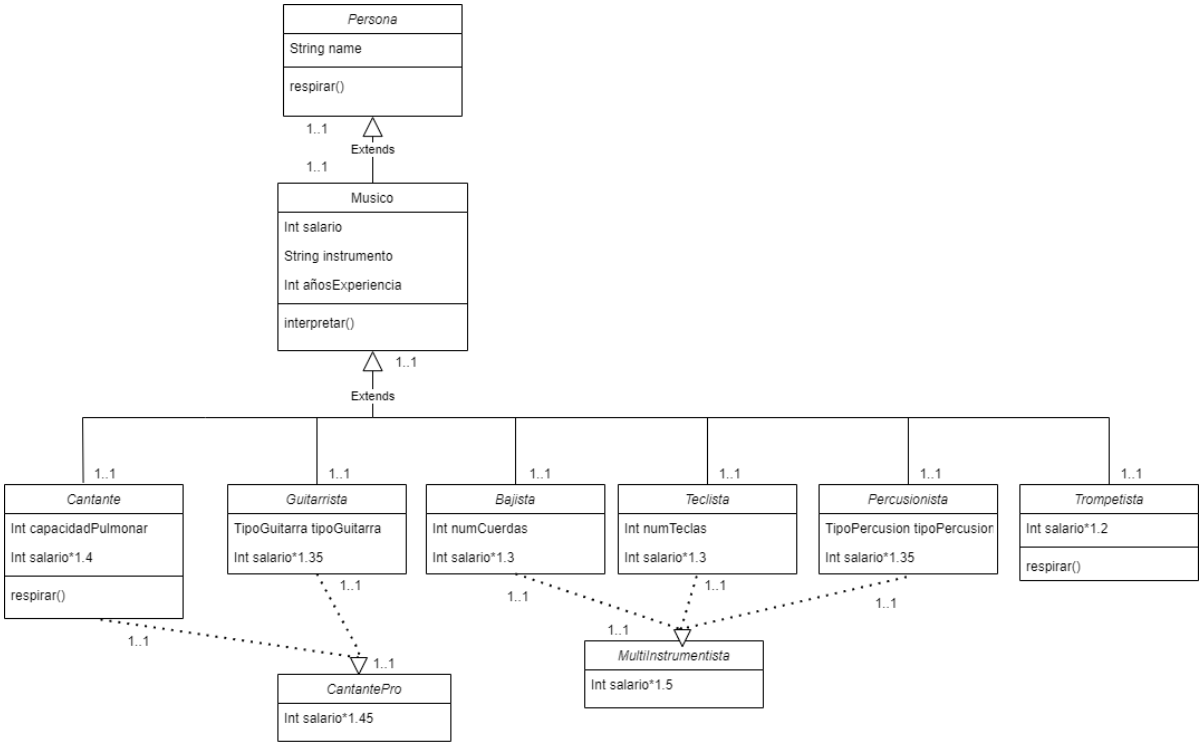
Índice 1
1. Java & Kotlin 3
2. Ejecución de la simulación 4
2.1. Instanciar objetos 4
2.2 Menú 4
2.2.1 Listar banda 4
2.2.2 Salario total (mantenimiento banda) 4
3. Pruebas unitarias 5
4. Diseño implementado 6
4.1 Enums Class 6
4.1.1 Color 6
4.1.2 TipoGuitarra 6
4.1.2 TipoPercusion 6
4.2 Factories Class 6
4.2.1 BandaFactory 6
4.2.2 BajistaFactory 6
4.2.2 CantanteFactory 6
4.2.2 CantanteProFactory 6
4.2.2 GuitarraFactory 6
4.2.2 MultFactory 6
4.2.2 PercusionistaFactory 7
4.2.2 TeclistaFactory 7
4.2.2 TrompetistaFactory 7
5. Models Class 7
5.1 Banda 7
5.2 Bajista 7
5.3 Cantante 7
5.4 CantantePro 7
5.5 Guitarra 7
5.6 Multi 7
5.7 Percusionista 7
5.8 Teclista 8
5.9 Trompetista 8

5.10 Musico 8
5.11 Persona 8

1. Java & Kotlin

Primero realicé la práctica en Java, para practicar la herencia y el polimorfismo siguiendo el árbol de herencias presentado.

Para la versión de Kotlin he utilizado la misma idea aplicando la “Sealed Class” para tener las clases de la banda más compactadas y selladas como dice la propia clase.



2. Ejecución de la simulación

2.1. Instanciar objetos

Instanciamos la Banda de Música gracias a mi BandaFactory, está definida de tal manera que se puedan asignar el tamaño de la banda y los porcentajes de probabilidad de aparición de miembros en una banda:

```
final int tammanoBanda = 50;

Musico[] vectorMusicos = new Musico[tammanoBanda];
final int probCantante = 20; //20%
final int probGuitarrista = 40; //20%
final int probBajista = 50; //10%
final int probTeclista = 60; //10%
final int probPercusionista = 75; //15%
final int probTrompetista = 80; //5%
final int probCantantePro = 95; //15%
```

Posteriormente se creará cada miembro con su correspondiente Factory, y se ejecutará un menú:

2.2 Menú

2.2.1 Listar banda

Se nos presentará en primer lugar, la cantidad de miembros total de la banda.

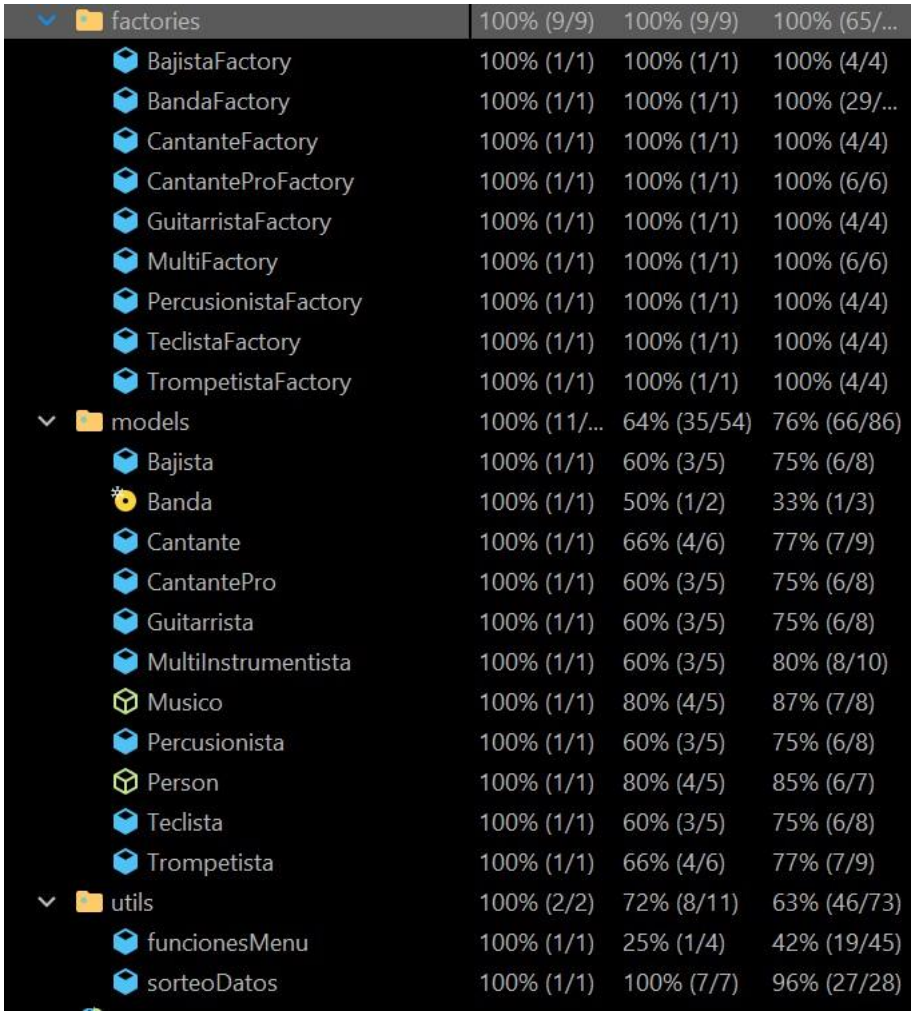
Y en segundo lugar cada miembro, con un contador por cada tipo de músico, con su correspondiente nombre, salario, aptitud del miembro, instrumento y años de experiencia

2.2.2 Salario total (mantenimiento banda)

Nos imprimirá la suma total de los salarios de cada miembro

3. Pruebas unitarias

Los Enum Class, Record y Data Class, y el código de impresión en pantalla no se encuentran probados, el resto del código está probado:



factories	100% (9/9)	100% (9/9)	100% (65/...
BajistaFactory	100% (1/1)	100% (1/1)	100% (4/4)
BandaFactory	100% (1/1)	100% (1/1)	100% (29/...
CantanteFactory	100% (1/1)	100% (1/1)	100% (4/4)
CantanteProFactory	100% (1/1)	100% (1/1)	100% (6/6)
GuitarristaFactory	100% (1/1)	100% (1/1)	100% (4/4)
MultiFactory	100% (1/1)	100% (1/1)	100% (6/6)
PercusionistaFactory	100% (1/1)	100% (1/1)	100% (4/4)
TeclistaFactory	100% (1/1)	100% (1/1)	100% (4/4)
TrompetistaFactory	100% (1/1)	100% (1/1)	100% (4/4)
models	100% (11/...	64% (35/54)	76% (66/86)
Bajista	100% (1/1)	60% (3/5)	75% (6/8)
Banda	100% (1/1)	50% (1/2)	33% (1/3)
Cantante	100% (1/1)	66% (4/6)	77% (7/9)
CantantePro	100% (1/1)	60% (3/5)	75% (6/8)
Guitarrista	100% (1/1)	60% (3/5)	75% (6/8)
MultiInstrumentista	100% (1/1)	60% (3/5)	80% (8/10)
Musico	100% (1/1)	80% (4/5)	87% (7/8)
Percusionista	100% (1/1)	60% (3/5)	75% (6/8)
Person	100% (1/1)	80% (4/5)	85% (6/7)
Teclista	100% (1/1)	60% (3/5)	75% (6/8)
Trompetista	100% (1/1)	66% (4/6)	77% (7/9)
utils	100% (2/2)	72% (8/11)	63% (46/73)
funcionesMenu	100% (1/1)	25% (1/4)	42% (19/45)
sorteoDatos	100% (1/1)	100% (7/7)	96% (27/28)

Para la versión en Java he utilizado el gestor de dependencias Maven mediante la dependencia de Test Junit5.

Para la versión de Kotlin he utilizado el gestor de dependencias Gradle mediante la dependencia de Test Junit5.

4. Diseño implementado

4.1 Enums Class

4.1.1 Color

Colores para mejorar la estética de la impresión en pantalla

4.1.2 TipoGuitarra

Tipo de guitarra que podemos encontrar en nuestra simulación:

- Eléctrica
- Acústica

4.1.2 TipoPercusion

Tipo de percusión que podemos encontrar en nuestra simulación

- Tambor
- Timbal
- Xilófono
- Campana
- Crótalos
- Celesta
- Cajón
- Triángulo

4.2 Factories Class

Creación de objetos aleatorios mediante factories.

4.2.1 BandaFactory

Creación general de la banda, definiendo el tamaño y la probabilidades con las que aparecerán los miembros.

4.2.2 BajistaFactory

Genera el nombre, los años de experiencia y cuerdas del bajo que tenga el bajista, de forma aleatoria.

4.2.2 CantanteFactory

Genera el nombre, los años de experiencia y capacidad pulmonar que tenga el cantante, de forma aleatoria.

4.2.2 CantanteProFactory

Genera el nombre, los años de experiencia, capacidad pulmonar y tipo de guitarra, que tenga el cantante, de forma aleatoria.

4.2.2 GuitarraFactory

Genera el nombre, los años de experiencia y el tipo de guitarra que tenga el bajista, de forma aleatoria.

4.2.2 MultFactory

Genera el nombre, los años de experiencia, número de teclas, tipo de guitarra y tipo de percusión que tenga el multi-instrumentista, de forma aleatoria.

4.2.2 PercusionistaFactory

Genera el nombre, los años de experiencia y tipo de percusión que tenga el bajista, de forma aleatoria.

4.2.2 TeclistaFactory

Genera el nombre, los años de experiencia y el número de teclas que tenga el teclista, de forma aleatoria.

4.2.2 TrompetistaFactory

Genera el nombre, los años de experiencia y capacidad pulmonar que tenga el trompetista, de forma aleatoria.

5. Models Class

Objetos que instanciaré a lo largo de la ejecución del programa

5.1 Banda

Record/Datal Class, únicamente almacenadora de información (POJO/POKO)

5.2 Bajista

Clase almacenadora de la información creada en su correspondiente clase factory generadora de sus campos aleatorios, y presente un auto_incrementador para calcular cuantas veces se ha creado, y un salario exclusivo.

5.3 Cantante

Clase almacenadora de la información creada en su correspondiente clase factory generadora de sus campos aleatorios, y presente un auto_incrementador para calcular cuantas veces se ha creado, un salario y una función exclusiva de respiración.

5.4 CantantePro

Clase almacenadora de la información creada en su correspondiente clase factory generadora de sus campos aleatorios, y presente un auto_incrementador para calcular cuantas veces se ha creado, y un salario exclusivo.

5.5 Guitarra

Clase almacenadora de la información creada en su correspondiente clase factory generadora de sus campos aleatorios, y presente un auto_incrementador para calcular cuantas veces se ha creado, y un salario exclusivo.

5.6 Multi

Clase almacenadora de la información creada en su correspondiente clase factory generadora de sus campos aleatorios, y presente un auto_incrementador para calcular cuantas veces se ha creado, y un salario exclusivo.

5.7 Percusionista

Clase almacenadora de la información creada en su correspondiente clase factory generadora de sus campos aleatorios, y presente un auto_incrementador para calcular cuantas veces se ha creado, y un salario exclusivo.

5.8 Teclista

Clase almacenadora de la información creada en su correspondiente clase factory generadora de sus campos aleatorios, y presente un auto_incrementador para calcular cuantas veces se ha creado, y un salario exclusivo.

5.9 Trompetista

Clase almacenadora de la información creada en su correspondiente clase factory generadora de sus campos aleatorios, y presente un auto_incrementador para calcular cuantas veces se ha creado, un salario y una función exclusiva de respiración.

5.10 Musico

Es una clase abstracta, ya que es un contenedor que nos interesa solo de él diferentes funciones o campos que se pueda adquirir como músico, en ningún momento instanciaremos un músico, pero sí a sus hijos, por ello es abstracta.

Presenta un auto_incrementador para calcular cuantas veces se ha creado, un salario base, y una función exclusiva de interpretar.

5.11 Persona

Es la clase abstracta padre inicial ya que es un contenedor que nos interesa solo de él diferentes funciones o campos que se pueda adquirir como persona, en ningún momento instanciaremos una persona, pero sí a sus hijos, por ello es abstracta.

Presenta auto_incrementador para calcular cuantas veces se ha creado y una función exclusiva de respiración.