Tout ce qui est sur fond rose est à remplir par le collaborateur

Collaborateur		**Date d'évaluation**	

Consignes :
Lisez bien toutes les questions avant de commencer.
Adoptez une stratégie intelligente pour optimiser votre score final.
Pour chaque question, notez son niveau de difficulté perçu par vous : facile - moyen - dur

1. Que se passe-t-il lorsqu'on appelle la méthode flush d'une session Hibernate ?

/ 0,5 - Difficulté perçue :

Hibernate synchronise l'état de sa session avec la base de données au moment du flush

2. Donnez le contrat qui doit être respecté entre hashcode et equals

/ 1,5 - Difficulté perçue :

Si deux objets sont égaux alors il doivent avoir le même hashcode, la réciproque n'est pas forcément vraie. La méthode boolean equals (Object o) permet de calculer une égalité champs à champs sur des instances de classe, la méthode int hashcode() permet de comparer des objets semblables sur un sous ensemble de champs

3. Quels soucis peut-on avoir sur un HashSet si hashcode ne remplit pas son contrat ?

/ 1 - Difficulté perçue :

Le hashSet est une collection java qui n'accepte pas les doublons, et qui utilise un algorithme de hachage pour regrouper les objets qui ont le même hashCode dans le même bucket. Soit o1 et o2 deux objets qui ne respectent pas le contrat entre equals et hashCode (deux objets égaux avec un hashCode différent), alors quand on va insérer o1 et o2 dans le hashSet ça va créer des doublons, ce qui va rompre le contrat du hashSet (n'accepte pas les doublons).

4. Quels design-patterns sont mis en place par Spring pour faciliter le développement ?

/ 0,5 - Difficulté perçue :

Injection de dépendances & AOP, Proxy,

Demander au consultant d'expliquer L'IOC (Inversion of Control Spring gère le cycle de vie de composants. Ça permet de réduire le couplage notamment)

5. Décrire le design-pattern Observer, et donner un exemple

1 / 1 - Difficulté perçue :

Plutôt que de devoir vérifier de manière répétée une information/événement auprès d'une entité (publisher), le subscriber va s'inscrire dans la liste de publication du publisher qui appellera une méthode de notification à chaque fois que ce sera nécessaire.

exemple RX et RXJS

6. Donner et expliquer par des exemples 3 design-patterns autre que Singleton, Builder et Observer

/ 1,5 - Difficulté perçue :

● AOP : Aspect Oriented Programming exemple @Transactional qui fait un try(...){...} etc et évite le boilerplate (approfondir sur est Weaver, advices pointcut etc..
)

● Validateur :
Pour valider une entité, éviter l'utilisation excessive de "if/else". Une interface ValidationRule est créée et définit une fonction validate(Object object). Ensuite pour chaque validation il faut créer une implémentation. On appelle ensuite simplement les méthodes validate sur les différentes entités. Un validateur non vérifié peut soit lever une erreur, soit ajouter un message d'erreur dans la liste des retours. Cette liste sera utilisée selon le besoin. PS : ça marche bien avec Spring car on peut @Autowire List<ValidationRule> rules

● Façade :
C'ese pour commst l'interface poussée à son paroxysme. Si une partie du code complexe utilise des outils précis sans réel intérêt pour le reste du projet, on peut décider de "cacher" le code complexe derrière une façade. C'est une interface qui est la porte d'entrée de l'usine à gaz cachée derrière. Dans notre projet on l'utiliuniquer entre les services, ça évite de devoir mettre des messages "manuellement" dans la bonne queue, avec les bons headers etc

● Factory : c'est un design pattern qui définit une interface ou une classe abstraite pour la création d'un objet mais qui laisse ses sous-classe décider quelle classe instancier. exemple :

- avoir une interface Forme avec une méthode dessiner() - des classes concrètes (Cercle, Rectangle) qui implémentent Shape - une classe ShapeFactory qui a une méthode “getShape(String shape):Shape” qui retourne une instance de Cercle/Rectangle selon l’argument passé à getShape.

● Proxy : c’est une classe qui se substituant à une autre classe, le proxy implémente la même interface que la classe à laquelle il se substitue. Il sert à ajouter du comportement à la classe qu’il substitue. exemple : on peut l’utiliser pour avoir une version simplifiée d’un objet complexe et le charger uniquement à la demande (lazy loading).

● Template: c’est une classe abstraite qui expose un modèle bien défini pour l'exécution de ses méthodes, ses sous-classes peuvent surcharger l'implémentation de la méthode selon les besoins, mais l'invocation doit se faire de la même manière que celle définie par une classe abstraite exemple : On peut utiliser ce template dans le cas d’une exécution séquentielle d’un certain nombre de méthodes, si on prend l’exemple du processus d’une commande dans une marketplace, on veut que la commande soit d’abord payée ensuite livrée

7. Donner 3 concepts de l’eXtreme Programming

/ 0,5 - Difficulté perçue :

Pair Programming
Tdd
KISS

8. À quoi sert un Proxy ?

/ 0,5 - Difficulté perçue :

C’est une classe qui se substituant à une autre classe, le proxy implémente la même interface que la classe à laquelle il se substitue. Il sert à ajouter du comportement à la classe qu’il substitue.

exemple : pour l'annotation @transactional

9. Quelles sont les nouveautés de Java 8, de Java 9, de Java 11, et de la dernière version de Java ?

/ 1 - Difficulté perçue :

● java 8:
- Les lambdas
- Les streams sur les collections nouvelles API de Date et Time (LocalDateTime)
- Api Optional pour gérer les null
- Concurrency
- Implémentation dans les interfaces des méthodes statiques et les méthodes par défaut

- Permgen remplacée par le Metaspace
- le try with resources

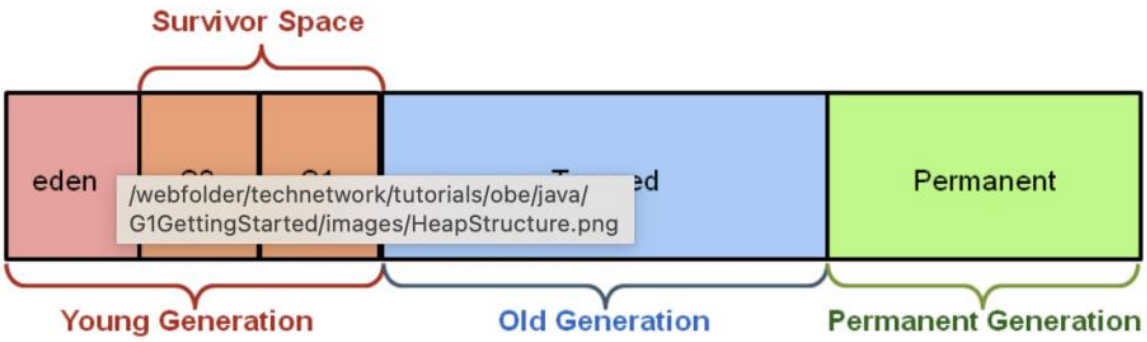
● Java 9 :
- Les modules (Jigsaw) Takima
- ClassPath est remplacé par le module-path
- variables finales dans le Try-with-resources
- méthodes privées dans les interfaces
- L'opérateur diamant dans les classes anonymes internes
- ajout d'API sur la classe Process
- fabriques pour des collections immutable
- ajout de Reactive stream avec l'API Flow
- Amélioration de la JVM - Compact String
- Concaténation des chaînes avec InvokeDynamic
- G1 utilisé comme Garbage Collector par défaut
- Segmentation du code du cache
- Unification du système de log de la JVM

● Java 14 :
- Nouvelles méthodes pour la classe String : isBlank, lines, strip, stripLeading, stripTrailing et repeat.
- Les méthodes writeString et readString de la classe Files.
- La méthode toArray dans l'interface Collection qui permets de créer un array à partir d'une collection.
- La méthode not dans l'interface Predicate.
- Utilisation de var dans les lambda ((@Nonnull var x) -> x.toUpperCase())
- Une nouvelle API HTTP qui améliore les performances générales et prend en charge les protocoles HTTP/1.1 et HTTP/2.
Java 11 introduit la notion d'imbrication de classes et les règles d'accès associées au sein de la JVM (Nest Based Access Control).
- Nous n'avons plus besoin de compiler les fichiers sources Java avec javac explicitement.

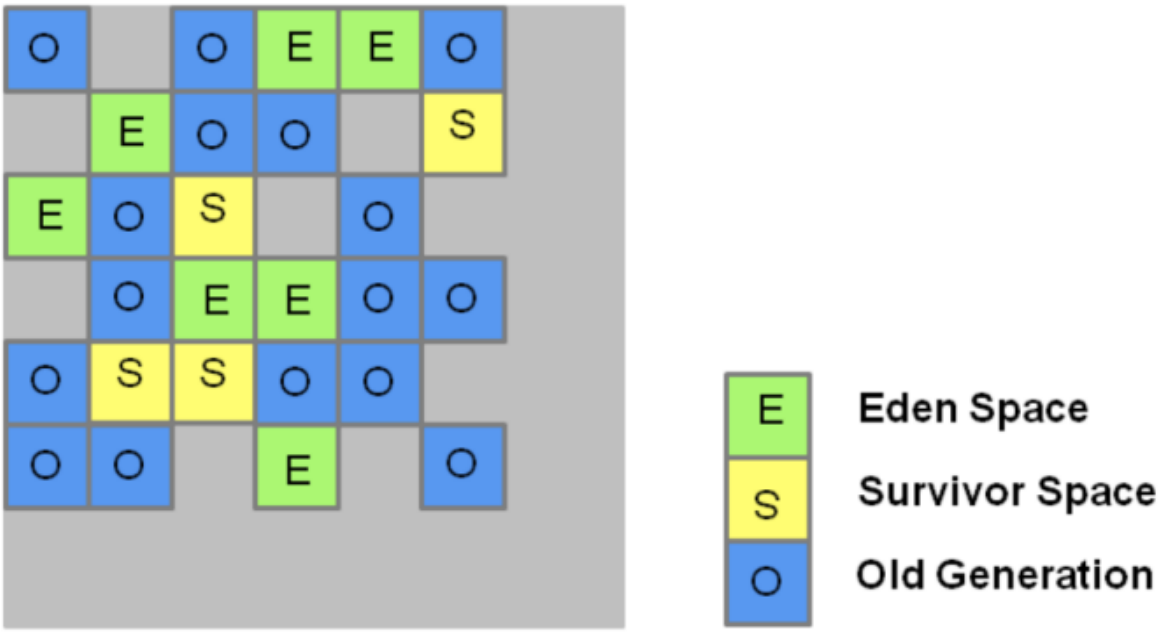
10. Expliquer le fonctionnement du Garbage Collector

/ 1,5 - Difficulté perçue :

Le garbage collector est le processus qui consiste à examiner la mémoire du tas, à identifier les objets qui sont utilisés et ceux qui ne le sont pas, et à supprimer les objets inutilisés. Le GC fonctionne en deux étapes simples: Mark et Sweep.
- Mark : c'est l'étape qui consiste à identifier les parties de la mémoire qui sont utilisées et celles qui ne le sont pas.
- Sweep : cette étape supprime les objets identifiés lors de la phase de marquage. Il existe plusieurs algorithmes du garbage collection (CMS, G1, ZGC,..). Les anciens garbages collectors (serial, parallel, CMS) structurent tout le heap en trois sections d'une taille fixe: young generation, old generation, et permanent generation.



Le G1 est prévu pour remplacer le CMS à long terme, le G1 lui partitionne le heap en un ensemble de régions de tailles égales.



Il fonctionne de manière similaire à CMS, il fait l'étape de Mark et Sweep pour les parties de la mémoire qui sont utilisées et celles qui ne le sont pas et une fois cette étape terminée, il sait quelles sont les zones mémoires qui sont susceptibles d'être remplies d'objets inutilisés pour qu'il collecte d'abord dans ces régions et libérer de la mémoire.

11. Qu’est-ce que Docker ?

/ 0,5 - Difficulté perçue :

12. Quels sont les différents scopes Maven ? Les expliquer

/ 1 - Difficulté perçue :

- compile : c’est le default scope, avec ce scope les dépendances sont disponible sur le classpath du projet et sur les opérations du build.

- provided : il est utilisé pour les dépendances qui doivent être fournies au moment de l'exécution par le JDK ou un conteneur.

- runtime : les dépendances avec ce scope sont requises au moment de l'exécution du code.

- test : utilisé pour indiquer que la dépendance n'est pas requise au moment de l'exécution standard de l'application, mais uniquement à des fins de test.

- system: très semblable au scope provided, La différence principale est qu’il doit pointer directement vers un jar spécifique sur le système.

- Import: disponible uniquement pour les dépendances de type pom. Cela indique que la dépendance doit être remplacée par les dépendances déclarées dans le pom.

13. Expliquer ce qu’est l’AOP, et les concepts associés

/ 1 - Difficulté perçue :

L’AOP (Aspect Oriented Programming), est un paradigme de programmation complémentaire à l’OOP qui consiste à séparer le code business et les aspects techniques, il est utilisé par exemple pour le monitoring, gestion d’accès et gestion des transactions. Il repose sur les concepts suivant:

- JoinPoint: l’endroit où appliquer le traitement
- PointCut: la règle de sélection du joinPoint
- Advice: indique quand il faut appliquer l’aspect (@Before, @After, @Around, …).

14. Donner un cas d'utilisation pratique du cache d'Hibernate (L1 ou/et L2)

/ 1 - Difficulté perçue :

Le cache L1 est le cache qui est lié à la session hibernate, il consiste à réserver la donner déjà demandée par l'entityManager pour un usage ultérieur. exemple: soit Order une entité déjà persister dans la base, o1 une order avec id = 1 et em un entityManager. Quand on va appelé em.find(Order.class, 1); o1 sera chargé dans le cache L1 donc si on rappelle em.find(Order.class, 1) une seconde fois il o1 sera récupéré du cache L1 et il y'aura pas de 2ème requête

15. Qu'est-ce que Spring ? Quels sont les modules de Spring que vous avez déjà utilisés ?

/ 0,5 - Difficulté perçue :

Spring est un framework qui facilite le développement d'applications web et qui ramère entre autres l'AOP et l'inversion de contrôle.
- Spring Boot
- Spring Data
- Spring MVC
- Spring Batch
- Spring web
- Spring Security
- Spring test

16. Combien de collections existe-t-il dans le JDK (héritant de l'interface Collection) ? Décrire leurs caractéristiques et différences

/ 1 - Difficulté perçue :

Queue : first in first out
List : Collection indexée et ordonnée d'éléments
Set : Collection d'éléments uniques non indexés

17. Quelle est la différence entre une ArrayList et une LinkedList ? Dans quels cas utiliseriez-vous l'une ou l'autre ?

/ 1,5 - Difficulté perçue :

La LinkedList est une liste chainée, elle ne peut contenir deux fois le même élément, elle peut ajouter un élément à la fin plus rapidement qu'une List car elle dispose d'un pointeur. La List est un tableau qui peut être un peu plus lent lors d'un ajout qui dépasse la taille de la liste si elle n'est pas définie avant. On utilise un linkedList si on veut pas accepter les doublants et si

on veut accéder rapidement au dernier élément ajouté. On utilise une liste si on accepte les doublants et on veut accéder rapidement à n'importe quel élément de la liste.

18. Nommez 4 DBs NoSQL. Expliquer leur spécialité ou leur type

/ 0,5 - Difficulté perçue :

- mongoDB: utilise des documents de type JSON pour stocker toutes les données.
- Cassandra: Cassandra est un système de stockage de données distribué pour le traitement de très grandes quantités de données structurées.
- Redis: base de données clé-valeur.
- HBase: base de données distribuée et non relationnelle qui est conçue pour la base de données BigTable par Google.

19. On vous reporte un "bug en production". Décrire votre stratégie et votre approche pour résoudre ce bug.

/ 0,5 - Difficulté perçue :

- Prendre des infos sur le bug (quand, où, comment, qui etc..)
- qualifier le bug, sa criticité
- Inspecter les logs
- Essayer de reproduire en local + Faire un IT/TU
- Lancer le test en débug pour comprendre le pb et le fix.
- Itérer la dernière étape jusqu'à ce que le bug soit fix
- suivre le cycle de déploiement

20. Comment pouvez-vous communiquer entre différents services ? Ou entre un client et un serveur ? Donner au moins 4 technologies différentes en exemple

/ 1 - Difficulté perçue :

- Par call API (http/SOAP)
- Par event (RabbitMQ / kafka)
- Par réseau (sockets)

21. Remplir le tableau suivant avec la dernière version que vous avez manipulé

/ 1 - N/A si non manipulé - Difficulté perçue :

Outil	Spring	Spring Boot	Hibernate	Maven	Gradle	PostgreSQL	Angular	React
Version								

Node.js	Docker	Kubernetes	Kafka	RabbitMQ	GraphQL	JUnit

22. Expliquer le concept du TDD. Donner un exemple avec la manière dont vous mettriez en œuvre le test d'un Service.

/ 0,5 - Difficulté perçue :

(Make it work, make it fail, make it better).

Le TDD consiste à commencer par écrire les tests. Ceux-ci doivent échouer.
Dès qu'un test échoue, on code la fonctionnalité le plus rapidement possible pour le faire passer.
Dès que le test est vert on refactore notre code.
On itère sur ces steps jusqu'à ce que toutes les fonctionnalités soient développées.

23. Aborder les différents types de tests que vous connaissez, et expliquez leur intérêt

/ 1 - Difficulté perçue :

unitaire : rapide (à écrire et exécuter) et simple, il vient tester le comportement d'une méthode en particulier. Il est indépendant des autres classes.
intégration : plus long qu'un tu, il permet de tester une fonctionnalité à travers une ou plusieures couches de l'application. Il permet de vérifier les interactions entre les couches et de se protéger contre les effets de bords.
test end 2 end: ils permettent de vérifier si l'application se comporte comme prévu.
charge : Un test permettant de simuler une forte charge sur une application pour se préparer au phases intensives (ex: black friday pour les site de vente)

24. Expliquer 3 manières de faire de la programmation Asynchrone en utilisant uniquement les APIs ECMAScript

/ 0,5 - Difficulté perçue :

Promesses
Await
Observables

25. Expliquer la différence entre une function et une arrow-function en ECMAScript

/ 1 - Difficulté perçue :

Une arrow function () => {} récupère implicitement le contexte d'appel dans son scope.Pour reproduire ce comportement avec une fonction classique, il faudrait y attacher un binding: function().bind(this).

26. Expliquer ce qu'est une sourcemap

/ 0,5 - Difficulté perçue :

La sourcemap est un mapping pour faire la transition entre le bundle et les fichiers sources. Cela permet d'afficher du code de-minifié etc..

27. Donner au moins 4 éléments / points d'attention permettant d'optimiser les performances d'un site web

/ 1 - Difficulté perçue :

Lazy Loading
Server Side Rendering
Images en webp
Attention à bootstrap => Importer uniquement les lib utiles/utilisés

28. Quel est l'intérêt d'avoir un bundler ?

/ 0,5 - Difficulté perçue :

Cela permet de séparer le code écrit par le développeur (user friendly) de celui interprété par le navigateur (machine friendly). Cela permet aussi de minifier/ugglyfier le code

29. Qu'est-ce qu'un web-worker ? Donner un exemple de cas d'utilisation

/ 0,5 - Difficulté perçue :

Un web-worker est donc une façon d'appeler un script JavaScript qui va s'exécuter dans un Thread différent de celui du script qui le crée. Il est possible pour les deux scripts de communiquer ensemble via l'envoi et la réception de messages
ex: traitement lourd de calcul

30. Quels sont les différents types en JS ?

/ 0,5 - Difficulté perçue :

boolean, number, string, object, null, undefined, Symbol