## [A] Could you please introduce yourself and tell us more about working experience?

Good morning sir!

I’m fine, thanks. Today, I’m very excited in talk with you.

So, Can I introduce my-self ?

As you see from my CV, my name is …. I have more than 4 **year experiences** as Java Software Engineer with 7+ projects joined, 3+ years with Agile Software Development (Daily meeting, planning, break task and estimate, weekly demo, …

I have chance to work with some big customer from UK market, US market or Australia market, …

Today, I’m going to share with you my experiences about some front-end and back-end technologies.

About **front-end technologies.** such as AngularJs, Here map API, Bootstrap fw, HTML5 and so on

About **back-end technologies**. I refer **3 layers** architecture.

* **The first** layer is **Presentation** layer.
  + I used to Spring MVC, JSP/Servlet to build a web application.
  + Or to build a web service,
    - I used to Spring Rest Web-services or Jersey to build a restful web-services.
    - I used to Spring Web-services or Axis to build a SOAP web-services
* **The middle** layer is **Business** layer: I used to Spring architecture to build system. This layer is helpful for handling exception, multi thread and transaction management. Sometime, I have some issue with multi thread; to handle that, I use synchronized and the collections are supported concurrency.
* **The bottom** layer is **Persistent** layer: I used to JDBC or Hibernate to manage connection to database

About Database, I used to MSSQL, MySQL, PostgreSQL and NoSQL with unstructured data.  
My strength is self-confidence and always willing to learn new things.  
So, do you want to ask me any question?  
  
**2. [A] Introduce about your recent projects, role, and responsibility?**

So, can I introduce about my latest project?

**\* Finantix**

- Finantix is an established provider of client-centric solutions for financial services, specializing in insurance, wealth management, and retail banking. These solutions help banks, insurers, and wealth managers attract new customers and maximize the potential of existing ones.

- By transforming their legacy silo systems into flexible model-driven architectures, Finantix enables agile businesses to react quickly to market changes, speed up the launch of new sales initiatives and optimize customer services.

**\* Ford-SmartParking**  
- The system aims to improve many of the negative points associated with the experience of parking vehicles in congested urban environments.   
- We use some technologies such as AngularJS, Here map API. About back-end technologies, we use Spring MVS, Spring Rest web-service and deployment our web application to IBM cloud bluemix!  
- At first, we use Google map API but after that the customer need change it to Here Map API, that is challenge ourselves because first time us use here map api and here map api is not popular.  
- Next challenge, this is IBM cloud Bluemix, we need to deployment web application to IBM bluemix but some time it not working or running slowly.

**3. Why you want to leave NT?**

- At first, NT is a good company with an active environment, good salary and a nice place for learning new knowledge. But, after almost three working years I need somethings that more challenges and apply for a new better place. I'm confident of my dev skill and my knowledge and today I'd like to apply for your job's position.

**4. Question for interviewer around his project**

What technologies will be used in your project?  
Do I need to research about some technologies?

# PART 1. BASIC

## [A] What equals() and hashCode() method respond for? How and when override them?

* equals() method: it's used to compare it-self with another object. If two objects are equals, it returns true. If not, it returns false.
* hashCode() method: provides a hash code value for the object. Hashcode is also used in hashing based collection classes such as Hashtable, HashMap, HashSet etc.
* hashCode() is a method that we override in our custom classes and its mandatory to override if the "equals" method is overridden because equal objects must have equal hash codes.

## [A] When you override hashCode, which method do you override?

* Equals, why? 🡺 refer to contract between hashcode and equals function (open source code of HashMap to view)

## [A] If two objects are not equals by equals() method 🡪 Is it require that hashcode must be the same?

* Possible, hashcode can be the same eventhough equals = false. (base on contract between hashcode and equals function)

## [A] What are differences between Deep copy and Shallow copy?

* **Deep copy:** clone object and related child object.

Ex: Deep copies duplicate everything. A deep copy of a collection is two collections with all of the elements in the original collection duplicated.

* **Shallow copy**: just copy reference address of an instance.

Ex: A shallow copy of a collection is a copy of the collection structure, not the elements. With a shallow copy, two collections now share the individual elements.

## How do we implement Shallow cloing? How do we implement Deep cloning?

* **Shallow cloning:** to implement shallow cloning we need to implement **cloneable interface** and override the **clone()** method with implementation.
* **Deep cloning:** Deep cloning can be done by two ways

**- All objects implement a clone() method**: Every object is responsible for cloning itself via its clone () method. So when the parent is called it makes calls to all the referenced objects inside the class and calls its clone method.

- **Serialization:** This is the best way to deep cloning.

Ensure that all classes in the object are serializable

-Create output stream for writing the new object and input stream for reading the same.

-Pass the object you want to copy to the output stream.

- And finally read the object using the input stream.

## Could you describe about "Strong typed"?

* **Strong type** is checking the types of variables at compile time.
* **Weak typing** is checking the types of the system at run-time.

## What are differences among String, StringBuilder and StringBuffer?

* **String**: is immutable, if you try to alter their values, another object gets created
* **StringBuffer** and **StringBuilder** are mutable so they can change their values
* **StringBuffer** is synchronous and **StringBuilder** is not. So when the application will only be accessed from a single thread then **StringBuilder** is better to use **StringBuffer**. **StringBuilder** is more efficient than **StringBuffer**.

## [A] Is String mutable? Why?

* String is immutable. Because we create a string object that means we can not change the value of string. when we assign a new value that means we creates a new object and the first object is still in memory

## [A] How to make a user defined object as immutable

* Access modifier: all attributes are private and final. No setter method.

## [A] What finally() block use for?

* To release/clean up resource after using or after throwing exception.

## [A] What is the difference between a constructor and a method?

* Constructor is a member function, doesn’t return type, name = class name, used to create objects of that class.
* Method, normal member function, have name and a return type

## [A] What if the main method is declared as private?

* Will compile, but it will not run. (it will give "main() method not public" message.)

## [A] What if the static modifier is removed from the signature of the main method?

* Will compile, but it will not run (throws a runtime error "NoSuchMethodError".)

## [A] What if I write “static public void” instead of “public static void”?

* Will compile, and will run.

## [A] What if I don’t provide the String array as the argument to the method?

* Will compile, but it will not run (throws a runtime error "NoSuchMethodError".)

## [A] What is the first argument of the String array in main method?

* The String array is empty. It does not have any element.

## [A] If I do not provide any arguments on the command line, then the String array of Main method will be empty or null?

* It is Empty but not null

## [B] How to print “Hello World” with only one line of code in Java?

* System.format(“Hello World”)

## [A] Can an application have multiple classes having main method?

* Yes

## [A] Can I have multiple main methods in the same class?

* No. the program fails to compile. The compiler says that the main() method is already defined in the class.

## [B] Do I need to import java.lang package any time? Why?

* No. Because all classes in the java.lang package are imported by default.

## [A] Can I import same package/class twice? Will the JVM load the package/class twice at runtime?

* Yes, I can import same package/class twice but JVM will load the package/class once at runtime

## [A] Does importing a package imports the sub-packages as well? E.g. Does importing com.MyTest.\* also import com.MyTest.UnitTests.\*?

* No, only classes on same package. Sub-package NO.

## [A] What is the difference between declaring a variable and defining a variable?

* String str; <- Declaring
* String str = "Hello"; *<- Declaring and Defining.*

## [A] What is the default value of an object reference declared as an instance variable?

* null

## [A] Can a top level class be private or protected?

* No, the compiler will complain that the "modifier private is not allowed here". It can public, abstract & final because they are allowed modifiers for a class.

## What is Pass by Value and Pass by reference?

* **Pass by Value**: the function or subroutine receives a copy of variable. The function or the method can’t change variable values.
* **Pass by Reference** the function or subroutine receives a pointer of the variable. And any changes to variable value are also visible outside the function or subroutine.
* JAVA is strictly Pass by Value. Java passes objects as references and those references are passed by value

## [A] Give a simplest way to find out the time a method takes for execution without using any profiling tool?

* Get time difference from **System.currentTimeMillis()** before and after execution.

## [A] What are wrapper classes? Why do we need wrapper classes?

* Java provides specialized classes corresponding to each of the primitive data types. These are called wrapper classes. They are example: Integer, Character, Double etc.
* It is sometimes easier to deal with primitives as objects. Moreover most of the collection classes store objects and not primitive data types. And also the wrapper classes provide many utility methods also. Because of these reasons we need wrapper classes. And since we create instances of these classes we can store them in any of the collection classes and pass them around as a collection. Also we can pass them around as method parameters where a method expects an object.

## [A] What is the difference between the instanceof and getClass, these two are same or not?

* **instanceOf**, operator
* **getClass**(), method

## [A] What is a static block?

* Executed first when an object is instantiated

## What is the difference between equals() and “==” ?

* Equals is intended to check logical equality and "==" checks if both references point to same object.

# PART 2. JAVA OOP

## What are 3 important concepts in Java?

* Write one, run anywhere (Platform Independent)
* Object Oriented Programming
* Robust and secure
* Interpreted
* Multi-threaded

## [A] Compare Object vs Class vs Instance?

* An object is a software bundle of related state and behavior.
* A class is a blueprint/template or prototype from which objects are created.
* An instance is a single and unique copy of a class that representing an Object.

## [A/B] Describe the principles of OOP? Explain for each principle?

* Encapsulation: Encapsulate states, behaviors in a class and restrict access by access modifiers.

*Ex: Class Student have 2 attributes:* ***name*** *and* ***age****. 2 attributes are private and provide access to 2 attributes via public method getter/setter.*

* Abstraction: An abstract class is a class that is declared abstract keyword – it may or may not include abstract methods. Abstract classes cannot be instantiated, but they can be sub-classed. Abstract classes let you define some default behavior and force subclasses to provide any specific behavior.
* Inheritance: extends states, behaviors of a super class.
* Polymorphism: Polymorphism is the ability of an object to take on many forms

*Ex: A a = new B(); //B extends A*

*A a = new C(); //C extends A*

## [A] How many ways to implement inheritance in Java?

* Extends: ChildClass extends ParentClass.
* Delegation: ChildClass contain one attribute is an instance of ParentClass.

## [A] Why multiple inheritance is not supported in Java?

* Ambiguity around Diamond problem.
* Ex: A have method call(); B extends A -> overridden method call(); C extends A -> overridden call() method. If java support multiple inheritance, D extends B and C, so D should inherit that overridden method which overridden method will be used? Will it be from B or C?

## [A] Explain the different forms of Polymorphism.

* Runtime Polymorphism (or Dynamic polymorphism): Overriding
* Compile time Polymorphism (or Static polymorphism): Overloading

## [A] What is an abstract class?

* An abstract class is a class that is declared abstract keyword – it may or may not include abstract methods. Abstract classes cannot be instantiated, but they can be sub-classed. You can only subclass one abstract class. Abstract classes let you define some default behavior and force subclasses to provide any specific behavior.

## [A] What is an Interface?

* When you want to only deal with a type and does not care about the actual implementation use interfaces. If you need to change your design frequently, you should prefer using interface to abstract. You can implement any number of interfaces.

## [A/B] [IMPORTANT] What is the difference between an Interface and an Abstract class?

* Different in syntax: …
* Different in usage:
  + Interface: for optional behavior, API definition. You can implement any number of Interface
  + Abstract class: for some common behavior, all sub-class need to have. You can only sub-class extends one abstract class

## [A/B] When should I use Abstract Class, when should I use Interface?

* Give 3 examples base on answer in previous questions.
  + Example about Abstract Class
  + Example about Interface
  + Example about combination of Abstract class and Interface.

## [A] What is the ‘static’ keyword in java?

* **‘static’** keyword can be used along with variable, method, block and nested class in Java.
* **Static variable**: that means when a class is loaded the static variables are initialized
* **Static method**: that mean when a class is loaded the static methods are loaded. You can invoke without the need for creating an instance of a class. Static method can access static data member and can change the value of it
* **Static block**: Is used to initialize the static data member. It is executed before main method at the time of class loading.
* **Static class** that mean Static Nested class: When a class is defined within a scope of another class, then it becomes nested class.

## [A] What is final in Java? How is it used?

* **‘final’** keyword can be used along with variable, method and Class in Java.
* **Final variable**: You can’t change value of it;
* **Final method**: You can’t override it in sub class
* **Final class**: that means it can’t be sub classed

## Difference between final, finally and finalize

* **Final**: final keyword can be used along with variable, method and Class in Java.
* **Finally**: finally block is used to release/clean up resource after using or after throwing exception.
* **Finalize**: finalize() method, it is called by Garbage collection thread just before collecting eligible Objects

## Public final static abstract class/method ?

* Compiler doesn’t accept that because final (can’t modify) and static (initialized when class loaded) with abstract (subclass will implement it)

## [A] What is method overloading, method overriding? What are the differences?

* **Overloading** means to have two or more methods with same name in the same class with different arguments.
* **Overriding** means to have two methods has them same name and same arguments in two class, one at child class and another at parent class

## Can we overload static method in Java ?

* Yes.

Public final void a(int x) {…}

Public final void a(double x) {…}

## [A] How to prevent a method from being overridden?

* Using ‘Final’ keyword for method: public final void…

## [A] Do interface have member variables?

* Yes. And the member variables must be final.

## [A] Which modifiers are allowed for methods in an Interface?

* Using keywords: public, abstract

## [A] How to override the main method?

* No

## [A] How to invoke a superclass version of an overridden method?

* Use **super.method()**

## [A] Compare Composition, Aggregation, Association vs Inheritance?

* Inheritance: is an **“is-a”** relationship.
* Association: is a relationship between 2 objects
* Aggregation: is a special form of **association**. It is called a **“has-a”** relationship.   
  When an object contains the other object then it is called aggregation
* Composition is also a **“has-a”** relationship. It is a special form of **aggregation.**   
  When an object contains the other object, if the contained object cannot exist without the existence of container object, then it is called composition.
* For example:

Computer is an ElectronicMachine *//this is inheritance*

Computer is used by Student *//this is association*

Computer has a Screen *//this is aggregation.*

Computer has a Mainboard *//this is composition. The Mainboard can’t exist without the Computer*

## [A] State the significance of public, private, protected, default modifiers?

* public: Public class is visible in other packages, field is visible everywhere (class must be public too)
* private: Private variables or methods are visible in the same class
* protected: is visible in all classes in the same package and all subclasses of this class.
* default: is visible to all within a particular package.

## [A] What are different types of inner classes?

* When a class is defined within a scope of another class, then it becomes nested class.
* Nested classes that are declared **static** are called **static nested classes**.
* Non-static nested classes are called inner classes.

## [A] Can we declare an abstract method in a normal class? Why?

* Impossible. Because what happen if people invoke the abstract method of the instance which instantiate from this normal class

## [A] How does the Java default constructor be provided?

* If you don’t define your own constructor.

## [A] Can constructor be inherited?

* Yes, use super(…)

## [A] How are this() and super() used with constructors?

* **this()** is used to invoke a constructor of the same class.
* **super()** is used to invoke a superclass constructor.

## [A] What are the differences between Class Methods and Instance Methods?

* Class method: static method
* Instance method: normal method

# PART 3. EXCEPTION HANDLING

## [A] Is it necessary that each try block must be followed by a catch block?

* Yes. Each try block must be followed by a catch block. And Finally block is optional.

## [B] What are Checked and UnChecked Exception?

* Checked: exception occurred at **compile time** (Ex: SQLException, IOException, ClassNotFoundException, …)
* Unchecked: exception occurred at **runtime** (Ex: ArrayIndexOutOfBoundException, NullPointerException, …)

## [A] What are runtime exceptions?

* Exceptions occurred at runtime (Ex: ArrayIndexOutOfBoundException, NullPointerException, …)

## [B] What is the difference between error and an exception?

* Error – Unrecoverable error (Ex: OutOfMemory error)
* Exception – Recoverable exception (Ex: FileNotFoundException, …)

## [A] How to create custom exception?

* extends Exception

## [A] What are the different ways to handle exceptions? When should you use a particular exception handling between the one you give in your answer?

* There are two ways to handle exceptions: Try – Catch block or Throws

1. Try – Catch: You handle the problem by wrapping the desired code in a try block followed by a catch block to catch the exceptions
2. Throws: You let them handle the problem. List the desired exceptions in the throws clause of the method and let the caller of the method handle those exceptions.

* Try-catch block: owning responsibility.
* Throws: when making libraries or let them own the responsibility.

## [A] If I write return at the end of the try block, will the finally block still execute?

* Yes

## [A] If I write System.exit(0); at the end of the try block, will the finally block still execute?

* No

# PART 4. COLLECTIONS

## [A] What is Collection API?

* Set of classes and interface that allows you to store objects in a collection.

## [A] How do you traverse through a collection?

* Loops, Iterator

## [A] What is the List interface?

* **List**: Ordered collection according to insertion

## [A] What are the main implementations of the List interface?

* ArrayList, LinkedList, Vector

## [A] What is the Set interface?

* No duplicates collection

## [A] What are differences between List and Set?

* **Set** is a no duplicates collection an unordered. Not support get element by index
* **List** is an ordered collection according to insertion. Support get element by index

## [A] How to remove the duplication of the Set Collection? How to check duplicated elements in the Set?

* **Set** is a no duplicates collection so nothing to remove.
* To check duplicated elements in the Set, you have to override the equals() and hashcode() method.

## [A] How to remove the duplication of the List Collection?

* Copy to Set then copy back to List.

## [A] What are the main implementations of the Set interface?

* HashSet, TreeSet, LinkedHashSet

## [A/B] What are differences between HashSet and TreeSet?

* **Hashset** = not sync, unordered, allow NULL.
* **Treeset** = not sync, natural order, faster and no allow NULL

## [A] What is a Map?

* Map is collection with the Key Value Pair

## [A] What are the main implementations of the Map interface?

* HashMap, TreeMap, Hashtable

## Have you ever worked with MultiMap?

* A multimap is like a Map but it can map each key to multiple values.

The Java Collections Framework doesn't include an interface for multimaps, but there's Guava framework that provide MultiMap interface like:

Multimap<String, String> myMultimap = ArrayListMultimap.create();

// Adding some key/value

myMultimap.put("Fruits", "Bannana");

myMultimap.put("Fruits", "Apple");

myMultimap.put("Fruits", "Pear");

myMultimap.put("Vegetables", "Carrot");

## [A] How do you sorting a list of user-defined objects?

* Comparable, Comparator

## [A] What are the differences between the Comparable and Comparator interfaces?

* Comparable interface provides **compareTo**() method which compares itself with another object.
* Comparator interface provides **compare**() method which compares two elements

## [A] What is difference between Arrays and ArrayList?

* **Arrays**: FixedSize, primitives, Objects, multidimensional. String[] array = new String[5]
* **ArrayList**: Growable, Objects only, single dimension. List<String> list = new ArrayList<String>();

## How can we obtain an array from an ArrayList class?

* You can convert from ArrayList to traditional arrays using “toArray()” method

## [A] What are the advantages of ArrayList over arrays?

* Lot of helper methods and dynamic.

## [A] What are differences between ArrayList and Vector?

* **ArrayList** is Unsynchronized and **Vector** is Synchronized.
* When both of them need expand then A **Vector** defaults to doubling the size, while the **ArrayList** increases size by 50 percent.

## [A] What are differences between HashMap and HashTable?

* **Hashmap**: not synchronize and allows NULL
* **HashTable**: synchronized and NO allows NULL
* For synchronization, I'd recommend **ConcurrentHashMap**. Because **HashTable** is old-fashioned.

## [A] What are differences between LinkedList and ArrayList?

* **ArrayList** uses index to access elements, faster than **LinkedList**, it’s more lighter because memory address only contains the value.
* **LinkedList** is using pointers to next and previous elements, It provides link-list data structure
* **LinkedList:** we can **add/remove/get** element to the beginning or the end of list addFirst(Object obj)/addLast(Object obj); getFirst()/getLast(); removeFirst()/removeLast()

## [A/B] How do you decide when to use HashMap and when to use TreeMap?

* Use **HashMap** if you don't care of the order of the elements, and use **TreeMap** if you want your elements to be sorted

## [A] What are differences between HashMap and TreeMap?

* **Hashmap** not sync and unordered, allow NULL
* **TreeMap** not sync and ordered, NOT allow NULL

## [A] What is Set interface? TreeSet?

* **Set** is a no duplicates collection
* **TreeSet** is the main implementations of the Set interface. **Treeset** = not sync and ordered, faster and NOT allow NULL

## [A] What is Iterator? How to use it? When you use For loop, when you use Iterator?

* The Iterator interface is used to step through the elements of a Collection.
* To use an iterator to traverse through the contents of a collection, follow these steps:  
  • Obtain an iterator to the start of the collection by calling the collection's iterator() method.  
  • Set up a loop that makes a call to hasNext(). Have the loop iterate as long as hasNext() returns true.  
  • Within the loop, obtain each element by calling next().
* Iterator have remove() method to remove element from the collection while For loop only support to iterate the collection.

## [B] Why are Iterators returned by ArrayList called Fail Fast?

* While traverse the collection (using iterator/for loop) if you remove element from list, JVM will throws exception then it is called fail fast. But if you are using remove method from iterator, it is ok then it is called Fail Safe

## [A] Do List & Set have common parent?

* Yes. Collection.

## [A] Is Iterator a Class or Interface? What is its use?

* Interface, it is used to traverse the collection.

## [B] What is Collection.synchronizeList()?

* make List synchronized. It is use to make non-thread safe collection to be thread-safe

# PART 5. SERIALIZATION

## [A] What is serialization and Why serialize? And deserialization?

* **Serialization** : Transformation of objects to bytes. Save to file, database or send it over the network.
* **Deserialization** is the process of restoring these objects from bytes

## [A] How do I serialize an object to a file?

* Implement Serializable. Use ObjectOutputStream.

## [A] Which methods of Serializable interface should I implement?

* None

## [B] How can I customize the serialization process? i.e how can one have a control over the serialization process?

* Externalizable

## [A] What is the common usage of serialization?

* Data transfer

## [B] What is Externalizable interface?

* Custom serialization process

## [A] When you serialize an object, what happens to the object references included in the object?

* The serialization mechanism generates an object graph for serialization. This it determines whether the included object references are serializable or not. This is a recursive process. This when an object is serialized, all the included objects are also serialized along with the original object.

## [A] What one should take care of while serializing the object?

* Make sure that objects included are also serializable

## [A] What happens to the static fields of a class during serialization?

* Not included.

## [B] What are serialization and externalization?

* Externalizable is an interface which contains two methods readExternal and writeExternal. These methods give you a control over the serialization mechanism. And You need to Override them.
* Serialization is an interface and no need to override any method.

## [A] What version Id when serializing use for?

* serialVersionUID

## [B] What is reflection?

* To describe code which is able to inspect other code in the same system

## [A] What is transient variable?

* Not include during serialization process

# PART 6. GARBAGE COLLECTION

## [A] What is the purpose of garbage collection in Java, and when is it used?

* To removed unreferenced Objects.

## [B] How to do GC tuning?

* The JVM controls the Garbage Collector; it decides when to run the Garbage Collector. JVM runs the Garbage Collector when it realizes that the memory is running low. The behavior of GC can be tuned by passing parameters to JVM. One can request the Garbage Collection to happen from within the java program but there is no guarantee that this request will be taken care of by JVM

## [B] How many type of memory in JVM?

* Heap Memory - storage for Java Objects
* Non-Heap Memory - stores loaded class and other metadata

## [B] What are differences between Stack and Heap memory?

* **Heap** memory is used by all the parts of the application while Stack memory is used only by one thread of execution.
* When an object is created, it’s always stored in the Heap space and Stack memory contains the reference to it. Stack memory only contains local primitive variables and reference variables to objects in heap space.
* Objects stored in the heap are globally accessible while Stack memory can’t be accessed by other threads.
* Memory management in stack is done in LIFO (Last In, First Out) manner while it’s more complex in Heap memory because it’s used globally. Heap memory is divided into Young-Generation, Old-Generation etc.
* Stack memory is short-lived while heap memory lives from the start till the end of application execution.
* If no memory for created object, JVM will throws OutOfMemoryError. If no memory left in stack, JVM will throws StackOverFlowError.

## [B] What is young-generation, old-generation memory?

* Young-generation – composed of 1 eden space and 2 survivor spaces. The place where all new objects are created. When young generation is filled, garbage collection is performed. This garbage collection is called Minor GC.
* Old-generation – contains the objects that are long lived and survived after many rounds of Minor GC. Usually garbage collection is performed in Old Generation memory when it’s full. Old Generation Garbage Collection is called Major GC and usually takes longer time.

## [B] What is isolated-island in term of GC?

* Describes one or more objects having no reference to them from active parts of an application.  
  When Object A references Object B and Object B references A but no other Objects references both.

## [B] When object will be remove by GC?

* If the object cannot be reached by any live thread.

## [B] Can you force GC to remove an object? Explain in detail.

* No, we can not force garbage collector to destroy objects, but we can request it by calling Sytem.gc() or its cousin Runtime.getRuntime().gc().

# PART 7. MULTITHREADING

## Describe life cycle of thread?

* 1. Newborn state  
  2. Runnable state  
  3. Running state  
  4. Blocked state  
  5. Dead state

## [B] Describe synchronization in respect to multithreading

* With respect to multithreading, synchronization is the capability to control the access of multiple threads to shared resources. Without synchronization, it is possible for one thread to modify a shared object while another thread is in the process of using or updating that object's value. This often leads to significant errors.

## [A] Explain different way of using thread?

* Extends **Thread abstract** class: each of your thread creates unique object and associate with it
* Implements **Runnable interface**: it shares the same object to multiple threads

## [A] How to implement Java thread?

* Implements Runnable Interface and override run() method

## [B] What is thread safe?

* Thread safety is the process to make our program safe to use in multithreaded environment, there are different ways through which we can make our program thread safe.

## [B] How to implement thread safe?

* Immutable class
* Semaphore
* Synchronization is the easiest and most widely used tool for thread safety in java.
* Using ‘volatile’ keyword with variables to make every thread read the data from memory, not read from thread cache
* Using thread safe collection classes (For ex: [ConcurrentHashMap](http://www.journaldev.com/122/hashmap-vs-concurrenthashmap-%E2%80%93-example-and-exploring-iterator) for thread safety)
* Optional: Use of Atomic Wrapper classes from *java.util.concurrent.atomic* package. (For ex: AtomicInteger). Or use of locks from *java.util.concurrent.locks* package.

## [B] What is deadlock?

* Circular waiting for resource.

## [B] What is thread monitor?

* Use some tool like: Jconsole, JvisualVM to monitor Thread.

## [B] What are Critical sections, Mutex (Mutual Exclusion) and Semaphore?

* Critical sections: road intersections. Vehicles move in different directions (different threads) so these intersections need traffic lights(synchronize) to avoid accidents.
* Mutual exclusion: If in your house there is only one restroom, once someone uses it, the others must wait for you to finish your business there before using it.
* Semaphore: You are in a restaurant with limited number of tables. The reception lady will act as a semaphore. You will only be accommodated when there is an empty table. if there's none, you have to wait after others will finish eating.

# PART 8. PERFORMANCE TUNING

## [B] How to monitor JVM performance?

* Use Jconsol, jVisualVM, logging CPU+Memory+TCP open connection, GC (overhead?)

## [B] How to do JVM tuning?

* Increase head size
* Increase perm size
* Configure how long for GC to collect unused variables.

## [B][IMPORTANT] Have you ever faced to performance issues and how to solve them?

* State your own problem and your solution. If you don’t have, skip this question. But, this is IMPORTANT question. Your assessment is very high if you can answer this question.

# PART 9. SPRING SECURITY

1. **What is Spring Security?**

* Spring security framework focuses on providing both authentication and authorization in java applications. It also takes care of most of the common security attack such as CSRF, session fixation attacks.

1. **What is authentication and authorization?**

* **Authentication** – Application will check it by a login form. User will enter user name and password and these inputs will be validated by the application. Once the validation is successful, user is declared as authenticated
* **Authorization** – is to check whether user can access the application or not or what user can access and what user cannot access.

1. **How to enable web security using java configuration in spring?**

Spring security provides **@EnableWebSecurity** annotation which is used with **@Configuration**.

**@EnableWebSecurity** provides spring security configuration defined in **WebSecurityConfigurer** or **WebSecurityConfigurerAdapter**. To override security configuration we can extend **WebSecurityConfigurerAdapter** class in our java configuration class.

*@Configuration*

*@EnableWebSecurity*

*public class SecurityConfig extends WebSecurityConfigurerAdapter {}*

1. **Which filter class is needed for spring security?**

* Using XML file: we need to configure DelegatingFilterProxy in web.xml

*<filter>*

*<filter-name>springSecurityFilterChain</filter-name>*

*<filter-class>org.springframework.web.filter.DelegatingFilterProxy</filter-class>*

*</filter>*

* Or extend **AbstractSecurityWebApplicationInitializer** *(from Servlet 3)*

*public class SecurityInitializer extends AbstractSecurityWebApplicationInitializer { }*

1. **How many level Spring Security access?**

* **View:** access user role in JSP – import taglib and use **<security:authorize>** tag

<%@ taglib uri="http://www.springframework.org/security/tags" prefix="security" %>

<security:authorize access="hasRole('ROLE\_SUPERWISER')"> Your Message </security:authorize>

* **Controller:** access user role in Controller using **GrantedAuthority, SecurityContextHolder and UserDetails**

*Collection<GrantedAuthority> authorities = (Collection<GrantedAuthority>);*

*UserDetails userDetails = (UserDetails)SecurityContextHolder.getContext().getAuthentication().getPrincipal();*

* **Database**: Spring security provides **<jdbc-user-service>** tag using which we access user information from database.

*<authentication-manager>*

*<authentication-provider>*

*<password-encoder hash="sha"/>*

*<jdbc-user-service data-source-ref="dataSource" authorities-by-username-query="…" users-by-username-query=".."/>*

*</authentication-provider>*

*</authentication-manager>*

1. **What is OAuth2?**

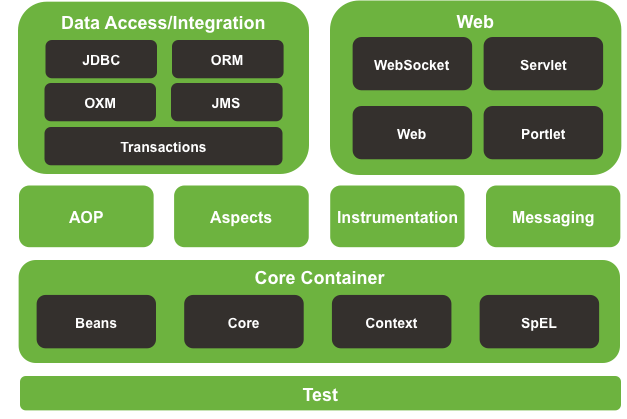
* OAuth is an open standard for authorization, which allows accessing the resources of the resource owner by enabling the client applications on HTTP services such as Facebook, GitHub, etc. It allows sharing of resources stored on one site to another site without using their credentials.

# PART 10. SPRING

## What are the advantages of Spring framework?

* Spring has layered architecture. Use what you need and leave you don't need now.
* Spring Enables POJO Programming. There is no behind the scene magic here. POJO programming enables continuous integration and testability.
* Dependency Injection and Inversion of Control Simplifies JDBC
* Open source and no vendor lock-in.

## [A] How many modules in Spring Framework?

* The Spring Framework consists of features organized into about 20 modules. These modules are grouped into Core Container, Data Access/Integration, Web, AOP (Aspect Oriented Programming), Instrumentation, Messaging, and Test, as shown in the following diagram.

## [A] How many modules do you work with?

* **Core Container**

The spring-core and spring-beans modules: using initialize bean in spring. It is Including the IoC and Dependency Injection features.

The spring-context: using manage beans in spring

* **AOP (Aspect Oriented Programming)**

The spring-aop: using define an optional feature to integrate into your app but it doesn't impact/modify your current implementation

* **Data Access/Integration**

The spring-jdbc: using manage connection pool and connect to database

The Spring-orm (including [JPA](http://docs.spring.io/spring/docs/current/spring-framework-reference/html/orm.html#orm-jpa) and [Hibernate](http://docs.spring.io/spring/docs/current/spring-framework-reference/html/orm.html#orm-hibernate)): using accessing databases through Object Relational Mapping technologies such as Hibernate.

The Spring-jms (Java message service): using process of publishing and subscribing to messages using a JMS broker.

* **Web**

The spring-web, spring-web MVC: using develop web application with Model View Controller design partten

The spring-websocket: using communication between client and server in web applications by web-sockets protocol

* **Test**

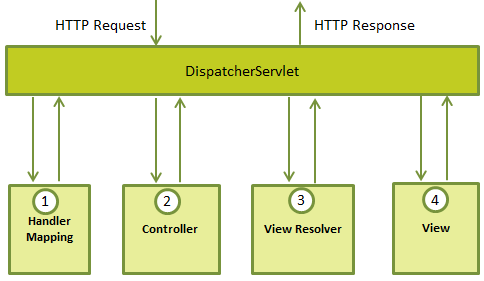
The spring-test module supports the [unit testing](http://docs.spring.io/spring/docs/current/spring-framework-reference/html/unit-testing.html) and [integration testing](http://docs.spring.io/spring/docs/current/spring-framework-reference/html/integration-testing.html) of Spring components with JUnit or TestNG.

## [A] Give me 5 reasons to use spring.

* Modularity – Spring has layered architecture. Use what you need and leave you don't need now
* Dependency Injection and Inversion of Control: removes the dependency from the programming code.
* Spring web MVC is helpful for develop web application. We can develop web application to fast
* Spring bean and spring context are helpful for initial and manage bean

## Could you decribe the life cycle of request in SpringMVC?

The Spring Web MVC framework is designed around a **DispatcherServlet** that handles all the HTTP requests and responses.

* 1. After receiving an HTTP request, **DispatcherServlet** consults the **HandlerMapping** to call the appropriate Controller.
* 2. The Controller takes the request and calls the appropriate service methods based on used GET or POST method. The service method will set model data based on defined business logic and returns view name to the **DispatcherServlet**.
* 3. The **DispatcherServlet** will take help from **ViewResolver** to pick-up the defined view for the request.
* 4. Once view is finalized, The **DispatcherServlet** passes the model data to the view which is finally rendered on the browser.

## [B] What is IoC vs DI?

* Inversion Of Control (IoC) is a design pattern that removes the dependency from the programming code. That means we have inverted the control of creating the object. We provide metadata to the IOC container either by XML file or annotation and container will create the object for us.
* Dependency Injection (DI) is a sub-type of IoC and It is implemented by Constructor Injection and Setter Injection. ID makes our programming code loosely coupled and easier for testing.

## What are different types of DI?

There are three types of dependency injection:

* **Constructor Injection** (e.g. Pico container, Spring etc): Dependencies are provided as constructor parameters.

<bean id="shape" class="com.abc.Shape">

<constructor-arg>

<ref bean="Circle" />

</constructor-arg>

</bean>

<bean id="Circle" class="com.abc.Circle" />

* **Setter Injection** (e.g. Spring): Dependencies are assigned through JavaBeans properties (ex: setter methods).

<bean id="shape" class="com.abc.Shape">

<property name="shape" ref="Circle" />

</bean>

<bean id="Circle" class="com.abc.Circle" />

* **Interface Injection** (e.g. Avalon): Injection is done through an interface.

## What are the benefits of DI?

* Minimizes the amount of code in your application.
* They make the code easier to test and maintain.
* Loose coupling is promoted with minimal effort and least intrusive mechanism.
* IOC containers support eager instantiation and lazy loading of services.

## What is IoC Container? Types of IoC containers?

* **The IoC container** is responsible to instantiate, configure and assemble the objects. The IoC container gets informations from the XML file and works accordingly.
* 2 Types of IoC containers: **BeanFactory** and **ApplicationContext**

## [B] What are important ApplicationContext implementations in Spring framework?

* The ApplicationContext (located in the org.springframework.context package) is a type of IoC container which makes Spring fw. It is extended from BeanFactory, it provides all the functionality of BeanFactory.
* An ApplicationContext provides:  
   a) Bean factory methods for accessing application components  
   b) Access to resources, such as URLs and files  
   c) The ability to publish events to registered listeners  
   d) Loading of multiple (hierarchical) contexts, allowing each to be focused on one particular layer, for example the web layer of an application
* Create ApplicationContext

*ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");*

## [B] What is BeanFactory interface?

* The **BeanFactory** (located in org.springframework.beans.factory package) is a type of IoC container which makes Spring framework. It is responsible to instantiate, configure and manages beans.
* These dependencies are reflected in the configuration data used by the BeanFactory.
* 3 ways you can configure BeanFactory:

*Resource res = new FileSystemResource("beans.xml");*

*XmlBeanFactory factory = new XmlBeanFactory(res);*

Or

*ClassPathResource res = new ClassPathResource("beans.xml");   
XmlBeanFactory factory = new XmlBeanFactory(res);*   
Or  
*ClassPathXmlApplicationContext appContext = new ClassPathXmlApplicationContext(   
new String[] {"applicationContext.xml", "applicationContext-part2.xml"});   
// of course, an ApplicationContext is just a BeanFactory  
BeanFactory factory = (BeanFactory) appContext;*

## What is the different between BeanFactory and ApplicationContext?

* The BeanFactory and the ApplicationContext interfaces acts as the IoC container.
* The ApplicationContext interface is built on top of the BeanFactory interface
* The ApplicationContext adds some extra functionality than BeanFactory such as simple integration with Spring AOP, message resource handling (for I18N), event propagation, application layer specific context (e.g. WebApplicationContext) for web application
* So it is better to use ApplicationContext than BeanFactory.

## [B] What is WebApplicationContext interface?

* The WebApplicationContext is an Interface to provide configuration for a web application. It extends from ApplicationContext and add getServletContext() method. And like generic application contexts, web application contexts are hierarchical

## What is the different between ApplicationContext and WebApplicationContext?

* The WebApplicationContext extends from ApplicationContext
* Central ApplicationContext interface to provide configuration for an application and WebApplicationContext interface to provide configuaration for a web application

## How many types of bean scopes supported by Spring? Default bean scope in Spring?

* In Spring 5 types of bean scopes supported:

- **Singleton** – Return a single bean instance per Spring IoC container. This scope is the default in Spring

- **Prototype** – Return a new bean instance each time when requested

- **Request** – Return a single bean instance per HTTP request. In a HTTP request same object will be created but if the request is new then new object will be created. Each request will have its unique object instance of bean. Applicable in case of Spring Web Context.

**- Session** – Return a single bean instance per HTTP session. If a new session is created it will have a new instance object of bean. Applicable in case of Spring Web Context.

**- GlobalSession** – Return a single bean instance per global HTTP session. It is same as HTTP session scope but applicable in portlet-based web applications.

**XML config**: <bean id="customerService" class="com.mkyong.customer.services.CustomerService" scope="prototype"/>

**Anotation**:

@Service

@Scope("prototype")

public class CustomerService {...}

* In Spring, default bean scope is Singleton.

<bean id="customerService" class="com.mkyong.customer.services.CustomerService"/>

## What’s the difference between @Component, @Controller, @Repository & @Service annotations in Spring?

* **@Component** is used to indicate that a class is a component. These classes are used for auto detection and configured as bean, when annotation based configurations are used
* **@Controller** is a specific type of component, used in MVC applications and mostly used with RequestMapping annotation.
* **@Repository** annotation is used to indicate that a component is used as repository and a mechanism to store/retrieve/search data. We can apply this annotation with DAO pattern implementation classes.
* **@Service** is used to indicate that a class is a Service. Usually the business facade classes that provide some services are annotated with this.

## [B] What is AOP (Aspect-oriented programming)?

* AOP define an optional feature to integrate into your app
* Develop AOP feature doesn't impact/modify your current implementation
* AOP can be reused among multiple component inside your app
* Spring supports the @AspectJ annotation style approach and the schema-based approach to implement custom aspects

*Ex: When a method is executed, you can add extra functionality before or after the method execution.*

*System has 10 methods; and I need write log when call each of 10 methods that.*

***If without AOP****: I need change the code in all methods. But, in future, I don't need write log for them. I need to change code in all methods and maintenance problem.*

***If using AOP****: I don't have to call methods from method (don't change the code in all methods). Now we can define the additional concern in the xml file. In future, I don't need write log for them, I need to change only in the xml file. So, maintenance is easy with AOP*

## [B] What are the 5 types of AOP Advice/What are different types of AOP Advice?

* **Before Advice** – it executes before a join point.
* **After Returning Advice** – it executes after a joint point completes normally.
* **After Throwing Advice** – it executes if method exits by throwing an exception.
* **After Advice** – it executes after a join point regardless of join point exit whether normally or exceptional return.
* **Around Advice** – It executes before and after a join point.

## [B] What is a Jointpoint vs Advice vs Pointcut?

* **Jointpoint** –is any point in your program such as method execution, exception handling, field access etc. Spring supports only method execution join point.
* **Advice** – Advice represents an action taken by an aspect at a particular join point.
* **Pointcut** – It is an expression language of AOP that matches join points.
* **Advisor** – Group ‘Advice’ and ‘Pointcut’ into a single unit, and pass it to a proxy factory object.

## Can you compare Stateless and stateful session bean?

* **Stateless Session Bean:**
* Stateless session bean these are single request business process is one that does not require state to be maintained across method invocation. Stateless session bean cannot hold the state.
* There should be one and only one create method that to without any argument in the home interface.
* Stateless session bean instance can be pooled. Therefore “n” number of beans can cater to n+1 number of clients.
* Stateless bean will not be destroyed after client has gone.
* If the business last only for a single method call, S.S.B are suitable
* Stateless session bean cannot have instance variable
* EjbRemove() method does not destroy the bean , it remains in the pooled state.
* **Stateful Session Bean**
* Statefull session bean is a bean that is designed to service business process that span multiple methods request/transaction, S.S.B can retain their state on the behalf of individual client.
* There can be one or more create methods with or without arguments in the Home Interface.
* Statefull session bean do not have pooling concept. Stateful bean will be given individual copy for every user.
* Stateful bean will be destroyed once the client has gone (or after session time out)
* If the business process spans multiple invocations there by requiring a conversational then S.S.B will be ideal choice.
* Stateful session bean will have instance variable and state is maintained in these instance variables
* Stateful bean can be destroyed by calling the ejbRemove() method

# PART 10. REST – WEB SERVICE

## [A] Have you worked with web services before? (should explain SOAP or REST)

* **Explain REST:** A RESTFul web services are based on HTTP methods and the concept of REST. A RESTFul web service typically defines the base URI (Uniform Resource Identifiers) for the services, the supported types (XML, text, JSON, user-defined, ...) and the set of operations (POST, GET, PUT, DELETE) which are supported.

So, a client can directly access a RESTful Web Services using the URIs of the resources (same as you put a website address in the browser’s address bar and get some representation as response).

* **Explain SOAP:** Simple Object Access Protocol (SOAP) is a standard protocol specification for message exchange based on XML. Communication between the web service and client happens using XML messages.

A simple web service architecture has two components: **Client** and **Service provider**

**Service provider** will create a standard XML file which will have all above information. So If this file is given to client then client will be able to access web service. This XML file is called WSDL.

## [A] What is REST and RESTful web services?

* **REST -** [**Representational state transfer:**](http://en.wikipedia.org/wiki/Representational_State_Transfer) is a style of software architecture. REST is an "architectural style" that basically exploits the existing technology and protocols of the Web.
* **RESTful** is typically used to refer to web services implementing such an architecture.

## [B] REST vs SOAP?

* **REST** is an **architectural style** to create web services.

**REST** is acronym for **REpresentational State Transfer**.

**REST** exposes methods through URIs, there are no technical details. **REST** doesn’t have any contract defined between server and client

**REST** web services are **loosely coupled**.

**REST** supports any data type such as XML, JSON, image etc.

***REST*** *learning curve is simple, POJO classes can be generated easily and works on simple HTTP methods.*

***REST*** *web services are easy to maintain when compared to SOAP, a new method can be added without any change at client side for existing resources.*

***REST*** *can be easily tested through CURL command, Browsers and extensions such as Chrome Postman.*

* **SOAP** is a standard protocol for creating web services.

**SOAP** is acronym for Simple Object Access Protocol.

**SOAP** uses WSDL to expose supported methods and technical details. **SOAP** web services and client programs are bind with WSDL contract

**SOAP** web services and client are **tightly coupled** with contract.

**SOAP** supports XML data format only

***SOAP*** *learning curve is hard, requires us to learn about WSDL generation, client stubs creation etc.*

***SOAP*** *web services are hard to maintain, any change in WSDL contract requires us to create client stubs again and then make changes to client code.*

***SOAP*** *web services can be tested through programs or software such as Soap UI.*

## [B] What is differences between RESTful web services and SOAP web services?

* **About RESTfull web services:**  
  - Lightweight - not a lot of extra xml markup  
  - Human Readable Results  
  - Easy to build - no toolkits required  
  - Uses the HTTP headers to hold meta information (although it is protocol-agnostic)  
  - Can be used with XML, JSON or whatever necessary (Usually used with JSON due to the easily parsable content)  
  - Faster than SOAP  
  - It uses semantic media types
* **About SOAP web services:**  
  - XML based  
  - Runs on HTTP but envelopes the message  
  - Slower than REST  
  - Very mature, a lot of functionality  
  - Not suitable for browser-based clients  
  - Development tools

## Differences between cookies and sessions? If browser disable cookie?

* **Sessions** store information on the server-side. Sessions start when you open browser and sent to the server with first request; So when you close your browser you also lose the session. So, if you had a site requiring a login, this couldn't be saved as a session like it could as a cookie, and the user would be forced to re-login every time they visit
* **Cookies** store information on the client-side (browser) and sent to the server with every request. Cookies have expiration time. A cookie can keep information in the user's browser until deleted. If a person has a login and password, this can be set as a cookie in their browser so they do not have to re-login to your website every time they visit
* **If browser disable cookie**: A cookie can't keep information in the user's browser so if you had a site requiring a login, you couldn't be saved information and you would be forced to re-login every time they visit.

## [A] How would you implement a rest web service? (should explain SOAP or REST/ Could you give me some steps to create a restful Web Service)

* use the following keywords in your discussion

Spring Web MVC

Dispatcher Servlet

Presentation Layer

Service Layer

Persistent Layer

Annotations

## [A] What is JAXB?

* **JAXB** stands for Java Architecture for XML Binding, is a Java standard that defines how Java objects are converted from and to XML and vice-versa. It provides mechanism to marshal (write) java objects into XML and unmarshal (read) XML into object.

## [A] How to Marshal and Unmarshal in JAXB?

* **To Marshal**

Create POJOs or bind the schema and generate the classes.

Create the content tree by using set methods.

Create a JAXBContext object. (javax.xml.bind.JAXBContext)

Create a Marshaller object. (javax.xml.bind.Marshaller)

Call the marshal method to persist the created content tree as XML document.

* **To Unmarshal**

Create POJOs or bind the schema and generate the classes.

Create a JAXBContext object. (javax.xml.bind.JAXBContext)

Create an Unmarshaller object. (javax.xml.bind.Unmarshaller)

Call the unmarshal method.

Use the get methods available in schema-genearated classes to access the values.

## [A] How can we parse Json and XML with REST?

* Parse Json using Jackson: Jackson is a simple Java-based library to serialize Java objects to JSON and vice versa.
* Parse XML using JAXB

## [A] Can you explain how to implement Restful by using Spring framework?

* (cite the libraries needed, discuss the components you need to create)

## [A] If I want a json response or xml response, what’s difference between requests?

## [A] What are differences between GET and POST method? How about other method?

(mention the supported length, data type, resource, cache)

* **GET** defines a reading access of the resource without side-effects. The resource is never changed via a GET request, e.g., the request has no side effects (idempotent).
* **PUT** creates a new resource. It must also be idempotent.
* **DELETE** removes the resources. The operations are idempotent. They can get repeated without leading to different results.
* **POST** updates an existing resource or creates a new resource.

## [A] What is WSDL?/ Explain WSDL types?

* WSDL stands for Web Service Description Language. It is an XML file that describes the technical details of how to implement a web service:

- Port / Endpoint – URL of the web service

- Input message format

- Output message format

- Security protocol that needs to be followed

- Which protocol the web service uses

## [B] What is JAX-WS and JAX-RS?

* **JAX-WS** stands for Java API for XML Web Services (SOAP). JAX-WS is a set of APIs for creating web services in XML format and specification to provide support for created RESTful Webservices.
* **JAX-RS** stands for Java API for RESTful Web Services. JAX-RS is a Java based programming language API and specification to provide support for created RESTful Webservices

## [D] What is UDDI?

* **UDDI** stands for Universal Description, Discovery and Integration. It is a directory service. Web services can register with a UDDI and make themselves available through it for discovery:

Service provider registers with UDDI.

Client searches for service in UDDI.

UDDI returns all service providers offering that service.

Client chooses service provider

UDDI returns WSDL of chosen service provider.

Using WSDL of service provider, client accesses web service.

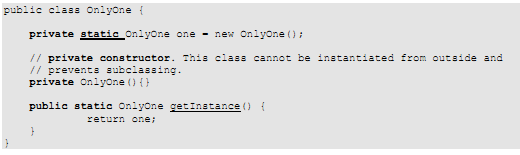
# PART 12. DESIGN PATTERN

## [A] What kind of design pattern you know?

* Singleton, Factory, Visitor, Observer, Façade, Template, Service Locator, Front Controller, IoC, MVC, …

## What is Facade pattern, Factory pattern, Singleton pattern and Observer design pattern? When you use them?

* **The Facade Pattern** provides a unified interface to a set of interfaces in as subsystem. Facade defines a higher-level interface that makes the subsystem easier to use. For example, you have a database access class with different methods to read the different tables. The client requires the complete result. You could use a facade pattern which hides the complex database access interface behind a few easy to understand and maintainable interface, e.g. load() and get().
* **Factory pattern**: you could create objects without exposing the instantiation logic to the client and refer to the newly created object through a common interface.
* **Singleton pattern**: A singleton in Java is a class for which only one instance can be created provides a global point of access this instance.



* **The observer pattern** defines a one-to-many dependency between objects so that when one object changes state, all of its dependents are notified and updated automatically.

# PART 13. DESIGN

## [A] Which tool do you use to draw diagram? Which kind of diagram do you use?

* Sequence diagram: Visio, Rational Rose.
* Activity diagram: Visio
* System Integration diagram: Visio, Power Point, Excel, Word

## Maven vs Ant

* **Ant** doesn't have formal conventions like a common project directory structure, you have to tell Ant exactly where to find the source and where to put the output. Informal conventions have emerged over time, but they haven't been codified into the product.

Ant is procedural, you have to tell Ant exactly what to do and when to do it. You had to tell it to compile, then copy, then compress.

Ant doesn't have a lifecycle, you had to define goals and goal dependencies. You had to attach a sequence of tasks to each goal manually."

* **Maven** has conventions, it already knew where your source code was because you followed the convention. It put the bytecode in target/classes, and it produced a JAR file in target.

Maven is declarative. All you had to do was create a pom.xml file and put your source in the default directory. Maven took care of the rest.

Maven has a lifecycle, which you invoked when you executed mvn install. This command told Maven to execute a series of sequence steps until it reached the lifecycle. As a side-effect of this journey through the lifecycle, Maven executed a number of default plugin goals which did things like compile and create a JAR"

# PART 14. FRONT-END

1. **What are differences between Responsive and Adaptive design?**

* **Responsive:** uses **CSS media queries** to change styles based on the target device such as display type, width, height etc., and only one of these is necessary for the site to adapt to different screens.

Responsive can be more complex but takes much less work to both build and maintain. *(more flexible and sites load faster)*

* **Adaptive:** works to **detect the screen size** and load the appropriate layout for it – generally you would design an adaptive site for six common screen widths (break points): **320, 480, 760, 960, 1200 & 1600.**

Adaptive is useful for retrofitting an existing site in order to make it more mobile friendly. And it requires more work as you have to design layouts for a minimum of six widths