-team building

member roles: leader, facilitator, scribe, time-keeper,
stages of team development:
Forming: 开始组队
Storming：开始work together
norming：beginning to work more effectively as a team. no longer focused on their individual goals, but rather are focused on developing a way of working together
performin：functioning at a very high level. The focus is on reaching the goal as a group. The team members have gotten to know each other, trust each other and rely on each other.
adjourning：time for the team to celebrate the success of the project and capture best practices for future use.
Team rules:
Communication and conflict resolution strategies:
ice-breakers
brainstorming

-debugging

print statements
breakpoints
step-in：The Step Into command will walk into any method calls that occur in the code, possibly opening additional files.
step-out of methods ：The Step Out command will finish running the current method and stop at the next line of code from where the method was called.
monitor variable values：

-testing

expected vs actual
unit testing vs end to end testing (integration testing):
unit testing 是单个单个看（每个部分的作用）
end to end testing 是看总体（开始和结尾）

INTEGRATION TESTING is a level of software testing where individual units are combined and tested as a group. The purpose of this level of testing is to expose faults in the interaction between integrated units. Test drivers and test stubs are used to assist in Integration Testing.

black box vs white box testing
assert the pre-conditions：assume other pre-requirements are correct.

test driven development
The following sequence of steps is generally followed:

- Add a test
- Run all tests and see if the new one fails
- Write some code
- Run tests
- Refactor code
- Repeat

junit tests
@Test

-Command-line development (linux commands and utilities)

Pwd: print directory
ls: list all files
cd: change directory
cd .. :go to previous directories
cp: copying files and directories
mv: mv is a unix command that moves one or more files or directories from one place to another. if both filenames are on the same filesystem, this results in a simple file rename;

rm: to remove objects such as computer files,
mkdir: make a new directory
rmdir: remove an empty directory
diff: compare files line by line

grep: search word in a file or files, and return the line

echo: ▸ Linux and Unix commands help

About echo

The **echo** program displays text. It's a handy way to create customized output in your terminal.

Javac

Java

Jar

Pico Linux and Unix commands help

About pico

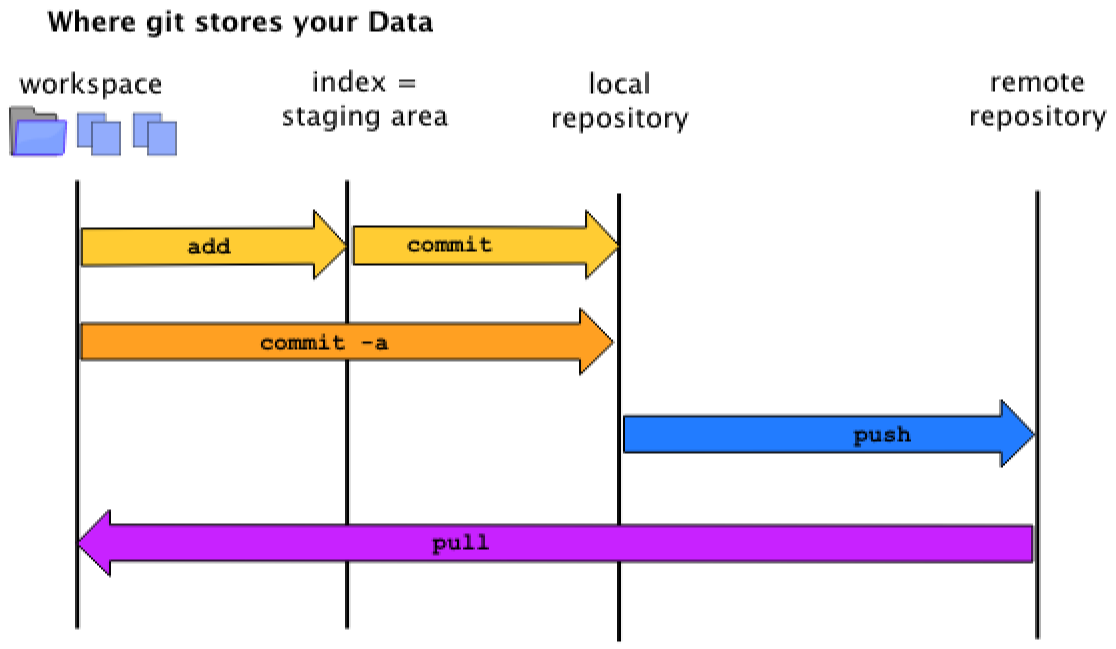
pico is a simple text editor in the style of the pine composer.

nano
Vi :提供兩種操作模式：輸入模式(insert mode)和指令模式(command mode), also a screen editor
:wq
Emacs :screen editor
zip

-make (w11t)
how to use make
define basic makefile rules

-git/github

init, status (current repository, and is there any file ready to be push), log (see history commit), clone, pull, add, commit, push,



-object-oriented design and diagrams

UML class diagrams (and class summary "table") object model, instance diagram, use-case diagram memory model (heap vs stack memory management)

-ADTs

position-oriented: lists, stacks(push pop), queues,

```
enqueue -> | 3 | 2 | 1 | -> dequeue
```

value-oriented: priority queues (heaps), search trees, hash tables, relation-oriented: graphs, sets,

Graphs:

Traversals

In-order:(Left, Root, Right)

Post-order:(Root, Left, Right)

Pre-order:(Left, Right, Root)

level-order: top to bottom

spanning trees

DFS:

BFS:

Prim's MST:

Kruskal's MST

finding paths & cycles,

Dijkstra's algorithm :

topological ordering

-Trees

terminology: height, root, leaf, interior node, ascendant, descendent, sibling, full binary
search trees: ordering constraint
search trees: full, complete, height-balanced, balanced,
avl: rotations

-RBT: root property, red property, black property, tri-node restructure, re-color b-tree:
2-node, 3-node, 4-node, 2-3 tree,
b+tree: m-branching factor, linked leaf nodes,
maintain balance on insert: detect, fix,

traversals: level-order, pre-order, in-order, post-order

-hash tables
ble, table size, load factor, key, value,
hash functions: int, string, double, other types
collision resolution: linear probe sequence, quadratic probe sequence, double hashing,
buckets: array, "chained", tree
resizing and rehashing
map types: (key:value pairs) hashmap vs treemap

-graphs: graphnode

terminology:
vertex (node)
edge
size = edge
order = number of vertice
undirected
directed
weighted = the number in edges
connected,
graph vs graphnode
degree of a node
implementations: adjacency matrix, adjacency list, edge list (vertex pairs)
complexity analysis of space and operation time

terminology: hash ta

operations: insert/remove vertices and edges traversals: depth-first, breadth-first, cycle-detection spanning trees: simple, depth-first, breadth-first, minimum spanning trees (mst): prim's, kruskals dijkstra's shortest path algorithm
topological ordering

-sets

Sets:

A = {3,7,9,14},
B = {9,14,28}

notation:

∆ :
symmetric difference

A = {3,9,14},
B = {1,2,3},
A ∆ B = {1,2,9,14}

∈ :
element of, belongs to
A={3,9,14}, 3 ∈ A

∉ :
not element of
A={3,9,14}, 1 ∉ A

∩ :
intersection
A ∩ B = {9,14}

∪ :
union
A ∪ B = {3,7,9,14,28}

⊂ :
proper subset / strict subset
{9,14} ⊂ {9,14,28}

⊃ :
proper superset / strict superset
{9,14,28} ⊃ {9,14}

$\not\subset$:
not subset
$\{9,66\} \not\subset \{9,14,28\}$

$\not\supset$:
not superset
$\{9,14,28\} \not\supset \{9,66\}$

$\subseteq$:
 subset
$\{9,14,28\} \subseteq \{9,14,28\}$

$\supseteq$:
 superset
$\{9,14,28\} \supseteq \{9,14,28\}$

$\nsubseteq$:
not subset
 $\{9,66\} \not\subset \{9,14,28\}$

$\nsupseteq$:
Not superset

$2^{A , p(A)}$
power set

notation: is a member of , union , intersection

operations: union, intersection, difference, symmetric difference

complexity analysis of various implementation options: array, list, hash table, tree

-sorting

radix sort（d = 執行回合數
n = 數列長度
r = 基數）: n, radix - range of unique digits, length of sequence - iterations, stable sort flashrm sort (長度,最大值,最小值): classification, peutation, insertion

a, n, amax, amin, m, ai, k(a[i]), use of boundary array (l)

-java 8

enumerations: define and use, .ordinal()
interfaces: abstract methods, constants, static methods, default methods,

Functional interfaces
A functional interface is an interface that contains only one abstract method.
A functional interface can have any number of default methods.

They can have only one functionality to exhibit. From Java 8 onwards, lambda expressions can be used to represent the instance of a functional interface. A functional interface can have any number of default methods. **Runnable**, **ActionListener**, **Comparable** are some of the examples of functional interfaces. Before Java 8, we had to create anonymous inner class objects or implement these interfaces.

Default method
Before Java 8, interfaces could have only abstract methods. The implementation of these methods has to be provided in a separate class. So, if a new method is to be added in an interface, then its implementation code has to be provided in the class implementing the same interface. To overcome this issue, Java 8 has introduced the concept of default methods which allow the interfaces to have methods with implementation without affecting the classes that implement the interface.

java.util.comparator

Comparator Interface

One obvious approach is to write our own sort() function using one of the standard algorithms. This solution requires rewriting the whole sorting code for different criterion like Roll No. and Name.
Using comparator interface- Comparator interface is used to order the objects of user-defined class. This interface is present in java.util package and contains 2 methods [compare(Object obj1, Object obj2)] and [equals(Object element)]. Using comparator, we can sort the elements based on data members. For instance it may be on rollno, name, age or anything else.

Comparable

A comparable object is capable of comparing itself with another object. The class itself must implements the **java.lang.Comparable** interface to compare its instances.

Consider a Movie class that has members like, rating, name, year. Suppose we wish to sort a list of Movies based on year of release. We can implement the Comparable interface with the Movie class, and we override the method compareTo() of Comparable interface.

inner classes
anonymous inner classes

https://www.cis.upenn.edu/~matuszek/General/JavaSyntax/inner-classes.html

lambda expressions

1. Enable to treat functionality as a method argument, or code as data.
2. A function that can be created without belonging to any class.
3. A lambda expression can be passed around as if it was an object and executed on demand.

Stream:
https://www.youtube.com/watch?v=t1-YZ6bF-g0
Introduced in Java 8, the Stream API is used to process collections of objects. A stream is a sequence of objects that supports various methods which can be pipelined to produce the desired result.

The features of Java stream are –

- A stream is not a data structure instead it takes input from the Collections, Arrays or I/O channels.
- Streams don't change the original data structure, they only provide the result as per the pipelined methods.
- Each intermediate operation is lazily executed and returns a stream as a result, hence various intermediate operations can be pipelined. Terminal operations mark the end of the stream and return the result.

Different Operations On Streams-

Intermediate Operations:

1. **map:** The map method is used to map the items in the collection to other objects according to the Predicate passed as argument.
   ```
   List number = Arrays.asList(2,3,4,5);
   List square = number.stream().map(x->x*x).collect(Collectors.toList());
   ```
2. **filter:** The filter method is used to select elements as per the Predicate passed as argument.
   ```
   List names = Arrays.asList("Reflection","Collection","Stream");
   List result =
   names.stream().filter(s->s.startsWith("S")).collect(Collectors.toList());
   ```
3. **sorted:** The sorted method is used to sort the stream.
   ```
   List names = Arrays.asList("Reflection","Collection","Stream");
   List result = names.stream().sorted().collect(Collectors.toList());
   ```

Terminal Operations:

1. **collect:** The collect method is used to return the result of the intermediate operations performed on the stream.
   ```
   List number = Arrays.asList(2,3,4,5,3);
   Set square = number.stream().map(x->x*x).collect(Collectors.toSet());
   ```
2. **forEach:** The forEach method is used to iterate through every element of the stream.
   ```
   List number = Arrays.asList(2,3,4,5);
   number.stream().map(x->x*x).forEach(y->System.out.println(y));
   ```
3. **reduce:** The reduce method is used to reduce the elements of a stream to a single value.
   The reduce method takes a BinaryOperator as a parameter.`List number =`
   ```
   Arrays.asList(2,3,4,5);
   int even = number.stream().filter(x->x%2==0).reduce(0,(ans,i)-> ans+i);
   ```

Here an variable is assigned 0 as the initial value and i is added to it

dictionarygraph: wordprocessor.getwordstream() - .map, .filter

-javafx

graphic user interface (design and layout)
controls: label, textfield, button, scrollbar, scrollpane
layout containers: borderpane, hbox, vbox, gridpane, getchildren().addall() scene
stage setscene(scene) .show()
events: setonaction, setonmouseentered, setonmouseexited

-html/css/js

html: html, head, title, body, header, footer, p, br, img, ol, ul, dl, table
chrome developer tools: elements, console, device view
css: use of a stylesheet for body, h1, class vs id style rules
js: var, function, if, else, for, while, return, console.log

HTML

html : tells the browser that this is an HTML document.
https://www.w3schools.com/tags/tag_head.asp
title :
head : https://www.w3schools.com/tags/tag_head.asp
The `head` tag is used for holding Meta information, title, links, etc. and is not displayed on the page.
body : https://www.w3schools.com/tags/tag_body.asp
defines the document's body
Header : 页眉
The `header` tag is used within the `body` of the website and can be used multiple times if required, e.g. to determine the top of an `article`.

footer : 页脚
p : paragraph
a : a hyperlink, which is used to link from one page to another.
img : <img src="smiley.gif" alt="Smiley face" height="42" width="42">
ol : ordered list https://www.w3schools.com/tags/tryit.asp?filename=tryhtml_lists
ul : unordered (bulleted) list.
https://www.w3schools.com/tags/tryit.asp?filename=tryhtml_lists4
dl : a definition list. https://www.w3schools.com/tags/tryit.asp?filename=tryhtml_dd_test
table : https://www.w3schools.com/tags/tryit.asp?filename=tryhtml_table_test

Chrome Developer Tools:
1. click Customize and control Google Chrome icon
2. mouse over More tools
3. click Developer Tools Developer tools allow you to:
● view and change page styles ● debug JavaScript ● view console message ● analyze

performance：
Elements:
Console:
Device view: