1. **What are OOP and OOPS?**

- OOP is short form for Object Oriented Programming;
- OOPS is a short form for Object-Oriented Programming and System.
- It is a programming paradigm with core concept of programming completely on the objects, which are analogous to real world objects. It means everything (thinking and programming) is around objects.

1. **What are fundamental principles or characteristics or properties of OOP?**

- Abstraction
- Encapsulation
- Inheritance
- Polymorphism

1. **What are the benefits or advantages of using OOP?**

- Modularity
- Reusability
- Pluggability

1. **What is an Object? What it contains?**

- It is analogous to real world object where it contains State (or attributes or properties or data) and Behavior (or actions or services) together. The object's state is controlled by its behavior and NOT by other external objects directly.

1. **What is a Class? Why do we need?**

- The Class is a blueprint or prototype or type or categorization for objects. The common state and behavior are put-together as Class.
- It is the basis for creation or instantiation of new objects of that class type.

---

1. **What is an instance?**

- It is merely an object that has been created from a class (a thing created at some point in time or at that instant).

1. **What is a method?**

- It is the behavior/logic encapsulated in the object (as a block of statements), which can do the actions on behalf of the objects. It acts on the data that the object contained.
- Objects communicate via methods.
- It is similar to a function in procedural programming.

1. **What is meant by object state?**

- It is the data of the object. The attributes or data of the object will be added as variables to store the data. The data type or type of the data it contains depends on the class or data types. The methods act on this state or data. For best practice, protection (who can access) of the state is important and should only manipulate by the methods in its object.

1. **What is Interface? Why do we need?**

- An interface is a external view or contractual agreement between class and external world without exposing the internal state & internal behavior. When a class is going to implement the interface, then external objects would rely on the interface for bindings of that class. Without interfaces, managing the external dependencies (making sure bindings are ok) would be hard and more work.

1. **What is Abstraction? Generalization? Concreteness?**

- Abstraction is nothing but the hiding of the details, which helps in reduction of the complexity. It is like analogous to an abstract (summary) of a project report. The abstract classes will be having none or partial implementation. Interfaces can be realized as pure abstract classes.
- Generalization is looking at higher levels and very common characteristics among the objects. It is opposite to the specialization. It is like till 10th grade everyone reads similar courses but after that goes into specialization (into doctor, engineer etc.). Generalization gives the advantage of loosely coupling system and replaceable.
- Concreteness is specialization and has all the implementation details. It is a complete class to create objects.

1. **What is Encapsulation? How can be achieved?**

- Encapsulation is nothing but data hiding or sealing of the object's internal state. It is achieved by making the fields as private and provide appropriate methods for any access or modifications to the external objects.

1. **What is Polymorphism? How can be achieved?**

- Poly means many and morphisms means forms or shapes. It is nothing but invoking same behavior/method name but acts differently based on the different types of input parameters or overriding of behavior in a subclass or during the inheritance.
- Polymorphism is achieved through overloading, overriding or inheritance.

1. **How do you differentiate Object Oriented Programming with Procedural Programming?**

- See the differences between procedural and object-oriented programming.

Procedural Programming	Object-Oriented Programming
The Procedural programming is a programming paradigm that divides the problem into logical modules that are procedures/functions, which are a sequence of steps.	The Object-oriented programming is a programming paradigm that focuses on the abstraction and real world objects.
In this, procedure /functions and data are separated.	In OOP, the data and the methods are bound together as classes/objects.
In this, some level of re-usability and modularity can be achieved.	OO programming follows certain principles: Abstraction, Encapsulation, Inheritance, Polymorphism and benefits more re-usability, modularity and pluggability.
In procedural, it might raise more runtime errors and difficult to maintain the large code.	The OOP is in general to address the issues with procedural programming, also have less runtime errors and easy to maintain the large code.
Example languages: BASIC, C, Pascal etc.	Example languages: Java, C++ etc.

1. **What are the differences between Object-Oriented vs. Object-based programming**

- OOP paradigm has the core principles such as abstraction, encapsulation, inheritance, polymorphism defined to call object-oriented whereas Object-based programming paradigm doesn't have all principles such as inheritance, encapsulation, polymorphism but merely working with objects (instances, constructors, methods, properties).
- OOP have inbuilt real world objects constructed from classes/ blueprints. Object based uses prototyping concepts and adding members to the objects.
- OOP example: Java, C++ languages
- Object-based example: JavaScript language

1. **What is Abstract Class?**

- Any class with partial implementation, in which some of the methods are abstract (i.e., with only declaration but no implementation) and some of the methods could have implementation is called Abstract class.
- Please note that real objects can't be created directly from Abstract classes and can be only be done after creating the concrete class by extending the Abstract class with remaining abstract methods implementation.

1. **What is Inheritance? What are the benefits? What are different types of Inheritance? Describe them.**

- The inheritance is one of the core OOP principles where a class inherits from the parent or multiple parent classes to leverage the behavior/methods and data/properties.
- It could be a single inheritance (inherit from single class) or multiple inheritance (inherit from multiple classes) type.
- Example: Java supports only single inheritance for classes and supports multiple inheritance for interfaces. C++ supports multiple inheritance for classes.
- The current class is called Subclass and the parent call is called Superclass. In the inheritance tree, a child is called as a subclass and parent is called as a superclass.
- The process of inheritance can be viewed as making a generalization/generic object and to bring specialization/specific object.
- The analogy is like going from abstraction to concreteness.
- The inheritance helps in reusing of the existing code.

1. **What is the difference between Abstract class and an Interface?**

- The below are the differences at a high level.

Interfaces	Abstract classes
Interfaces form a contract to the external world/client.	the required behavior by saying what all the objects can do instead of how the object is doing.
Abstract classes provide the abstraction/generalization and thereby form the class/object relationships.	behavior at super/parent classes so that sub/child classes can inherit.
An interface is a complete or full abstraction without any implementation.	
Abstract classes provide a partial implementation (at least one method is not implemented).	Can't instantiate objects
Interfaces must be implemented by Concrete or Abstract classes.	Can't instantiate objects
	Java supports multiple inheritance for interfaces.
	Java supports only single inheritance for abstract classes.

1. **How do you create Objects?**

- Objects can only be created from concrete class constructors.
- Create the objects using “new” keyword in the Java.
- General syntax:
- <ObjectType> object = new <Constructor>
- Examples:
- HelloWorld w = new HelloWorld(“JD”);
- Car c = new Car(“Toyota”, ”Sienna”, 2006);

1. **What is a constructor? What is a destructor?**

- The constructor in OOP is a special method in class, which is invoked automatically during the construction of objects from the class.
- Java class constructors can be declared similar to methods but as a special type of method with name same as the class name and no return type.
- The destructor in OOP, again it is a special type of method/function and used when objects are no longer needed, those objects would be deleted. In Java, there is no destructor to be written, and instead, unreferenced objects would be collected automatically by the garbage collector.

1. **What is default constructor? What is parametric constructor?**

- The default constructor is a no-argument constructor in the class.
- The parametric constructor is an argument constructor in the class.
- In Java, if none of the constructors defined in class, then there will a default constructor provided automatically (from the top level parent class, Object class). If any constructors (default or parametric constructor) defined, then Java doesn't provide any other default constructor.

1. **What is overloading? What are different types?**

- The concept of having the same name but does different actions (or different behavior) is called "Overloading".
- Using below two types or ways, one can achieve the overloading.
- Method overloading

```
public class Calculator {
        public int sum(int x, int y) { }
        public int sum(int x, int y, int z) { }
        public int add(int x) { }
}
```

- Operator overloading

1. **What is overriding? How to do?**

- The concept of having the same method signature (method name and parameters) with different implementations during the inheritance of classes is called Overriding.
- The overriding is achieved through inheritance in the derived or child or subclass.
- In this, superclass's method would be overridden in the subclass with a different implementation (note that it can't be done within the same class).
- Use optional @Override annotation in the subclass's method while overriding the method so that compiler check can happen and can throw an error if Override rule (same method signature) is not happening.

1. **What is data hiding? Why do you need it?**

- Data hiding is the general concept in computer science to hide the data or implementation from the external world.
- The goal is not to leak the implementation details to the user.
- Data hiding is to prevent the unexpected changes and also make changes in future if required.
- Use the data access modifiers to achieve this.
- Data hiding in the OOP is accomplished through Encapsulation.

1. **What is "this" and "super" keywords w.r.t objects?**

- The keyword "this" refers to the current object reference and "super" reference to the super class's object reference.

1. **What is meant by method signature? Does it include return type?**

- The method signature is the method name, parameters of a method.
- It doesn't include the method return type.
- Example:
- In the below method, "main(String [] args)" is called the method signature.
- public void main(String [] args) { ..}

1. **Can an abstract method be private? Why or why not?**

- No. The abstract method can't be private as the abstract class has to implement this method by extending the super abstract class.

1. **What are the class relationships?**

- The class relationships are the weak (other object can exist without current object), strong (other object can't exist without the current object), shared and non-shared types.
- The following are some of the major relationships among the classes.
- Association
- Aggregation
- Composition

1. **What is UML and why do you need?**

- UML stands for Unified Modeling Language.
- It is for documenting the architecture or design of the programs with classes, interfaces, relationships, etc.

1. **What is UML notation for relationships?**

- The below are the simple notation to understand the relations.

Association
Inheritance
Composition
Aggregation
Dependency
Realization/Implementation
Association
Inheritance
Composition
Aggregation
Dependency
Realization/Implementation

← End: Skill#3. Object Oriented Programming & System (OOPS) ←

Good!
Keep going and never give up!!
Please re-read again for more clarity and feel free to contact for help!!!

→ Start: Skill#4. XML →

Skill#4. XML

Most of the existing dynamic content, e-commerce sites and many software configurations have been done using XML file format. It is essential to understand this universal XML file format to survive in the Software dev and QA profession.

1. **What is XML? Is it a standard?**

- XML stands for eXtensible Markup Language. It is derived from SGML family, originally used to do electronic publishing.
- It is a W3C standard.
- XML is a flexible human readable text file to have the data exchange format that is independent of platform, protocol and language.
- Example:

```
<?xml version="1.0" encoding="UTF-8" ?>
<software>
    <titles>
<title id=”12345” vendor=”abc”>ABC1</title>
    </titles>
</software>
```

1. **What are the main objectives of XML?**

- The following are some of key objectives of XML:
- It should be flexible and simple text-based human-readable data exchange format.
- It should support structured data.
- It should be an extensible and self-describing document.
- It should be independent of operating platforms or protocols or computer languages.

1. **What is the structure of XML?**

- XML structure is more of a tree structure with parent/child/siblings relations.
- XML starts with a root tag and contains elements with open tags and corresponding closing tags.
- XML files are used in creating portable documents to transfer among different systems, system/software configuration files, and software builds creation, etc.

1. **How do you define the structure of XML? How do you write XML?**

- The XML structure is based on the DTD (Document Type Definition) or XSD (XML Schema Definition).
- Based on the DTD/XSD, XML is going to be created by following the doctype, elements, and the order of those elements in the document.

1. **What are the differences between XML and HTML?**

- XML is mainly for data exchange whereas HTML is mainly for data viewing or display.
- XML has no built in tags whereas HTML have built-in tags (such as html,header,body,h1,div,center etc.) with specific purpose/meaning.
- XML should have closed empty tag whereas HTML can skip.
- XML is processed by backend code whereas HTML is processed by front-end tools like browser and javascript.

1. **What is meant by well-formed XML?**

- Well-formed XML is an XML, which meets the following conditions for correct syntax.
- Having only one root tag.
- All open tags must have to be closed.

1. **What is meant by validation of XML?**

- XML validation is a check for the following and is valid XML if and only if (iff).
- Well-formedness of XML
- Semantics matching against corresponding the DTD or XSD.

1. **What is DTD? What is XML Schema or XSD? What are the differences?**

- DTD stands for Document Type Definition.
- XSD stands for XML Schema Definition.
- Both DTD and XSD are for defining the structure definition to write or process the XML files. These metadata definition files define the list of allowed elements & structure in the XML document.
- DTD or XSD is required only if the entities need to agree for communication otherwise simple XML is ok without DTD/XSD (like for testing or experimenting).
- XSD is newer than DTD for defining the structure definition for actual XML file to follow with much more powerful than DTD.

1. **What are some of the popular techniques to parse XML files?**

- There are three well-known parsing techniques.

i) DOM parser

- Entire XMl doc is being loaded into memory.
- Traverse the Nodes and NodeList.

ii) SAX parser

- It is different than DOM parser and is based on events.
- Not all the XML document is loaded into the memory but instead read line by line based on events triggered like open tag, closed tag, characters, comments, etc.

iii) StAX parser

- StAX stands for Streaming API for XML.
- SAX parser pushes the data based on events whereas StAX parser pulls the data by maintaining a cursor position instead of events generation.

1. **What are XML declaration and processing instruction in XML?**

- The XML declaration is as below with first line having version and other details between <?xml and ?> tags. Note that this is not a processing instruction and instead to indicate as XML.

<?xml version="1.0" encoding="UTF-8" ?>

- The processing instruction is enclosed between <? and ?> and optional target like guide the instructions for application.
- The PI can be anywhere in the document.
- General processing instructions

<?PITarget PIContent?>

- See more details on processing instructions at http://www.xmlplease.com/xml/xmlname/pi#s6

1. **What is namespace in XML?**

- XML namespace (NS) is mainly to avoid the conflicts on the names/tags in the XML documents.
- It is to have the same names from different URLs but different definitions without collision among those names.
- This can be achieved by prefixing the namespace to the name. You can see with xmlns:<prefix>="URL" in the root element attribute.
- Default namespace will not have the prefix.
- Example:

```
<root xmlns:m="URL1" xmlns:n="URL2">
        <m:tag1></m:tag1>
        <n:tag1></n:tag1>
</root>
```

1. **What are real applications of XML?**

- XML is being widely used since the evolution of the internet. Some of the applications include -
  - XML files as portable documents
  - XML configuration or metadata for tools/servers
  - XML files as exchange of data/catalog in the automated processing
  - XML also the basis for Universal Business Language (UBL) and electronic publishing.
  - ANT, Maven build files are XML files.
  - XML files can be used places where custom structured data needs to be defined.

← End: Skill#4. XML ←

Good!
Keep going and never give up!!
Please re-read again for more clarity and feel free to contact for help!!!

→ Start: Skill#5. XPath →

1. What is XPath? Why does it matter?

- XPath is an expression language for addressing/defining/extracting parts of the XML document.
- XPath is used to identify or select and extract the parts.
- XPath uses path expressions/predicates to navigate in XML and contain libraries for standard functions.
  - There are more than 100 built-in functions to handle strings/ numeric values, date and time comparison, node manipulation, boolean values, etc.
- XPath is mainly designed to be used by XSLT and XPointer. It is based on the tree representation of XML. It helps in transforming or extracting the data from XML/HTML.
- There are many XPath implementations in Java, C#, Python, JavaScript, etc.

1. **What are different node types in XPath?**

- The following are the different types of nodes:
- Element
- Text
- Attribute
- Namespace
- Preprocessing-instruction
- Comment
- Document

1. **What are different node relations?**

- Below are different node relations:
- Root
- Parent
- Children
- Siblings
- Ancestors
- Descendants

1. **What is a predicate? What is its purpose?**

- The predicate is a condition, and its purpose is to filter nodes from the list of nodes.
- Predicate is embedded in the brackets [].

1. **What is the XPath for selecting the first node? And also last but one node? Or at a particular position?**

- The below are the XPath expressions with different inbuilt functions.
- //nodename[1] : selects first nodename of nodenames
- //nodename[last()] : selects last nodename
- //nodename[last()-1] : selects last but one nodename
- //nodename[position()<3] : selects first 2 nodenames

1. **What is the XPath for selecting nodes with expected attribute value?**

- To select matching nodename elements with attributename's value, use the following expression.
- //nodename[@attributename=value]

1. **What is the xPath wild characters to match any node and nodes with any attribute?**

- Use "*" wild character for matching any node(s).
- Example://book[@*] : all elements that have any attribute
- //*[@attributename=value] : to select all elements to match attribute-name matching value.

1. **How to select several XPaths in a single selection/expression?**

- Use "|" operator to combine several XPath expressions.
- Example: To select all title and author nodes, use //title | //author

1.What is axis? Why do you need? Please provide few expressions with axes.

- Axis is the name that defines a node-set relative to current node using relationships. It is useful to reference nodes without going through root node and instead with a relative in reference to the current node.
- General syntax is Axis::nodename[predicate]
- Different Axes:
- ancestor : selects all ancestors in the node hierarchy of the tree.
- ancestor-or-self : selects ancestors and the current node itself
- attribute : selects all attributes of the current node
- child : select all children of the current node
- descendant : selects all descendants of the current node
- descendant-or-self : selects all descendants of the current node and current node itself
- following: selects everything after the current node
- following-sibling : selects all siblings after the current node
- namespace : selects all namespace nodes of the current node
- parent : selects the parent of the current node
- preceding : selects all nodes the current node
- preceding-sibling : selects all siblings before the current node
- self : selects current node
- Examples:
- child::store → selects all store nodes that are children of the current node
- attribute::value → selects value attribute of the current node
- child::text() → selects text node children of current node
- child::node() → selects children nodes of current node
- Examples:
- child::store → selects all store nodes that are children of current node
- attribute::value → selects value attribute of current node
- child::text() → selects text node children of current node
- child::node() → selects children nodes of current node

1. **What are different XPath operators?**

- Below are the operators. These are semantically similar to other programming operators.
- Arithmetic operators: +, -, div, *, mod
- Conditional operators: =, !=, <, >, >=,<=
- Logical operators: or, and

1. **What is JSON? How does JSON format look like (snippet)?**

- JSON stands for Java Script Object Notation.
- JSON is a lightweight data-interchange text format.
- JSON is a subset of JavaScript (JS) programming language.
- It is language independent, and many high-level languages have libraries to create and parse the format.
- The JSON format is human readable, faster and easier than XML.
- There are no reserved words/keywords and support structure/ hierarchical (values within values).
- The JSON file is a collection of JSON objects and the object contains name/value pairs. The typical extension is ".json".
- The JSON object is {"name":"value",..} and ordered list of values arrays like [value,..]. See more at json.org. See more details at json.org.
- A JSON file format example is as below.

```
{ "employees": [
{ "firstName":"Michael" , "lastName":"J" , "address":""},
{ "firstName":"Jagadesh" , "lastName":"Munta", "address":"Fremont, CA" }
]
}
```

1. **Is JSON a programming language? Or a protocol? Or a message format? Is it language dependent?**

- No. JSON itself is not a programming language nor a protocol. It is simply a data or message format.
- The JSON files are independent of the high-level programming languages. These JSON files can be generated and parsed by many programming languages.

1. **Where do you use JSON?**

- JSON format can be used in any application where data serialization and deserialization along with some structure is required.
- JSON is extensively used in RESTful Web Services/REST APIs for the data embedding in requests and responses.

1. **What is the HTTP header used for passing the JSON data?**

- In the HTTP, below header and value is used to indicate that JSON content is available in requests or responses. Set this header while sending the HTTP request or response.

Content-Type: application/json

1. **How do you process (parse, transform, generate, query) the JSON text?**

- The parsing of the JSON text can be done using APIs provided by different languages.
- In JavaScript itself, JSON.parse(text) can be used to get the JS object and then simply reference the names with dotted notation and arrays.
- In Java, there are many APIs available. Some of the free and popular APIs and references in use are as below.
- JSON-P (Java API for JSON Processing) - https://json-processing-spec.-va.net/
- Jackson Library - https://github.com/FasterXML/jack-databind/databind/
- GSON (Google's) library - https://github.com/google/gson
- Overall, the Jackson library is very simple and highly performed APIs to use for JSON processing.

Skill#7. Source Code Control System (SCCS)/ SCM - SVN & GIT

The source code is one of the major intellectual property of any software organization. It is essential to collaborate with the developing teams using a source code control system such as SVN or GIT. It is important to understand the SCM concepts and working knowledge on SVN and recently popular GIT software for both developers and QA automation engineers.

1. **What is source code control system (SCCS) and SCM? Why do you need?**

- The SCCS stands for Source Code Control Software and SCM stands for Source Code Management. In a practical sense, both are referring the same and SCM is more widely used.
- The main purpose of SCM is to have the source code versioning, collaboration, and sharing with others local or geographically dispersed teams or open source.

---

1. **Describe the key elements in SCM such as repository, workspace, versioning?**

- Repository is the storage that is being used to store the project source files/code. Also termed as source code repository. The repository (repo) is typically referred to URL using HTTP or client specific protocol.
- Workspace is the tree structure of the project source files organized to work with various build systems like ANT/Maven.
- Versioning is the change management to keep track of all prior changes.

1. **What do you store in the repository?**

- The project source files will be saved in the SCM repository and typically the text files like .java, .html, .properties, .xml, etc. files and NOT any binaries or generated code or other generated artifacts. This way the versioned sources can be downloaded quickly and also saves the space. Besides, losing binaries is not a big deal as such, unless only third party binaries are only available and can be stored.

---

1. **What is source code? Why is it important? What happens if you keep the code at individual machines?**

- The source code is the set of program files and related files developed by the developers or QA teams to program for the desired requirements. These are the text files and typically read and modified by more than one engineer as a team.
- The source code is important to preserve because it is the actual IP (Intellectual Property) of any software project.

1. **What are the primary functions of SCCS software?**

- The standard functions of SCCS software include the following.
- Atomic operations like commits to be consistent state
- Concurrent access and file locking
- Versioning and merging
- Branching, labeling, tagging, and the snapshot of the source trees.

1. **What are different popular SCCS software available?**

- Below are some of the popular free/open source software:
- CVS - Concurrent Versioning System (oldest)
- SVN - Subversion
- Mercurial
- Git (Latest and distributed architecture)
- Below are some of the popular commercial software:
- Visual SourceSafe(Microsoft)
- ClearCase (IBM)
- Perforce

See more at https://en.wikipedia.org/wiki/List_of_version_control_software

1. **What is meant by the checkout, check-in or commit, merge, delete, log, status?**

- Checkout is the operation to retrieve (or) pull sources from the repository to the local directory, called workspace (or working copy).
- Check-in is the operation to store or commit the sources or changes from local working workspace to the files in the repository.
- Merge is the operation to apply two sets of changes on a single file or multiple files.
- Delete is the operation to remove the sources from repository.
- Log is the operation to show the history of changes on a single file or multiple files.
- Status is the operation to determine commit status of the local working copy of files w.r.t the files in the repository.

1. **What is meant by the trunk or mainline? Branching? And tagging?**

- The trunk or mainline is the main trunk of code tree path in the repository that is not a separate branch. It is also called master or tip. From this trunk or master, the branches can be built so that parallel code paths can exist.
- Branch is a forked tree path that is to build a different code path. This way, it is easy to develop and release the product independent of other branches including mainline or master in parallel with the same set of files.
- Typically, before stabilizing of a product release, new branches would be created and allowed only changes required for that release. In Git, the common practice is to create branches for their module code and later merge to the master or other top level branches.
- Tagging is labeling of master or a branch at that time to retrieve the versioned files at that point. It is like taking a snapshot.
- Please note that one can only check-in the files in a branch or mainline but not in a tagged label itself.

1. **What is meant by merging conflicts? What happens if you don't resolve the conflicts? How do you resolve the conflicts?**

- Merging conflict is a checkin error raised when the local workspace is not in sync with remote repo before applying the changes and the same lines of the files got changed in the remote repo. In this state, the tool can't reconcile the changes.
- If conflicts can't be resolved, then one can't commit the changes directly. The client will not allow the files check-in to the remote repository.
- In such cases, before check-in, one has to resolve by selecting in-favor-of one change (either local or remote) than the other change. Otherwise, manually merge each change in those files to get the unified changes and then override in the repo.
- Merge conflicting resolution: there are 3 options
- manual resolution
- first changes replace the second changes
- second changes replace with first changes

1. **Can you describe SVN? What are the supported protocols?**

- SVN stands for Subversion. It is one of the free/open source SCM used by enterprises. It is kind of older but still being used in bigger companies.
- SVN supported protocols are svn:// and http:// or https://.

1. **What kind of architecture/design used in SVN? Any other architectures?**

- SVN is based on client/server technology with centralized server architecture.
- The other alternative is distributed protocol with a decentralized architecture where developers have their own development or work repositories and only shared when published. Example: Git.

1. **What do you need for interacting with SVN server?**

- One should have repository URL with svn or http/https protocol and svn client or browser.

1. **What is the svn command for checkout workspace?**

- To checkout the existing source code from SVN repository, use svn client software (to be installed before using on your laptop/desktop) and then project URL.
- The general svn command line syntax is below.
  svn <cmd> <repourl>
- Use the below command style where host and port refer to the SVN hostname/IP and port number; <repo> is the repository name; trunk or branch name;<projectdir> refers to project directory.

svn co http://host:port/<repo>/<trunk-or-branch>/<projectdir>

1. **What are the svn commands to do a fresh checkout, check-in or commit all files in the current directory?**

- To do fresh checkout the workspace, use similar to the following command.

svn co http://host:port/<repo>/<trunk or branch>/<projectdir>

- To add the files or directory:

  svn add <file/dir>

- To commit the files/directories:

  svn commit <files/dir>

  Example: *svn commit .* (this will commit all the files in the current dir)

1. **What is the command to see the list of files modified locally?**

- Use the below to check the status:

svn status <file/dir>

---

1. **What is the svn command to see the history of changes done on a particular file?**

- *svn log <file/dir>*

1. **What is the svn command to see the differences between the locally changed file vs. file in the repository?**

- The command to see the differences to the remote revision is as below.

svn diff –r <revision> <file>

- For more help on commands, reference the free online book at http://svnbook.red-bean.com/en/1.7/svn-book.pdf

1. **Can multiple users checkout and check-in in parallel? Is there any lock?**

- Yes. Multiple users can do the checkout and check-in without any issue in parallel, but if there is a change on the same file at the same time, internally SCM software maintain the locking mechanism.

1. **What are the key features that you need for any version control systems like Git?**

- A typical sequence of operations with any SCM is as follows.

i) Checkout a project from repository
ii) Create new files or subdirectories or edit files
iii) Update from repository
iv) Check-in or commit
v) Go to or repeat from above step 2
vi) Check status
vii) Push to master in case of Git or distributed SCMs.

1. **What are the typical steps to work with Git?**

- To create a new git project or start a new repository for an existing project, use below 3 steps

git init --> which creates .git folder

git add .

git commit .

- First time to get the code from remote repository/git server

1. *git clone https://github.com/jagadeshmunta/myproject1.git*

- Add files to the workspace

2. *git add .* ---> adds all files and directories in the current directory

- Check status w.r.t remote repository

3. *git status*

(shows untracked/to be committed files)

- Update the workspace from a remote repository

4. *git pull*

- Commit (to the local workspace)

5. *git commit -m " message" <files/dir>*

- Send the files to remote repository.

6. *git push origin master (or) git push origin branchname*

- Repeat #2 to #5 for further modifications.
- Operate with branches with below commands.
- git branch --> lists the branch names and * for the current branch
- git branch <branchname> --> if no branch exists, it will create a new branch
- git checkout <branchname> --> switch the branch to given branchname
- Use below commands to do modifications in the workspace:
- --Checkin--
- git add .
- git commit -m "message" <files/dir>

- git push origin <branchname>
- Merging to master
- git checkout master
- git pull
- git merge <branchname> (NOTE: if conflicts occur, then resolve manually and do the commit - git commit -m "")
- git push origin master --> push the changes to the remote master

← End: Skill#7. Source Code Control System(SCCS)/SCM - SVN & Git ←

Good!

Keep going and never give up!!

Please re-read again for more clarity and feel free to contact for help!!!

→ Start: Skill#8. Unix/Linux OS →

1. **What is an Operating System (OS)? Brief on its typical functions.**

- The Operating system is the software that manages or controls the underlying hardware or machine and other software resources. Most of the computing devices are programmable with the help of OS.
- Typical functions of OS are:
- Process Management
- Memory management
- I/O management
- File Management
- Network management
- Applications Management
- User management

1. **What are the different OS architecture types?**

- OS architecture is of broadly 2 types.
- Monolithic OS and is a single bundle that does all the management.
- Kernel/Micro OS is the layered operating system (like onion) with a small footprint of centered nucleus or kernel that does some core OS to machine interaction functions. In this, the kernel is going to separate the other top level activities and user applications. Because of this kernel style, the program that effects can only cause that process go down/hang and can not impact/crash entire OS.

1. **What is meant by a Platform?**

- Platform means Hardware + Operating System and sometimes referred with OS only or HW only.
- Examples: Intel x86 Platform, Apple Mac Platform, Windows platform, Linux platform, Solaris Sparc platform, IBM mainframe, HP Hardware etc.

1. **What is Unix/Linux and how is it different from Windows?**

- The Unix and Linux are the multi-user, multi-task, kernel based operating systems (OS) widely used on server side hosting of enterprise applications and websites. Typically, users would do remote login to these systems. These OSes support command line terminal mode and GUI also.
- Unix OSes are mostly commercial. Some of the examples are Sun Solaris, HP-UX, IBM AIX, Mac OS, etc. that are tied to hardware platforms.
- Linux OSes are typically free and open source. Some flavors of Linux OSes are commercial for support and having more security. Example Linux OS are CentOS, Ubuntu, RedHat Enterprise Linux (RHEL), Oracle Linux, SuSE Linux, etc.
- The Windows operating system is a single user desktop-based operating system where the user is going to interact with machine physically. Windows is widely used and easy to use GUI OS, which is commercial from Microsoft.
- Windows platforms are typically used for client-side computing whereas Linux used for server side computing. Some of the latest Windows versions are Windows 7, Windows 8 and Windows 10.

1. **How do you see Linux and windows differences at usage level?**

- Windows --> It is case insensitive (lower/upper case treated as same)
- Path separator is ; (semi-colon)
- File separator is \ (back slash)
- Prompt : >
- super user: Administrator account
- Linux --> It is case sensitive (lower and upper cases are different)
- Path separator is : (colon)
- File separator is / (forward slash)
- Prompt: $
- Super user: root (#)
- Everything is a file

1. **What are different parts of linux OS?**

- Linux kernel + Utilities and libraries + Shell
- Shell is the user interaction program in the unix/linux OS. Below are the popular shell programs:
- sh (Bourne Shell)
- bash (Bourne Again Shell)
- csh (C Shell)
- ksh (Korn Shell)
- tcsh (Turbo C Shell)
- Selecting a particular shell is more of preference with regular usage. Recently bash is popular.
- Comparison of various shell programs can be found at https://en.wikipedia.org/wiki/Comparison_of_command_shells

1. **What are different parts of linux OS?**

- Linux kernel + Utilities and libraries + Shell
- Shell is the user interaction program in the unix/linux OS. Below are the popular shell programs:
- sh (Bourne Shell)
- bash (Bourne Again Shell)
- csh (C Shell)
- ksh (Korn Shell)
- tcsh (Turbo C Shell)
- Selecting a particular shell is more of preference with regular usage. Recently bash is popular.
- Comparison of various shell programs can be found at https://en.wikipedia.org/wiki/Comparison_of_command_shells

1. **How do you connect to a remote unix/linux machine?**

- To log in to a remote host, one can use the below protocols or client/server programs:
- ssh (secured shell) is to make a secured connection to the server using ssh protocol and is widely used & recommended.
- telnet and rlogin are less secured. These are not being used lately because of security concerns.
- VNC (Virtual Network Connection) software is to connect to a remote server GUI.
- On linux, use the ssh program.
- It uses ssh protocol
- Credentials as one or the other as below:
- VNC for GUI connection to the remote server
- On windows, use the SSH client such as putty.exe and can be download from http://www.putty.org/. Also, VNC can be used for GUI access on remote linux/unix server.
- From mac or another linux, use the following way and enter the username and password at the prompt.
- Usage: ssh <hostname or IP address>
- VNC can be used to connect to the remote server VNC for GUI access.

1. **How do you search given string or word in a file?**

- Use grep or egrep command.
- Case sensitive
- Case in-sensitive (use -i option)
- Enclose the word in quotes “” or ‘ ‘ if spaces are there.

1. **How do you run a command or process in the background? How do you bring it to the foreground?**

- To start a command in the background, enter that command and then append with an ampersand (&).
- Type “bg” command in the same terminal to send it back to the background.
- Type “fg” command in the same terminal to bring it to the foreground.
- Example:
- *cp -r dir1 /home/jmunta &*
- *fg*
- *bg*

1. **What is the command to find running processes and process ids (PID)?**

- Use "ps" command.
- Example: *ps -ef* (NOTE: prints all running processes including command file name)

1. **What is the command to kill a particular process id?**

- Use "kill" command to shutdown/stop a running process. Also, apply -9 option if process does not respond (to normal kill command) and to do forcefully.
- Example: *kill -9 <pid>* (replace <pid> with actual pid shown in the ps command)

1. **What is the command to list the hidden files?**

- Use "ls" command with -a option to get all the files including hidden ones.
- Example: *ls -a*

1.What is meant by file permissions? How to set the permissions? What is meant by 644?

- File permissions are the attributes for files to authorize read, write, execute by different users, groups or all public. Also note that in Linux/Unix, everything is based on files.

First, have an understanding of the binary numbers, conversion to and from decimal to binary. Binary means 2 and actual values are either 0 or 1 only.

Example: decimal 5 as binary and 8 as binary.
5 --> 0101 (binary form)
5/2 -- 1 (remainder)
2/2 -- 0
1 -- 1

5--> 101
8 --> 8/2 --> 0
4/2 --> 0
2/2 --> 0
1/2 --> 1
1000 (binary form)

When you do long listing of the files (ls -l command), then you would see the permissions in the first column.
The symbols rwx stands for read write execute permission. 1 means permission is there (represented as 1) and - means not there (0).

1 1 1
- : 0

The permissions for a file is represented as 3 octal fields as below.
(owner) (group) (others/world)
rwx rwx rwx

111 : is 7 (as decimal)

rw- : 110 --> 6

1x2^2 + 1x2^1 + 0x2^0 --> 4+2+0 = 6

rwx------
111 000 000
7 0 0
700 --> permission

currently 775 (111 111 101)

b) To change the permissions, use chmod command.
Example: chmod 700 <file/dirname> (or) chmod a+x <file/dirname>
Examples to see the final octal value for the permission to use in the chmod.
111 001 001 : 711
100 000 000 : 400
110 000 000 : 600

c) 6 4 4 : permission means that owner can read and write, group and world can only read. 110 100 100

1. **Why do you get the error 'command not found' while executing a command/file (with right directory path)? How to make it work?**

- When the command program file is not in the PATH environment variable or if the command file doesn't have the executable permissions for the current logged in user (or group it belongs to), then 'command not found' error would be thrown by the shell.
- To fix the above problem, make sure it is set in the PATH or enter full path or make sure executable permissions given to the users/groups/own-ers.
- Example:

chmod a+x <command/filepath>

export PATH=${PATH}:<command/filepath> where <command/filepath> is the actual path of the program.

1. **What is the command to see the contents of a file? How to pause after each page filled?**

- Use "cat file" command to display all contents of the file.
- Use "more file" command to view the file page by page and click enter button to display next line or click spacebar button to display next page.

---

1. **What is the command to get first 10 lines of a file? And getting first 5 lines?**

- Use "head file" command to show first 10 (default case) lines from the file.
- Use "head -5 file" command to show first 5 lines from the file.

1. **What is the command to get last 10 lines of a file? And getting last 5 lines?**

- Use "tail file" command to get last 10 (default case) lines of file.
- Use "tail -5 file" command to get the last 5 lines of file.

1. **What is the command to monitor a running log file (such as a tomcat server.log or other file where content is getting modified all the time)?**

- Use "tail -f file" command to monitor (display contents as it gets updated) a running/continuously updated file.
- Example: tail -f server.log

1. **What permission to be set to a directory so that 'cd dir' command would work?**

- Set executable permission to the directory along with other permission or as below to give executable permission to everyone.

  chmod a+x dir

1. **What is meant by pipeline (|)? Where do you use?**

- The pipe or pipeline is to pass the output of a command to the next command as input.
- It is used to get the chained actions together in a single command.
- Example: to get the count of the java files in the current directory.
  *ls *.java |wc -l*

1. **What is meant by symbolic link and the command to create a symbolic link to a file or directory?**

- The symbolic link means creating a new file name referencing to another file or inode (i.e., underlying structure pointer of a file) of a file. If the link file pointed to a filename, then it is called soft link else if it is pointed to the inode, then it is called hard link. The hard link will follow the name (even if it is moved to a different file) whereas soft link will not move if the original file moved to a different file.
- Use "*ln -s frompath topath*" to create a soft link as topath for frompath file. That means topath is simply a reference to the frompath and both refer to the same content.
- Use "ln frompath topath" to create the hard link as topath for frompath file.

1. **How do you find given file and directory sizes?**

- Use "du" command to find the size. The below will give the size in MB/GB.

du -sh fileordir

1. **How do you find a file in a given directory?**

- Use "find" command to find the file in a directory.
- The below command is used to find all java files under the dirpath.
- *find dirpath -name "*.java"*

1.What is 'vi' editor? Describe on how to edit files (insert mode, save, copy lines, paste lines, delete lines, etc.)?

- The vi is default editor on unix/linux and stands for "visual instrument".
- Type vi at the shell prompt and follow below shortcut keys to edit the files.

Press Esc key to enter into escape mode to activate the shortcut keys.
press i --> for insert (before inserting any characters)
press o --> Inserts a new line below your current line
press O --> Inserts a new line above your current line
press : --> enter into command
w --> save and be there in the editor
q --> quit
wq --> save and exit
x --> save and exit
press <n>yy --> Yanking of n lines (copy of n lines into buffer)
p --> print below the cursor line
<n>p --> prints n lines below cursor line
u --> undo
dd --> delete current line
<n>dd --> delete n number of lines (it is a like cut operation and you can paste like above p command)

A --> insert at the end of line
I --> insert at beginning of line
x --> delete a single character at cursor position
<n>x -> delete n number of characters from your cursor position
w --> word by word move the cursor
dw -> delete a word at cursor position
<n>dw --> delete n words starting at cursor position
/<word> --> search for word
n --> next in the search
N --> reverse in search
:set number --> display line numbers

ctrl + g --> display the current position
:$ -> go to bottom line
:1 -> go to first line
:<n> --> go to nth line
:1,$:s/Hello/Hi/g : replace Hello with Hi entirely in the file
r : mark for replace and press whatever letter you want

1. **What is an alias? How do you set an alias and how to remove alias?**

- The alias is setting a name for a command. The main purpose is to have a short name for a long command or with different options command.
- To set alias, use “alias <aliasname> <command>” command.
- To unset an existing alias, use “unalias <aliasname>”
- Examples:
- alias l “ls -lrt” → after this command execution, running just l as the command would give the long list with reverse in timestamp.
- unalias l after this command execution, l command would not found.

1. **How do you count lines, words, and bytes in a file?**

- Use “wc” command would give the number of lines, words, and bytes.
- Usage: wc filename
- Example:

$ wc test2.txt

34 240 1429 test2.txt

(#lines #words #bytes)

1. **What is the command to check which user being used in the prompt?**

- Use “id” command. Run “id” to check the username.

1. **What is a 'touch' command do? Where do you use?**

- Usage "touch file" command is to update the timestamp of an existing file to current time or to create a new file if the file doesn't exist.
- It is useful for creating an empty markup file for some events to track.

1. **What is the file separator character and path separator character in unix/ linux?**

- In Unix/Linux,
- path separator is : (colon)
- file separator is / (forward slash)

1. **What is the kernel? What is the command to see the kernel version?**

- Kernel is the nucleus of the unix/linux, which is set of core programs that interact with the hardware and other software resources.
- The "uname -a" command would give the kernel version along with other details.
- Example:

$ *uname -a*

Linux jdcloudapps 3.2.0-24-virtual #37-Ubuntu SMP Wed Apr 25 12:51:49 UTC 2012 i686 i686 i386 GNU/Linux

1. **How to find the OS release version?**

- One way is to check the OS version using the files /etc/*release*. See below.

```
$ cat /etc/*release*
DISTRIB_ID=Ubuntu
DISTRIB_RELEASE=12.04
DISTRIB_CODENAME=precise
DISTRIB_DESCRIPTION="Ubuntu 12.04.3 LTS"
NAME="Ubuntu"
VERSION="12.04.3 LTS, Precise Pangolin"
ID=ubuntu
ID_LIKE=debian
PRETTY_NAME="Ubuntu precise (12.04.3 LTS)"
VERSION_ID="12.04"
```

1. **How do you see the system details like total memory(RAM), processors and monitor the processes?**

- Use "*cat /proc/meminfo*" command to get the memory information.
- Use "*cat /proc/cpuinfo*" command to get the cpu/processors information.
- Use "top" command to monitor/refresh the memory, running processes, cpu utilization, etc. on the console until ctrl+c pressed.

1. **How do you redirect the output of a command to a file (such as log file)?**

- To redirect output to a file.
- Use "*tee*" command to write output to a file while displaying on the console.
- Use "> filename" to write to a file (if file exists, then its contents would be overwritten).

1. **How do you append text to a file without overwriting the existing content?**

- Use ">> filename" to append output to an existing file.
- Usage: command >> filename
- Example: ls -l >> files.txt

1. **What is the command to pause the screen scrolling temporarily and resume again?**

- Process ctrl+s to pause scrolling on the shell console screen.
- Process ctrl+q to resume scrolling on the shell console screen.

1. **How do you suspend a running process? And then send to the background or bring to the foreground?**

- First, process ctrl+z to suspend a running process on the shell console.
- Then type "bg" command to send the suspended process back to the background.
- Later type "fg" to bring back the background process to the foreground.

1. **What is a 'tar' command? How to use it and how to untar a file?**

- The "tar" stands for "Tape Archive" and is used to package a set of files as a single file with .tar extension.
- The below tar command with option "-c" is to create a new tar with all files in dir. NOTE: If the extraction should not have the dir name, then go to that directory and do the tar on the current directory.

tar -cvf filename.tar dir/

- The below tar command is to peek through for list of files/dir in that tar file.

tar -tvf filename.tar

- The below tar command is to extract/untar the files/dir from tar file.

tar -xvf filename.tar

1. **Write a single command to count all java files in a given directory and have a word 'test'.**

- The below is the required command with subcommand and pipe.

grep test `find dirpath -name "*.java"`|wc -l

1. **Write a command to get the 3rd word (separated by comma) of a string in a variable.**

- Use "cut" command. Below is the syntax:

cut –f<wordnumber> -d'<separator>'

- To get the 3rd word with comma separation, use the below example (mainly cut –f3 –d','). See the 3rd word "result" in the output.

```
$ export VALUES=val1,val2,result,comments
$ echo $VALUES|cut -f3 -d','
result
$
```

1. **What is a sudo command? Why do you need?**

- The sudo command is used to run a command with another user than the currently logged-in user.
- Typically, sudo is used to run commands that need more privileges (root privileges) than the current user (say jmunta). From the security point of view, this is the best practice. For example, run a command such as packages installation without login as root user or without getting to root user with su command. Depending on the configuration, the root privileges will be used for administrative commands/files with sudo access.
- Typically, current user's password should be entered during the sudo command run. Without sudo command, one has to login with root or su (super user), which might be a bad practice due to potential security hacks.
- Examples:

ssh jmunta@host

$ sudo vi /etc/hosts to edit the root only accessible file with current user.
$ sudo rpm –iv xxx.rpm to install rpm packages with current user.
$ sudo -u jagadeshbmunta ls –jagadeshbmunta to list the files of a different user than current user.
$ sudo su to enter into root.
#

1. **What is a ping command and how it helps in checking the host reachability?**

- The ping command is a basic utility to check if a network interface works locally or a connection works between the current machine to another machine. It is used to check if a machine is reachable and alive over the network.
- The ping command is to send packets to and receive from another machine over ICMP protocol and also determines how much time it takes. If the machine is not reachable, then ping will time out or else give the time taken to receive back the response. Note that if ICMP is blocked, then ping doesn't work.
- General syntax and examples are as below:

ping <ip or hostname> press ctrl+c after few responses because it is continuous check and provide statistics after ctrl+c.

ping <ip or hostname> -c 1 for 1 time check and provide statistics at the end.

~ jagadeshmunta$ ping localhost
PING localhost (127.0.0.1): 56 data bytes
64 bytes from 127.0.0.1: icmp_seq=0 ttl=64 time=0.042 ms
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.063 ms
...
64 bytes from 127.0.0.1: icmp_seq=9 ttl=64 time=0.073 ms
^C
--- localhost ping statistics ---
10 packets transmitted, 10 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 0.042/0.056/0.073/0.010 ms
~ jagadeshmunta$ ping localhost -c 1
PING localhost (127.0.0.1): 56 data bytes
64 bytes from 127.0.0.1: icmp_seq=0 ttl=64 time=0.061 ms

--- localhost ping statistics ---
1 packets transmitted, 1 packets received, 0.0% packet loss

- *round-trip min/avg/max/stddev = 0.061/0.061/0.061/0.000 ms*

1. **What is curl command and why do you need?**

- The curl is a simple command line utility to retrieve the contents of a given URL. There is no need for browser to get the Web content because the browser requires graphical terminal whereas curl can work on non-graphical terminal or at command prompt.
- The curl can be used to quickly check URL responses, download files, and also automating them through shell scripts. Recently, the curl is being used heavily to test the REST APIs/endpoints in a quick way.
- The curl supports different protocols such as http/https/ftp/ftps, etc. and have lot of options. Run "*man curl*" command to see more details.
- The below are some of the frequently used options, especially while working with HTTP/REST endpoints.
- -o *filename* Use this option to store the response or output in this given filename.
- -X *requestmethod* Use this option to specify the HTTP URL request method such as GET/POST/PUT/DELETE etc., otherwise it will be GET by default.
- -H "*headername: value*" (or) --header Use this option to specify the required HTTP headers in the request. Supply multiple –H options to send multiple headers.
- -u "*username:password*" Use this option to supply the authentication credentials such as username and password in the URL request.
- -d *@filename (or) --data @filename* (or) –d *name@filename* Use this option to send the data in the file (after URL encoding) to the HTTP POST request. Also, multiple –d options can be specified. This causes the header to include as *application/x-www-form-urlen-coded*.
- -F "*fieldname=value*" (or) --form "*fieldname=value;content-type*" (or) –F "name=@file" Use this option to supply the HTTP form data POST to the URL. Multiple field name/values can be specified by multiple –F options.

- -k or --insecure Use this option to ignore the certificate check and do non-secure https request.
- Some of the examples to understand the usage:
- curl http://www.mycompanyinc.com/files/file1.zip
- curl –o file.zip http://www.mycompanyinc.com/files/file1.zip
- curl –X GET –u "username:password" –o file2.zip http://www.mycompanyinc.com/securefiles/file2.zip
- curl –X POST –u "username:password" -H "Content-Type: application/json" –d @prod.json http://www.mycompanyinc.com/api/v1/prod-ucts
- curl –X POST –u "username:password" -H "Content-Type: application/json" –H "prodchannel=social" –F "prodtitle=Prod1" –F "file=@prod.json;filename=listfile" http://www.mycompanyinc.com/api/v1/products
- curl –k –u "username:password" –o file3.zip https://www.mycompanyinc.com/securefiles/file2.zip

1. **How do you write associative arrays in bash?**

- Use the below statements in the bash shell script to use associative arrays (to reference with a name instead of index numbers).
- declare -A myarray
- myarray[${VARNAME}]=${VAL}
- Referencing the value as ${myarray[${VARNAME}]}

1. **How do you run a command on the terminal without terminating when the shell closed or terminal closed?**

- Use “nohup” also called “no hangup” command. Usually used along with & to start background.
- Example: The below command starts the Jenkins process and stays without hanging-up even after the terminal closed.

nohup java -jar jenkins.war &

1. **How to see the help for commands in unix/linux?**

- Use “man” command.
- Typical usage is “man commandname”.

1. **What is a shell script? Write a script to do 1) Given Java code compilation and execution of class 2) Print hostname and how many java files are there in a given directory.**

- The shell script is an executable text file with inbuilt shell statements, and other executable commands put together to perform desired logic. It is usually having .sh or .csh extension and should have the executable permission.
- The shell provides the most of the programming constructs like if, while, for etc. to do branching of code or conditions or loops etc. and can be used in the shell script.
- The first line of the script indicates which interpreter or shell to use, and it starts with #!.
  - Example: #!/bin/bash or #!/bin/sh
- The input arguments to the script can be read using below variable references ($0 gives the filename itself)
  - $1 - first argument value
  - $2 - second argument value
  - ..
  - $9 - ninth argument value
  - ${10} - 10th argument value (note the curly braces and is to avoid the conflict with $1).
- Program#1: compileandrun.sh

```
#!/bin/sh
#####################
# Description: Compilation and run
#####################
        FILE_NAME=$1
javac -d . $FILE_NAME
# HelloWorld.java --> HelloWorld
CLASS_NAME="`echo $FILE_NAME | cut -f1 -d'.'`"
java $CLASS_NAME
```

- Program#2: countjavafiles.sh

```
#!/bin/sh
#############################################
# Description: A simple program for printing hostname
#############################################
echo "*** Hostname is `hostname` ***"
DIR=$1
NUM_FILES=`find $DIR -name "*.java" | wc -l`
echo $NUM_FILES java files found in $DIR
```

← End: Skill#8. Unix/Linux OS ←

Good!

Keep going and never give up!!

Please re-read again for more clarity and feel free to contact for help!!!

→ Start: Skill#9. Java →