1. **Java Basic**
   1. **What are the advantages of Object Oriented Programming Languages (OOPL)?**
   2. **What do you mean by polymorphism, inheritance, encapsulation?**
      * **Polymorphism** – means the ability of a single variable of a given type to be used to reference objects of different types, and automatically call the method that is specific to the type of object the variable references
      * **Inheritance** – When you want to create a new class and there is already a class that includes some of the code that you want, you can derive your new class from the existing class. In doing this, you can reuse the fields and methods of the existing class without having to write! There are two types of inheritances:
      * Implementation inheritance (aka class inheritance)
      * Interface inheritance (aka type inheritance)
      * **Encapsulation** – The values of the variables inside an object are private, unless methods are written to pass that information outside of the object. *Refers to keeping all the related members (variables and methods) together in an object. Specifying member variables as private can hide the variables and methods. Objects should hide their inner workings from the outside view. Good encapsulation improves code modularity by preventing objects interacting with each other in an unexpected way, which in turn makes future development and refactoring efforts easy.*
   3. **How does the Object Oriented approach improve software development?**
      * Re-use of previous work: using implementation inheritance and object composition.
      * Real mapping to the problem domain: Objects map to real world and represent vehicles, customers, products etc: with encapsulation.
      * Modular Architecture: Objects, systems, frameworks etc are the building blocks of larger systems.
   4. **Interface?**
      * In the Java programming language, an *interface* is a reference type, similar to a class, that can contain *only* constants, method signatures, default methods, static methods, and nested types. You want the base class to present only an interface for its derived classes. This means, you don’t want anyone to actually instantiate an object of the base class. You only want to upcast to it (implicit upcasting, which gives you polymorphic behavior), so that its interface can be used.
      * **When to use an interface?** –when you wants 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
   5. **Abstract**
      * 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 subclassed. If anyone tries to make an object of an abstract class, the compiler prevents it.
      * **When to use an Abstract?** –Abstract classes are excellent candidates inside of application frameworks. Abstract classes let you define some default behavior and force subclasses to provide any specific behavior.
   6. **Interface vs Abstract class?**
      * **Abstract –** Have executable methods and abstract methods. Can only subclass one abstract class.
      * **Interface –** With Java 7 below: All methods are abstract. And with Java 8: Have non-abstract methods when you using “default” keyword. A class can implement any number of interfaces.
   7. **Interface can have abstract method?**

All methods in Interface are abstract method.

* 1. **Overloading**
     + In the same class, Overloading deals wih multiple methods in the same name but different parameter.
     + Overloading lets you define the same operation in different ways for different data.

* 1. **Override**
     + Overriding deals with two methods, one in the parent class and the other one in the child class and has the same name and same parameter.
     + Overriding lets you define the same operation in different ways for different object types.
  2. **Static Variable vs Instance Variable**
     + **Static Variable** –Define at class level. Class variables are called static variables. When a class is loaded the S variables are initialized.

public Class A{

public static int a = 10;

}

* + - **Instance variables –** Defined at Instance/Object level. It is non-static variables and instance variable initialized when Object for that Class is instantiated

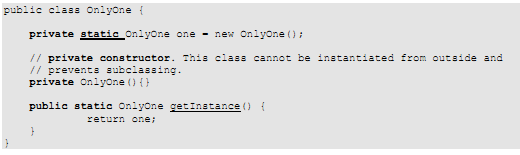
public Class A{

public int a = 10;

public Object b;

}

* 1. **In which case we should use Singleton?**
     + **A singleton** – is a class that can be instantiated only one time in a JVM per class loader. Repeated calls always return the same instance. Ensures that a class as only one instance, and provide a global point of access. It can be an issue if singleton class gets loaded by multiple class loaders or JVMs.



* + - **When to use?** – Use it when only a single instance of an object is required in memory for a single point of access. For example the following situations require a single point of access, which gets invoked from various parts of the code. 
    - Accessing application specific properties through a singleton object, which reads them for the first time from a properties file and subsequent accesses are returned from in-memory objects. Also there could be another piece of code, which periodically synchronizes the in-memory properties when the values get modified in the underlying properties file. This piece of code accesses the in-memory objects through the singleton object (i.e. global point of access).
    - Accessing in-memory object cache or object pool, or non-memory based esource pools like sockets, connections etc through a singleton object (i.e. global point of access).
  1. **What's collection?**

Collections framework provides flexibility, performance, and robustness :

* + - **Polymorphic algorithms** – sorting, shuffling, reversing, binary search etc.
    - **Set algebra** - such as finding subsets, intersections, and unions between objects.
    - **Performance** - collections have much better performance compared to the older Vector and Hashtable classes with the elimination of synchronization overheads.
    - **Thread-safety** - when synchronization is required, wrapper implementations are provided for temporarily synchronizing existing collection objects.
    - **Immutability** - when immutability is required wrapper implementations are provided for making a collection immutable.
    - **Extensibility** - interfaces and abstract classes provide an excellent starting point for adding functionality and features to create specialized object collections.

* 1. **What is the main difference between an ArrayList and a Vector? What is the main difference between HashMap and Hashtable?**
     + **Vector / Hashtable** – Original classes before the introduction of Collections API. Vector & Hashtable are synchronized. Any method that touches their contents is thread-safe.
     + **ArrayList / HashMap** – So if you don’t need a thread safe collection, use the ArrayList or HashMap. Why pay the price of synchronization unnecessarily at the expense of performance degradation.
  2. **String vs StringBuffer?**
     + **String –** is immutable. you can’t modify a string object but can replace it by creating a new instance. Creating a new instance is rather expensive.
     + **StringBuffer / StringBuilder –** is mutable. You can modify the contents. StringBuilder was added in Java 5 and it is identical in all respects to StringBuffer except that it is not synchronized, which makes it slightly faster at the cost of not being thread-safe and offer better performance.
  3. What's different between Resful and SOAP?
  4. What purpose of web services?
     + It might be aimed to reuse the services
  5. What are 3 important concepts in Java?
     + 1. write one, run anywhere
     + 2. OOP
     + 3. Robust and secure.
  6. Different between Public vs Private vs Protected?
  7. Jboss server vs tomcat Server
  8. Multi-threading in java? What kind of list should be used in multi-threading
  9. Method overloading same as polymorphism? =>No
  10. Method overriding same as polymorphism? =>Yes
  11. Can we override a final method?
  12. Can we overload a final method?
  13. 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)
  14. Vector/array list is thread-safe?
  15. Can we add a null object to the array list?
      + Yes
  16. Collection framework: List # Map # Set, can we put NULL into List, Map, Set
      + Hashtable, Queue don’t allow to put null value
      + Other, collection -> yes
  17. **Database connection?** How to get db connection from java class, writhe the syntax
  18. **DataSource?**
  19. **Jdbc?**

1. **J2EE**
   1. **Servlet**

A **servlet** is a Java class. It's written like normal Java.

Servlets provide a component-based, platform-independent method for building Web-based applications

Java Servlet technology provides Web developers with a simple, consistent mechanism for extending the functionality of a Web server and for accessing existing business systems. A servlet can almost be thought of as an applet that runs on the server side--without a face. Java servlets make many Web applications possible.

* 1. **JSP**

A **Java Server Page** (JSP) is a file that is often used in place of a servlet because it makes it easier to output data. It’s running on the server machine which allows you to write template text in (the client side languages like HTML, CSS, JavaScript and so on)

* 1. **MVC Model? Front-Controller?**

**Model View Controller** is a classical design pattern used in applications that needs a clean separation between their business logic and view that represents data.

* + - **Model –**This component manages the information and notify the observers when the information changes. It represents the data when on  which the application operates. The model provides the persistent storage of data, which manipulated by the controller.
    - **View –** The view displays the data and also takes input from user. It renders the model data into a form to display to the user .There can be several view associated with a single model. It is actually representation of model data.
    - **Controller –** The controller handles all request coming from the view or user interface. The data flow to whole application is controlled by controller. It forwarded the request to the appropriate handler. Only the controller is responsible for accessing model and rendering it into various UIs.
  1. **Design Pattern**

+ Singleton

+ Factory

+ Facade

* 1. **Web Service**

+ How to implement web service

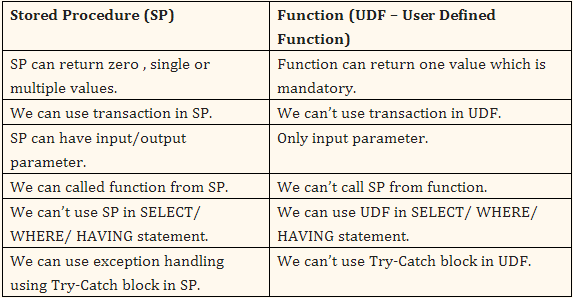
+ Which web service framework are you using to develop WS?

1. **SQL**
   1. What's index?
      * Index is a database structure that provides quick lookup of data in column of table
   2. What's primary Key?
      * Primary key is a single field or combination fields that uniquely define a record. None of the field that are part of the primary key can contain a Null value. A table can have only one primary key
   3. What's Foreign Key?
      * A **foreign key** is a way to enforce referential integrity within your Oracle database. A foreign key means that values in one table must also appear in another table.
      * The referenced table is called the *parent table* while the table with the foreign key is called the *child table*. The foreign key in the child table will generally reference a [primary key](http://www.techonthenet.com/oracle/primary_keys.php) in the parent table.
   4. Unique Key vs Primary Key?

|  |  |  |
| --- | --- | --- |
|  | **PRIMARY KEY** | **UNIQUE KEY** |
| **NULL** | It doesn’t allow Null values. Because of this we refer PRIMARY KEY = UNIQUE KEY + Not Null CONSTRAINT | Allows Null value. But only one Null value. |
| **INDEX** | By default it adds a clustered index | By default it adds a UNIQUE non-clustered index |
| **LIMIT** | A table can have only one PRIMARY KEY Column[s] | A table can have more than one UNIQUE Key Column[s] |

* 1. What's Constraint?
     + A CONSTRAINT clause is an optional part of a [CREATE TABLE statement](http://docs.oracle.com/javadb/10.3.3.0/ref/rrefsqlj24513.html#rrefsqlj24513) or [ALTER TABLE statement](http://docs.oracle.com/javadb/10.3.3.0/ref/rrefsqlj81859.html#rrefsqlj81859). A constraint is a rule to which data must conform
     + A CONSTRAINT can be one of the following:
* a column-level constraint *(NOT NULL, PRIMARY KEY, UNIQUE, FOREIGN KEY, CHECK)*
* a table-level constraint *(PRIMARY KEY, UNIQUE, FOREIGN KEY, CHECK)*
  1. Inner Join vs Left Join



* 1. What's different between procedure and function?d
  2. Performance of EXISTS and NOT EXISTS better than IN, NOT IN
  3. Explain a sub-query? How does a sub-query impact on performance
  4. Different types of contraints in db?

1. **Server**
   1. Which application server are you using?
2. **Introduce Yourself**

Good evening,

My name is Phuong. I work as SE at HvN VietNam.  
I have completed my graduation from Nong Lam University with good certificate.

I have more than two years experiences in development. Especially 1 year experiences with java development.

I have a good knowledge about Spring framework (like Spring MVC, Spring Web-service), JSP/Servlet, HTML, CSS, Javascript, Jquery, MySQL, Oracle and so on.

My domain knowledge is Race Enquiry, Race Management (Phase 1 and Phase 2) with Weatherbys team and Ocado-Pick with Ocado team, etc.

My strength is self confidence, positive attitude and hard work.

Thank you!