Comparison between Languages

Shark Deng

March 4 2019

Abstract

Development of compilers, such as clang and llvm, booms many computer languages. To name a few, java, swift, php, go, javascript, html, python, c, c++, c#. In this article, we will go through building blocks of a language. With these elements, we may pick up a new language very quickly or even can create one by llvm.

Contents

Chapter 1

Computer Language

1.1 Overview

Languages can be divided into these categories:

C Familiy C, C++, C#, Objective-C

Based on C Java, Javascript, Swift, Php, Python

ML XML, HTML, YAML

SQL(Structured Query Language) MySql

Shell AppleScript

Other Latex, Markdown

Swift has several features (**?**):

1. Tuples
2. Multiple return values
3. Structs that support methods, extensions, and protocols
4. Variables are always initialized before use

1.1.1 Oriented

Object Oriented Programming(OOP)[1], composed of attributes and behaviors, includes four aspects: encapsulation, inheritance, and polymorphism.

Encapsulation	Hiding the private details of a class from other objects.
Inheritance	A process of using details from a new class without modifying existing class.
Polymorphism	A concept of using common operation in different ways for different data input.

Table 1.1: Source: https://www.programiz.com/python-programming/object-oriented-programming

Object-oriented C++, C#, Objective-C, Java, Javascript, Python,Php

Procedure-oriented C

Protocol-oriented Swift

Graphic-oriented Swift

[1] https://www.programiz.com/python-programming/object-oriented-programming

1.1.2 Compile

Phases of compiling are illustrated in Figure 1.2. LLVM serves as IR.

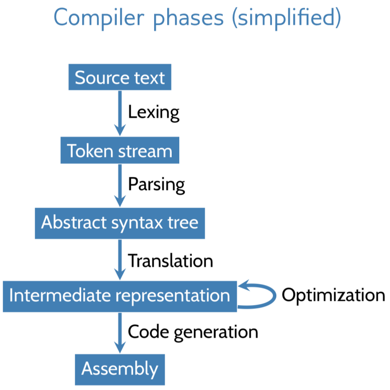


Figure 1.1: (?)

Table 1.2 will give an overview of three compile forms in these discussed languages.

Languages	Code on terminal	Compile file on terminal	Code on IDE
Java	No	Yes	Yes
Swift	Yes	Yes	Yes
Python	Yes	Yes	Yes

Table 1.2: Caption

Java

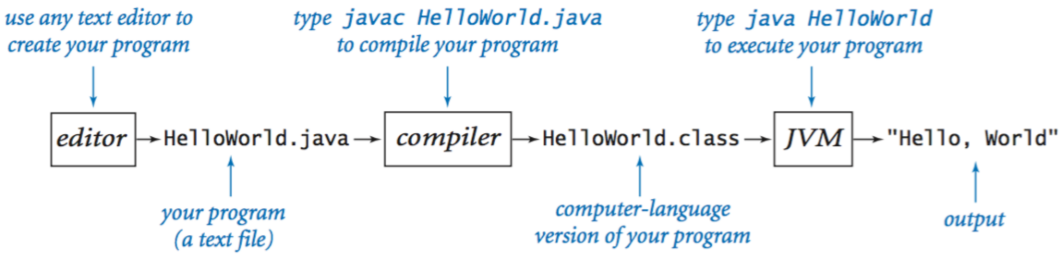

use any text editor to create your program
type javac HelloWorld.java to compile your program
type java HelloWorld to execute your program
editor
HelloWorld.java
compiler
HelloWorld.class
JVM
"Hello, World"
your program (a text file)
computer-language version of your program
output

Figure 1.2: (?)

1. **Code on terminal**: None
2. **Compile file on terminal**

```
    $ javac HelloWorld.java
    $ java HelloWorld
```

3. **Code on IDE**: IntelliJ. Features of the editor will be illustrated in the end.
4. **Product**: .jar

Swift

1. **Code on terminal**: Yes

```
    $ swift
Welcome to Apple Swift version 4.2.1. Type :help for assistance.
    1> var a = 88
a: Int = 88
    2> let b = 66
b: Int = 66
    3> let c = ”Hello!”
c: String = ”Hello!”
    4> let d = c + String(b)
d: String = ”Hello!66”
    5> import Foundation
```

   Control + d to exit.

2. **Compile file on terminal**: Yes

   Swift can be directly as an executable file. Just add the line !/usr/bin/swift on top of the file. And use chmod +x fibonacci.swift to allow the file has executive right. Run ./fobonacci.swift to execute the file. Sample code is from (**?**).

3. **Code on IDE**: Xcode

4. **Product**: .app

Python

1. **Code on terminal**

   Yes

```
 $ python
 >>> print(”Hello World”)
 >>> a = 10
 >>> import tensorflow
 >>> quit()
```

2. **Compile file on terminal**

```
 $ touch a.py
 $ echo ”print(’Hello World’)” >  a.py
    $ cat a.py
 $ python a.py
 $ rm a.py
```

3. **Code on IDE**

   Pycharm

Objective-C

1. **Code on terminal**

2. **Compile file on terminal**

```
    $ touch source.m
$ gcc -framework Foundation source.m -o source
$ ./source
```

3. **Code on IDE Xcode**

Php

1. **Code on IDE**: **PhpStorm**

C

1. **Code on IDE**: **Xcode**

C++

1. **Code on IDE**: **UE4**

C#

1. **Code on IDE**: **Unity**

1.1.3 Scalability

Java

Some languages have rich libraries.

Dependency Management
Maven.

Swift

```
import Foundation
```

Dependency Management
Codpods

Python

Every file in Python is **module**

Dependency Management
.yml

1.1.4 Annotation

Python

It uses # to annotate single line and ”’ for multiple lines.

1.1.5 Comments

Language	Single Line	Multiple Line
HMTL	<!– –>	

Table 1.3: Caption

1.2 Variable

1.2.1 Category

Strong or weak type means whether the language must specify its variable type. Swift is strong type because it's designed to run fast.So most of type check works is on programmers and IDE. Php, however, use Zend engine especially universal variable *zval* to parse variables, so its speed will be mush slower. C combines **strong and weak** types. In normal programming, it needs designate type but in micro programming, type is missed. Other languages, such as Java is strong type.

Static or dynamic type means whether the variable check is conducted during compile or run time. **Generic** means whether the type can be assigned to various type.

Language	Strong or Weak	Static or Dynamic	Support Generic
C	Strong		
C(micro)	Weak		
Swift	Strong		
Java	Strong		
Python	Strong		
PHP	Weak		

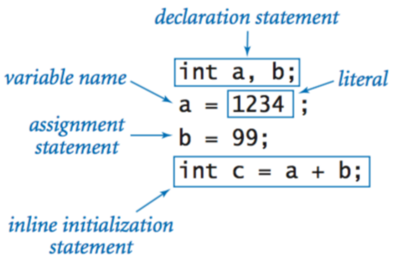
Table 1.4: Variable types in different languages

1.2.2 Statement

According to (**?**), there are three types of statements: declaration, assignment, and initialization. Declaration means.... Assignment means . Initialization means.

Table 1.5 is an overview of variable statements of these discussed languages. "-" means there are no specific supplement.

Languages	Supplement	Statements

Java	-	
		int a, b;
		a = 1234;
		b = 99;
		int c = a + b;
		 declaration statement
		variable name
		literal
		assignment statement
		inline initialization statement

Swift	Constant Variable Computed Variable Typealias	let a: Int = 1 var a: Int = 1

Table 1.5: Variable statements

1.2.3 Java

Variable type

Java has 4 types of variable. They are:

1. **Instance**
2. **Class**

   Described by **static** keyword.
3. **Local**
4. **Parameter and Argument**

   A formal parameter(also called parameter) is a variable named in the function definition. It will be a local variable that gets initialized as part of the function/method call.

   An actual parameter(also called argument) is a value that is supplied when a function is called. (**?**).

```
int square(int n) { return n * n; } // n is parameter
System.out.println(square(3)); // 3 is argument
```

Pass argument with value or address

Firstly, pass value

```
public static void main(String[] args) {
    int num1 = 100;
    int num2 = 200;
    take(num1, num2);
    System.out.println(”num1 = ” + num1);
    System.out.println(”num2 = ” + num2);
}

public static void take(int a, int b) {
    int temp = a;
    a = b;
    b = temp;
    System.out.println(”a = ” + a);
    System.out.println(”b = ” + b);
}
```

In this case, *a* and *b* copy the values of *num1* and *num2*, so changing *a* and *b* doesn't influence *num1* and *num2*.

Secondly, pass reference, namely address.

```
public static void main(String[] args) {
    int[] intArray = {1,2,3,4,5};
    change(intArray);
    System.out.println(intArray[0]);
}
public static void change(int[] array) {
    int len = array.length;
    array[0] = 0;
}
```

Thirdly, special reference still pass value.

Type Inference

1. **Generic**

2. **var** keywork

   With the var keyword, Java can infer the type of a local variable from its initialization expression

```
var theAnswer = 42;
var bike = new Bike();
var mystery; // invalid - no initializer
var nothing = null; // invalid - null has no type
```

3. **Lambda**

Data type

1. **Primitive data type - 8**

No.	Type	Description	Range	Default
1	byte	1-byte (8-bit) / signed / 2's complement integer	-128 - 127	0
2	short	2-byte (16-bit) / signed / 2's complement	-32768 - 32767	0
3	int	4-byte (32-bit) / signed / 2's complement	$-2^{31} - 2^{31} - 1$	0
4	long	8-byte (64-bit) / signed 2's complement	$-2^{63} - 2^{63} - 1$	0
5	float	(32-bit) single precision floating number		0.0f
6	double	(64-bit) double precision floating number		0.0d
7	boolean	logically just a single bit	true, false	false
8	char	2-byte (16-bit) Unicode character	0 - 65535	0

Table 1.6: Java 8 primitive data types

2. **Reference type - 3**

1	User user	class reference
2	java.lang.Runnable myThread	interface reference
3	int[] arr	array reference

Table 1.7: 3 reference types

3. **Boxing and Unboxing**

Table 1.8: Caption

Modifiers

Table below is an overview.

Name	Description
public	access control
private	access control
protected	access control
default	access control
static	non access
final	non access
abstract	non access
synchronized	thread
volatile	thread

Table 1.9: Caption

1.2.4 Python

__name__

Primitive data type

No.	Type	Description	Range	Default
1	Number	int float bool complex		
2	String	Non-changeable		
3	Tuple	Non-changeable		
4	List	Changeable		
5	Set	Changeable		
6	Dictionary	Changeable		

Use the code to check type

```
    type(10) # <class ’int’>
    type(5.5) # <class ’float’>
    type(True) # <class ’bool’>
    type(4+3i) # <class ’complex’>
    isinstance(10, int) # True
```

C

C++

C#

Objective-C

[2]

[2] $https://en.wikipedia.org/wiki/C_data_types$

1.3 Operator

Arithmetic Operators

No	Operator	Usage	Java	Python
1	+	Addition	✓	✓
2	-	Minus	✓	✓
3	*	Multiplication	✓	✓
4	/	Dvision	✓	✓
5	%	Modulus	✓	✓
6	++		✓	×
7	−−	Exponent	✓	×
8	//	Floor division	×	✓

Table 1.10: Arithmetic Operators

Relational Operators

No.	Operator	Usage	Java	Python
1	==	Equal to	✓	✓
2	!=	Not equal to	✓	✓
3	>	Greater than	✓	✓
4	<	Less than	✓	✓
5	>=	Greater than or Equal to	✓	✓
6	<=	Less than or Equal to	✓	✓

Table 1.11: Relational Operators

Bitwise Operators

No.	Operator	Usage
1	&	
2	|	
3		ˆ
4		˜
5	«	
6	»	
7	»>	

Table 1.12: Bitwise Operators

1.3.1 Logical Operators

No.	Operator	Usage	Java	Python
1	&&	and	✓	and
2	||	or	✓	or
3	!	not	✓	not

Table 1.13: Logical Operators

1.3.2 Assignment Operators

1.3.3 Misc Operators

1.4 Control Flow - 5

1.4.1 Selection - 3

If then else

Switch

Try ... except ...

1.4.2 Iteration - 2

For

While

1.5 Method

1.5.1 Define

Python

```python
def main():
    print('hello')

main()
```

1.6 Class

1.6.1 Access Level

Swift - 5

name	specifier	access	example
open		outside module (read and modify)	
public		outside module (read)	
internal		inside module	
fileprivate		inside file	
private		inside class	

Python - 3

name	specifier	access	example
public	-	-	self.public = 10
protected	single underline	-	self._protected = 10
private	double underlines	inside this class	self.__private = 10

1.6.2 Define

A class has **constructor**, **destructor**. These will be detailed in following table of responding languages:

Python

Example code for Magic Methods is available here Test5.py

Magic Methods	Meaning
__new__	create a new instance
__init__	constructor(initialize a new instance)
__del__	desctructor
__str__	print(obj)
__repr__	obj(on terminal)
__getitem__	
__setitem__	
__cmp__	
__eq__	=
__ne__	!=
__lt__	<
__gt__	>
__le__	<=
__ge__	>=
__add__	+
__sub__	-
__floordiv__	//
__truediv__	/
__mod__	%
__pow__	**
__lshift__	«
__rshift__	»
__and__	&
__xor__	
__or__	

Table 1.14: Magic methods

Python use **decorator** to realize static class and so on. Pre-made decorators are listed in the following table 1.15 and relevant example is Test1.py. Custom decorator is exampled here Test6.py.

Method Decorator	Meaning	Example
@staticmethod		
@classmethod		
@property	getter and setter	Test3.py

Table 1.15: Pre-made decorators

1.6.3 Java

Object is composed of state and behavior (**?**).

An **interface** is a group of methods without implementations (**?**). Any class that implements MovableThing must include definitions of these methods.

Inheritance forms a hierarchy by subclass **extends** super class

A **package** is a namespace that organizes a set of related classes and interfaces.

Abstract

According to (**?**), an abstract class is a class that is declared abstract, it may or may not include abstract methods. Abstract classes cannot be instantiated, but they can be subclassed.

An abstract method is a method that is declared without an implementation (without braces, and followed by a semicolon).

Abstract methods can be implemented or declared abstract in subclasses.

Inheritance

```
super.parentMethod();
```

Interface

Nested

Singleton

Enum

An enum is a special ”class” that represents a group of constants (unchangeable variables, like final variables) (**?**).

An enum can, just like a class, have attributes and methods. The only difference is that enum constants are public, static and final (unchangeable - cannot be overridden)

```
package DataStructure.Set;

import java.util.EnumMap;
import java.util.EnumSet;

public class Sample {
    enum Level {
        LOW, MEDIUM, HIGH;
    }

    enum Work {
        LOW(1),  //calls constructor with value 3
        MEDIUM(2),
        HIGH(3);

        private final int code;

        // 7 - constructor
        Work(int code) {
            this.code = code;
        }

        // 6 - method
        private int getCode() {
            return this.code;
        }
    }

    public static void main(String[] args) {
        Level a = Level.HIGH;
        System.out.println(a);


        //  1 – switch
        switch (a) {
            case LOW: System.out.println(”Low case”); break;
            case MEDIUM: System.out.println(”Medium case”); break;
            case HIGH: System.out.println(”High case”); break;
        }

        // 2 – if
        if (a == Level.HIGH) System.out.println(”Yes!”);

        // 3 – loop
        for (Level l:  Level.values()) {
            System.out.println(l);
        }

        // 4 – toString
        System.out.println(a.toString());

        // 5 – value
        Level b = Level.valueOf(”HIGH”);
        try {
            System.out.println(b);
        } catch (IllegalArgumentException e) {
            e.printStackTrace();
        }

        // 6 – methods
        Work w = Work.LOW;
        System.out.println(w); // print enum string
        System.out.println(w.getCode());


        // 8 – enum set: subset of the enum
        EnumSet<Level> set = EnumSet.of(Level.LOW, Level.HIGH);
        for(Level l: set) {
            System.out.println(l);
        }

        // 9 – enum map: enum as keys
        EnumMap<Level, Integer> map = new EnumMap<Level, Integer>(Level.class);
        map.put(Level.LOW, 10);
        map.put(Level.MEDIUM, 20);
        map.put(Level.HIGH, 30);
        System.out.println(map.get(Level.HIGH));

        }


}
```

Annotation

Classes, methods, variables, parameters and Java packages may be annotated. Built-in annotations: @override

1.7 Exception

1.8 Standard Library

1.8.1 Java

Comparable & Comparator

```
Internal Server Error
```

1.8.2 Python

Numpy

1. np.random.shuffle()
2. np.random.choice()

Chapter 2

Data structure and Algorithm

Data structure can be divided into linear and non-linear. Linear structure includes array and linked list. Non-linear structure includes tree and graph.

Data structure has a wide range of applications. For example, when studying natural language processing, Mr. Deng presented that put all study materials, such as thesis, models, code, data sets, and so on in download folder.

Abstract data types(ADT) is abstract, not concrete. List, set, map are ADTs.

2.1 Overview

2.1.1 Complexity

1. time
2. space
3. energy

Types of complexity:

1.

How to compute

1. Worst case
2. Average case
3. Best case

2.1.2 Types

1. Static and Dynamic
   (a) Static data structure: size is fixed, such as array.
   (b) Dynamic data structure: size is not fixed, such as linked list.
2. Linear and non-linear
   (a) Linear
   (b) Non-linear

2.1.3 Operations

1. add
2. remove
3. get specific node
4. get size
5. get whether is empty
6. get whether is full(only suitable for static type)
7. foreach all nodes

2.2 List

The list ADT is a container known mathematically as a *finite* sequence of elements. List has these 2 characteristics:

1. duplicates are allowed
2. order is preserved

List can be implemented by array, and linked list. Comparisons between them are as follows:

1. array:
   (a) easy to look up: O(1)
   (b) messy to grow and contract
2. linked list:
   (a) traverse list to find arbitrary element: O(n)
   (b) easy to grow, contract

Operations are:

```java
package DataStructure;

public interface List<T> {

    boolean add(T element);
    T get(int index);
    T remove(int index);
    int size();
    boolean contains(T element);
    boolean reverse();
    String toString();
}
```

2.2.1 Array

2.2.2 Single linked list

2.2.3 Double linked list

2.2.4 Circular single linked list

2.2.5 Circular double linked list

2.3 Stack

2.4 Queue

2.5 Tree

Characteristics:

1. tree is a node
2. a node contains value and a list of nodes
3. for value, it can be (key, value) pair or single key.
4. not allow duplicate items
5. ordered

2.5.1 Binary tree

Binary tree is different from binary search tree. Binary tree is each node has at most 2 sub nodes. Binary search tree is binary tree, in which the nodes are sorted, namely the value of the left child node is smaller than the value of the node and the value of the right node is bigger than the value of the node.

Complexity:

1. add
2. find
3. remove
4. foreach

2.5.2 Binary Search Tree - BST

2.5.3 Balanced Binary Search Tree - AVL

2.5.4 Huffman Tree

2.5.5 Quadtree

2.5.6 Octree

2.5.7 Red-Black tree

2.6 Graph

2.7 Set

1. not duplicate elements
2. unordered collection

2.8 Map

Characteristics:

1. (key, value) pairs

2.8.1 Java

Listing 2.1: Map.java

```java
public interface Map<K, V> {
    V put(K key, V value);
    V get(K key);
    boolean remove(K key);
    void clear();
    int size();
}
```

2.9 Hash Table

2.10 Finding

2.11 Sort

2.11.1 Merge sort

2.11.2 Selection sort

2.12 Python

2.12.1 List

```
empty = []  # create an empty list
letters = [‘a’, ‘b’] # create a list
numbers = [1, 2, 3]
mixed = [‘a’, 1]   # list can have mixed members
```

2.12.2 Dict

```
a = {‘Name’: ‘Zhang’, ‘Age’: 18, ‘Menu’:[‘Fruit’, ‘Vega’]} # create dict
a[‘Name’]   # visit value
a[‘Name’] = ‘Li’  # modify value
a[‘Schools’] = ‘Jobyme’ # add value
del a[’Name’]  # delete value
a.clear()   # clear the dict
del a     # delete the dict memory space
```

2.12.3 Set

2.12.4 Tuple

2.12.5 Collections-namedtuple

```
websites = [(’Sohu’, ’http://www.sohu.com’, ’Zhang’),
  (’Sina’, ’http://www.sina.com’, ’Li’),
  (’Yahoo’, ’http://yahoo.com’, ’Wu’)]
web = namedtuple(’Website’, [’name’, ’url’, ’author’])
for w in websites:
 w = web._make(w)
  print(w)
```

2.12.6 Collections-dequeue

2.12.7 Collections-Counter

2.12.8 Collections-defaultdict

2.12.9 Collections-OrderedDict

2.13 Java

2.13.1 HashSet

Characteristics:

1. unordered collection
2. ignore duplicate items

No	Function	API

1	create	HashSet()
		HashSet(Collection<? extends E> c)

2	add	add(E e)

3	remove	remove(Object o)
		removeAll(Collection<?> c)
		removeIf();
		clear()

4	get	size()
		contains(Object o)
		isEmpty()

5	foreach	

Table 2.1: Caption

stMap.java

Chapter 3

Algorithm

Three aspects to evaluate an algorithm: time, space, and energy.

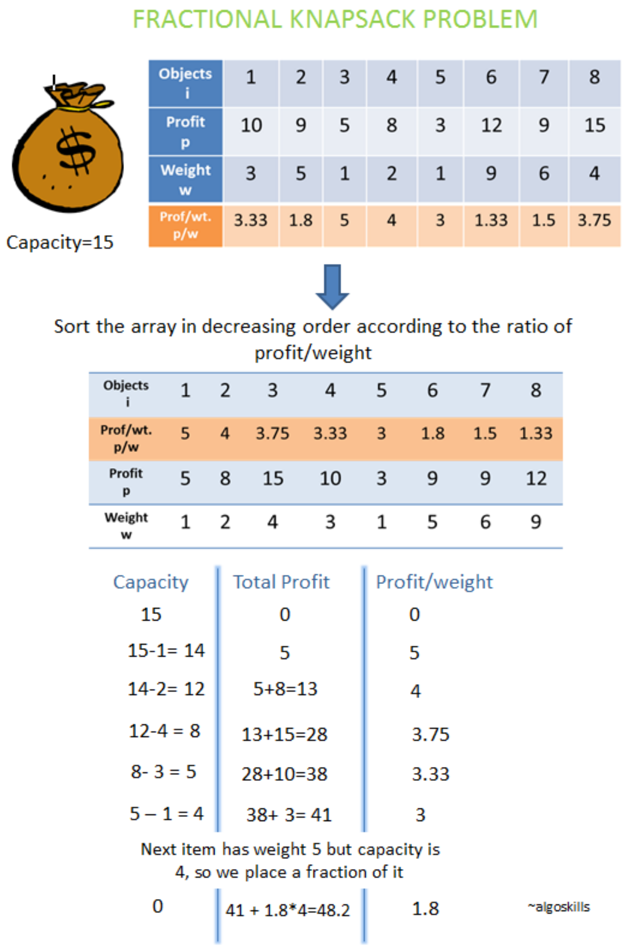
We consider worst case, best case, and average case for computation complexity.

Time	Example
Constant O(1)	
Logarithmic O(log(n))	finding an element in a balanced BST
Linear O(n) linked list	
O(n log(n))	average time to sort merge sort

Table 3.1: Caption

3.1 Greedy Algorithm

Fractional Knapsack Problem



FRACTIONAL KNAPSACK PROBLEM

Objects i	1	2	3	4	5	6	7	8
Profit p	10	9	5	8	3	12	9	15
Weight w	3	5	1	2	1	9	6	4
Prof/wt. p/w	3.33	1.8	5	4	3	1.33	1.5	3.75

Capacity=15

Sort the array in decreasing order according to the ratio of profit/weight

Objects i	1	2	3	4	5	6	7	8
Prof/wt. p/w	5	4	3.75	3.33	3	1.8	1.5	1.33
Profit p	5	8	15	10	3	9	9	12
Weight w	1	2	4	3	1	5	6	9

Capacity	Total Profit	Profit/weight
15	0	0
15-1= 14	5	5
14-2= 12	5+8=13	4
12-4 = 8	13+15=28	3.75
8- 3 = 5	28+10=38	3.33
5 – 1 = 4	38+ 3= 41	3

Next item has weight 5 but capacity is 4, so we place a fraction of it

0	41 + 1.8*4=48.2	1.8

~algoskills

Figure 3.1: Caption

Chapter 4

Core Computer Science

4.1 Basic IO

4.2 Basic data manipulation

4.3 File

4.4 Thread

In concurrent programming, thread, process, and coroutine are important.

Each process has its own memory space.

Threads share the process's resources, including memory and open files.

Reasons for using threads are concurrency and parallelism.

4.4.1 Java

A Java application can create additional processes using a **ProcessBuilder** object.

The **Thread** class is used to create threads and interact with them.

Two ways to create a thread: (**?**)

1. Subclass Thread,extending its run()method
   - Advantages: class inherits all of Thread's methods
   - Disadvantages: can't subclass anything else
2. Use the Runnable interface and implement its run() method.
   - General, but does not inherit Thread's methods

Related methods:

1. t.start() will start execution of the run() method within the thread t. If the there are multiple threads, they will start at the same time.
2. t.join() will cause the current thread to wait until thread t terminates. Usually it is the main thread to wait other threads to complete.

Race condition and synchronized keyword

1. synchronized is to qualify a method, ensures only one thread executes that method at any time

Samples: `Worker.java` `Tuna.java`

4.5 Recursion

4.5.1 Fibonnaci

4.6 Math

4.7 UI

4.7.1 Java-JavaFX

4.7.2 HTML

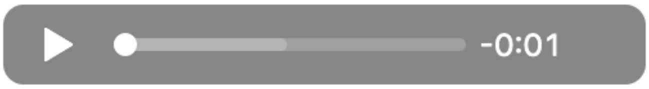
1. **<meta>**

```
<meta charset=”UTF-8”>
<meta name=”description” content=””>
<meta name=”keywords” content=””>
<meta name=”author” content=””>
<!-- responsive -->
<meta name=”viewport” content=”width=device-width, initial-scale=1.0”>
<!-- Refresh document every 30 seconds -->
<meta http-equiv=”refresh” content=”30”>
```

Media

2. **<audio>(<source>)** controls shows sound bar

```
<audio controls>
    <source src=”horse.ogg” type=”audio/ogg”>
    <source src=”horse.mp3” type=”audio/mpeg”>
</audio>
```



3. **<video>**

```
<video width=”400” controls>
    <source src=”mov_bbb.mp4” type=”video/mp4”>
    <source src=”mov_bbb.ogg” type=”video/ogg”>
    Your browser does not support HTML5 video.
</video>
```

4. **<iframe>** To embed Youtube video, simply right click to get code.

```
<iframe width=”806” height=”453” src=”https://www.youtube.com/embed/V1ejrlYY1Qs?list=LL7CoUzO3ZFpw–UbKQhDRweA” frameborder=”0” allow=”accelerometer; autoplay; encrypted–media; gyroscope; picture–in–picture” allowfullscreen>
</iframe>
```

5. **<object>** It is used to embed plug-ins (like Java applets, PDF readers, Flash Players) in web pages. (**?**)

```
<object width=”400” height=”50” data=”bookmark.swf”></object>
<object width=”100%” height=”500px” data=”snippet.html”></object>
```

6. **<p>**

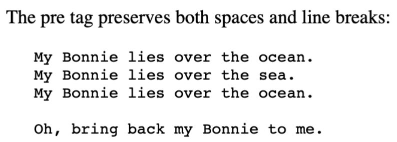
Tag	Description
<i>	
<strong>	
<em>	
<u>	underline text
<br>	break line
<hr>	horizon line
<sub>	subscript text
<sup>	superscript text
<h1> - <h6>	headers

Table 4.1: Caption

7. **<pre>**

```
<pre>
    My Bonnie lies over the ocean.
    My Bonnie lies over the sea.
    My Bonnie lies over the ocean.

    Oh, bring back my Bonnie to me.
</pre>
```



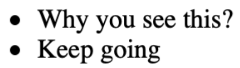
The pre tag preserves both spaces and line breaks:

```
My Bonnie lies over the ocean.
My Bonnie lies over the sea.
My Bonnie lies over the ocean.

Oh, bring back my Bonnie to me.
```

8. **<ul>**: unordered list

```
<ul>
    <li>Menu one</li>
    <li>Menu two</li>
</ul>
```



- Why you see this?
- Keep going

9. **<ol>**

10. **<dl>**

Graphic

11. **<svg>**

12. **<canvas>**

13. **<a>**

14. **<img>**

HTML 5 Semantic

15. **<header>(<article>, <main>, <footer>, <section>, <aside>)**

```
<header>
    <nav>
        Nav part
    </nav>
</header>
<main>
    <article>
        <section>
```

```
            <aside>
                Section one
            </aside>
        </section>
    </article>
</main>
<footer>
        Copyright &copy 2019
</footer>
```

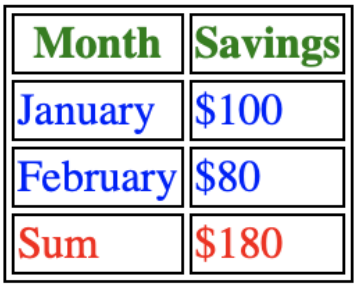
16. **<table> (<tr>, <th>, <td>, <thead>, <tbody>, <tfoot>)**

    <tr> defines a row and there are two tabel cells: <th> header cell and <td> standard cell.

    The <thead>, <tbody>, and <tfoot> will not affect the layout of the table by default. However, you can use CSS to style these elements.

Listing 4.1: table.html

```
<!DOCTYPE html>
<html>
    <head>
    <style>
    thead {color:green;}
    tbody {color:blue;}
    tfoot {color:red;}
    table, th, td {
      border: 1px solid black;
    }
    </style>
    </head>
    <body>
    <table>
      <thead>
        <tr>
          <th>Month</th>
          <th>Savings</th>
        </tr>
      </thead>
      <tbody>
        <tr>
          <td>January</td>
          <td>$100</td>
        </tr>
        <tr>
          <td>February</td>
          <td>$80</td>
        </tr>
      </tbody>
      <tfoot>
        <tr>
          <td>Sum</td>
          <td>$180</td>
        </tr>
      </tfoot>
    </table>
    </body>
</html>
```



Month	Savings
January	$100
February	$80
Sum	$180

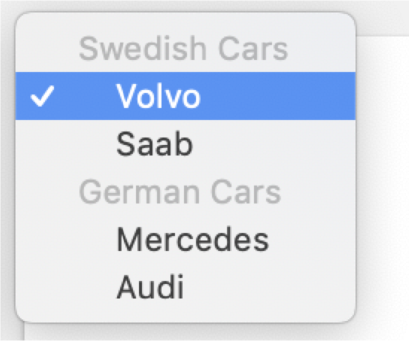
17. <form>

```
<form action=”/action_page.php”>
    First name: <input type=”text” name=”FirstName” value=”Mickey”><br>
    Last name: <input type=”text” name=”LastName” value=”Mouse”><br>
    <input type=”submit” value=”Submit”>
</form>
```

18. **<select>(<optgroup>, <option>**

A drop-down list

```
<select>
    <optgroup label=”Swedish Cars”>
        <option value=”volvo”>Volvo</option>
        <option value=”saab”>Saab</option>
    </optgroup>
    <optgroup label=”German Cars”>
        <option value=”mercedes”>Mercedes</option>
        <option value=”audi”>Audi</option>
    </optgroup>
</select>
```



19. **<detail>(<summary>)**

```
<details>
    <summary>Copyright 1999–2014.</summary>
    <p> – by Refsnes Data. All Rights Reserved.</p>
    <p>All content and graphics on this web site are the property of the company Refsnes Data.</p>
</details>
```

▼ Copyright 1999-2014.

- by Refsnes Data. All Rights Reserved.

All content and graphics on this web site are the property of the company Refsnes Data.

20. **<dialog>**

21. **<menu>**

Chapter 5

Software Industry

5.1 IDE

5.1.1 Java-IntelliJ

5.1.2 Swift-Xcode

5.1.3 C#-Unity

5.1.4 C++-UE4

5.1.5 Python-Pycharm

5.2 Version Control

1. Commit
2. Push
3. Pull
4. Update
5. Revert

5.3 Test

5.3.1 Java-Junit

5.3.2 Swift-UITest

5.3.3 Python-pytest

5.4 Debug

5.5 Code Naming Tradition

5.5.1 Java

1. case-sensitive
2. Whitespace not permitted

3. special characters are forbidden
4. Java keywords and reserved words cannot be used
5. Class names start with capital letters
6. Variable names start with lower case, and use upper case for subsequent words
7. Constant names use all caps and underscores

5.6 Packaging

5.7 Problem Shooting

Listings