Prueba técnica.

Hola primero de todo, felicidades por llegar a la última fase del proceso de contratación. En este punto tocará programar un poco, el objetivo del ejercicio es conocer como trabajas con una tarea real.

Te agradecemos el tiempo dedicado.

Preguntas:

Buenas prácticas:

¿Que piensas de Clean Code y los principios S.O.L.I.D ?

Define acoplamiento y cohesión en clases.

¿Qué nos indica una clase con baja cohesión?

Define el patrón Decorator:

Java:

¿Qué es un record?

¿Qué es un streams?

¿Cuando usarías un HashMap o TreeMap?

synchronized(this) vs synchronized(A.class)

Checked vs Unchecked Exception

¿Cómo funciona el paso de parámetros? (Valor vs Referencia)

SQL:

Diferencia entre inner join y left join

¿Para que sirve la instrucción HAVING?

¿Que es una regla de integridad referencial?

¿Para qué se usan los índices? Explica sus ventajas e inconvenientes en lectura y escritura.

Shell en Unix:

Cómo se edita el crontab:

Diferencia entre comillas dobles y simples:

Secuencia de comando para matar un proceso llamado PRUEBA:

Kata:

Descripción.

La tarea consiste en implementar, sin usar dependencias externas (se pueden usar sólo para testing), un asistente de contratación en memoria que ofrezca al cliente la combinación de servicios más

económica, que es aquella cuya media de los importes es mínima. Cada servicio sólo se puede combinar con su sucesor.

Por ejemplo si tenemos el siguiente listado de servicios: S1(10€), S2(5€), S3(5€).
Tendríamos las siguientes combinaciones:
S1,S2 => (10+5)/2 = 7.5
S2,S3 => (5+5)/2 = 5
S1,S2,S3 => (10+5+5) /3 = 6,67

En este caso la mejor opción para el ejercicio sería: S2,S3.

El asistente debe implementar los métodos de la interfaz *es.gossan.assistant.HiringAssistant*:
- *List<Services> searchMinimalAmount()*: Devuelve la combinación más económica del listado de servicios.
- *void add(Service service)*: Añade un servicio al listado.

Definimos un servicio como:
record Service (int id, String name, double amount){}

Ejemplos:

Caso 1:

```
[
 {"id": 0,"name":"S1","amount": 4.0},
 {"id": 1,"name":"S2","amount": 2.0},
 {"id": 2,"name":"S3","amount": 2.0},
 {"id": 3,"name":"S4","amount": 5.0},
 {"id": 4,"name":"S5","amount": 1.0},
 {"id": 5,"name":"S6","amount": 5.0},
 {"id": 6,"name":"S7","amount": 8.0}
]
```

Mejor combinación: S2,S3

Caso 2:

```
[
 {"id": 0,"name":"S1","amount": 5.0},
 {"id": 1,"name":"S2","amount": 6.0},
 {"id": 2,"name":"S3","amount": 3.0},
 {"id": 3,"name":"S4","amount": 4.0},
 {"id": 4,"name":"S5","amount": 9.0}
]
```

Mejor combinación: S3,S4

Caso 3:

```
[
 {"id": 0,"name":"S1","amount": 5.0},
 {"id": 1,"name":"S2","amount": 6.0},
 {"id": 2,"name":"S3","amount": 2.0},
 {"id": 3,"name":"S4","amount": 3.0},
 {"id": 4,"name":"S5","amount": 2.0}
]
```

Mejor combinación: S3,S4,S5

Importante: Respecto a la plantilla entregada, siéntete libre modificar lo que consideres oportuno: nombres paquetes, interfaz, cambiar el record Service a clase o la versión de Java (mín 8). Lo importante es que tu algoritmo sea eficiente y sigas buenas prácticas de programación.