JAVA ASSESSMENT

1- Which of the below do you have experience with?

- [x] J2EE Framework
- [x] Spring Boot
- [x] Hibernate
- [x] SOAP API
- [x] REST API
- [x] Collections API
- [x] Distributed Caching
- [x] Testing Automation
- [x] Code Quality / Static Code Analysis Tools
- [x] Git
- [ ] RTC
- [x] Continuous Integration / Deployment (Jenkins, Bamboo, Teamcity etc.)
- [ ] Struts

2- Which of the following statements are true?

- [x] You must override hashCode when you override equals
- [ ] You must override equals when you override hashCode
- [ ] You must always implement both hashCode and equals
- [x] If two objects are the same according to equals (), they must have the same hashCode

3- Explain Java *hashcode* and *equal,* and when to use these.

- equals is use when we want to compare contents equality of 2 objects (non-primitive)
- Hash type collection such as Hashset, Hashmap, Hashtable, using hashcode to locate the correct bucket to use.
- But since even 2 objects having same hashcode, might not necessary is equal object, so it use 'equals' to find the exact element
- According to java contract of 'hashcode', we must override hashcode when overrides equals, but if you didn't use hash type data structure, it is fine to not override hashcode but only override equals for non-hash comparison of objects.

4- Explain immutable objects.

- After object first initialize with some value, it will never change, u can't change it.
- If you wish to change it, you must create a new copy of the objects and point the refence to new object.

5- What is the difference between Set and List?

-set not allow duplicates but list allow
- List is in ordered collection, which mean it follow your insertion seqaence, but set itself not ordered collection, it doesn't maintain order, unless you use linkedhashset, then it will guarantee order for you
- set doesn't offer access through index position while list yes

6- How do you sort a collection in Java?

- The object in collection you want to sort must implements comparable interface, then only after that we can use collections.sort / Arrays.sort, because it use compareTo(x) to do the comparison behind the scene, otherwise your sorting won't work.
- Or alternatively, since java 8, we can use lambda expression, and provide our own Comparator directive without implement of class

7- What is the different between StringBuilder and StringBuffer? When is it preferable to use them over concatenating strings with the '+' operator?

- StringBuffer is thread-safe while StringBuilder not thread-safe, thus StringBuilder is faster than StringBuffer
- StringBuffer and Builder is mutate while string not mutable, so use Buffer and Builder over '+' operator, when there are frequent change of the string value for example in loop, so that it can work faster when combine the string together.
- When work with small number of string, only we will use '+' operator over Builder of Buffer

8- Write the code for a singleton class?

this is eager initialization ✓

```
public class SingletonClass {
    private static SingletonClass instance=new
                                        SingletonClass;

    private SingletonClass() {}
    public static SingletonClass getInstance() {
        return instance;
    }
}
```

9- Explain *LazyInitializException* in Hibernate and how to solve it.

When the parent entity didn't fetch child entity (fetchop) but access the child object before even it got initialized because the child entity was set as Lazy fetch type.

- Many ways to solve it, but most common in my current project is doing DTO project or join fetch

10- Explain *N+1* problem in Hibernate and how to solve it.

- It happens when u call the lazy-fetch type child entity for every single record of parent entity, end up n-extra queries are required to lazy load all the child entity and try to imagine if you have 1000 parents ? this will definitely kill the performance
- We can solve it by fetching lazy fetch type with batch size of child object Entity
- We can use subselect for subquery so that we can use parent entity subquery to return the foreign key (parent key), and use the subqueries result as in clause when fetch the lazy child entity

11- Explain allow – bean – definition – overriding in Spring framework.

- This happen when we define multiple bean that having same method name but didn't define unique bean name
- So the Spring Boot get confuse the bean name conflict
- This might happen when the conflict is come from 3rd party libraries instead of custom application created bean

12-What is the different between *BeanFactory* and *ApplicationContext* in Spring framework?

- ApplicationContext is subclass extends from Beanfactory, so it support far more features over Beanfactory
- For example Beanfactory only support Lazy loading while Application context support eager loading by default when WC ~~cont~~ container started
- BeanFactory doesnot support IIBN while Application context do
- BeanFactory does not support @annotation based injection
- If u don't want all bean pre-instantiate when startup, then can go for BeanFactory

13-Explain the different types of bean injection in Spring and which one do you prefer? Why do you prefer it?

- ~~So~~ Constructor Based vs setter Based
- Constructor Beseed set the field required by the Bean through Bean object creation time while setter is set after calling default construction of Bean object, can set the field/state later on.
- I prefer Constructor Based injection for easier maintainance of code, because throughout application, we will have thousand billion of Bean need to instantiate, so there always happened that some the developer forgot to instantiate the dependency and use it directly, it might cause null pointer ~~exp~~ exception
- But when the application startup might ~~ca~~ be slower
- But still need depends on project consistency, or project domain nature, if we need more flexibity for the Bean we used can customized the bean + value that assigned afterwards.

14-Explain Spring Bean scopes.

- Singleton, Prototype, request, session, global-session
- First 2 type is most commonly used, while the last 3 only used for web-aware spring context (MVC architecture)
- Singleton creates only one instance of object and whole world is using that, it is saved as cached, this is also default scope in Spring
- Prototype scope is when each object is needed, it will create new object for u to use, this is stateful and customization.
- But prototype might slow down performance because it do new object instantiation everytime they requested from spring, so in our application, prototype is seldom use
- Request scope bean create new bean when there is any incoming request
- session mean that as long as within same session, even 2 different request still using same bean, it create new bean every time new session for