Java

1. The difference between Java and C++.

First of all, their initial design is different. C++ is an extending of C, but support OOP, exception handling, generic programming and so on. Java however is initially designed as an OOP language. The difference here is that you can also write non-OOP code with C++, but you cannot write OOP with Java, Java forces you to write in an OOP way.

Second, C++ run on OS level but Java run inside JVM, so C++ programs need to be compiled against a particular OS but Java compiled classes are portable, no matter what OS it is, as long as it has JVM.

Third, C++ has pointers, Java does not want you to deal with pointers directly.

Fourth, Java has a garbage collector to free unused resources.

2. Abstract class vs. Interface

We use abstract class when we want to share code among several closely related classes. We use interface when you expect that unrelated classes would implement your interface. For example, the interfaces Comparable and Cloneable are implemented by many unrelated classes.

First, Interface only contains methods signatures and fields, not implementations. Abstract class can contain a mix of methods declared with or without an implementation.

Second, with interfaces, all fields are automatically public, static, and final, and all methods that you declare or define (as default methods) are public. But with abstract classes, you can declare fields that are not static and final, and define public, protected, and private concrete methods.

Third, You can extend only one class, whether or not it is abstract, whereas you can implement any number of interfaces.

3. Shallow copy vs. deep copy

Default clone() method creates the shallow copy of an object. The shallow copy of an object will have exact copy of all the fields of original object. If original object has any references to other objects as fields, then only references of those objects are copied into clone object, copy of those objects are not created. That means any changes made to those objects through clone object will be reflected in original object or vice-versa. Shallow copy is not 100% disjoint from original object. Shallow copy is not 100% independent of original object. Shallow copy is preferred if an object has only primitive fields. Shallow copy is fast and also less expensive.

Deep copy of an object will have exact copy of all the fields of original object just like shallow copy. But in additional, if original object has any references to other objects as fields, then copy of those objects are also created by calling clone() method on them. That means clone object and original object will be 100% disjoint. They will be 100% independent of each other. Any changes made to clone object will not be reflected in original object or vice-versa. To create a deep copy of an object, you have to override the clone() method. Deep copy is preferred if an object has references to other objects as fields. Deep copy is slow and very expensive.

4. Pass by reference vs. pass by value

Java is pass-by-value.

Pass by value: make a copy in memory of the actual parameter's value that is passed in.

Pass by reference: pass a copy of the address of the actual parameter.

This code will not swap anything:

```
swap(Type arg1, Type arg2) {
    Type temp = arg1;
    arg1 = arg2;
    arg2 = temp;
}
```

For this code, since the original and copied reference refer the same object, the member value gets changed:

```
class Apple {
        public String color="red";
}

public class Main {
        public static void main(String[] args) {
                Apple apple = new Apple();
                System.out.println(apple.color);

                changeApple(apple);
                System.out.println(apple.color);
        }

        public static void changeApple(Apple apple){
                apple.color = "green";
        }
}
```

5. Hashcode() / equals()

Defined in java.lang.Object

```
public boolean equals(Object obj)
public int hashCode()
```

The contract between equals() and hashCode() is:

1) If two objects are equal, then they must have the same hash code.

2) If two objects have the same hash code, they may or may not be equal.

So if we want to use HashSet<self-defined object>, we need to override both hashcode() and equal().

6. Final / finalize /finally

Final is a keyword, final is used to apply restrictions on class, method and variable. Final class can't be inherited, final method can't be overridden and final variable value can't be changed.

```
class FinalExample{
public static void main(String[] args){
final int x=100;
x=200;//Compile Time Error
}}
```

Finally is a block, finally is used to place important code, it will be executed whether exception is handled or not.

```
lock.lock();
try {
  //do stuff
} catch (SomeException se) {
  //handle se
} finally {
  lock.unlock(); //always executed, even if Exception or Error or se
}
```

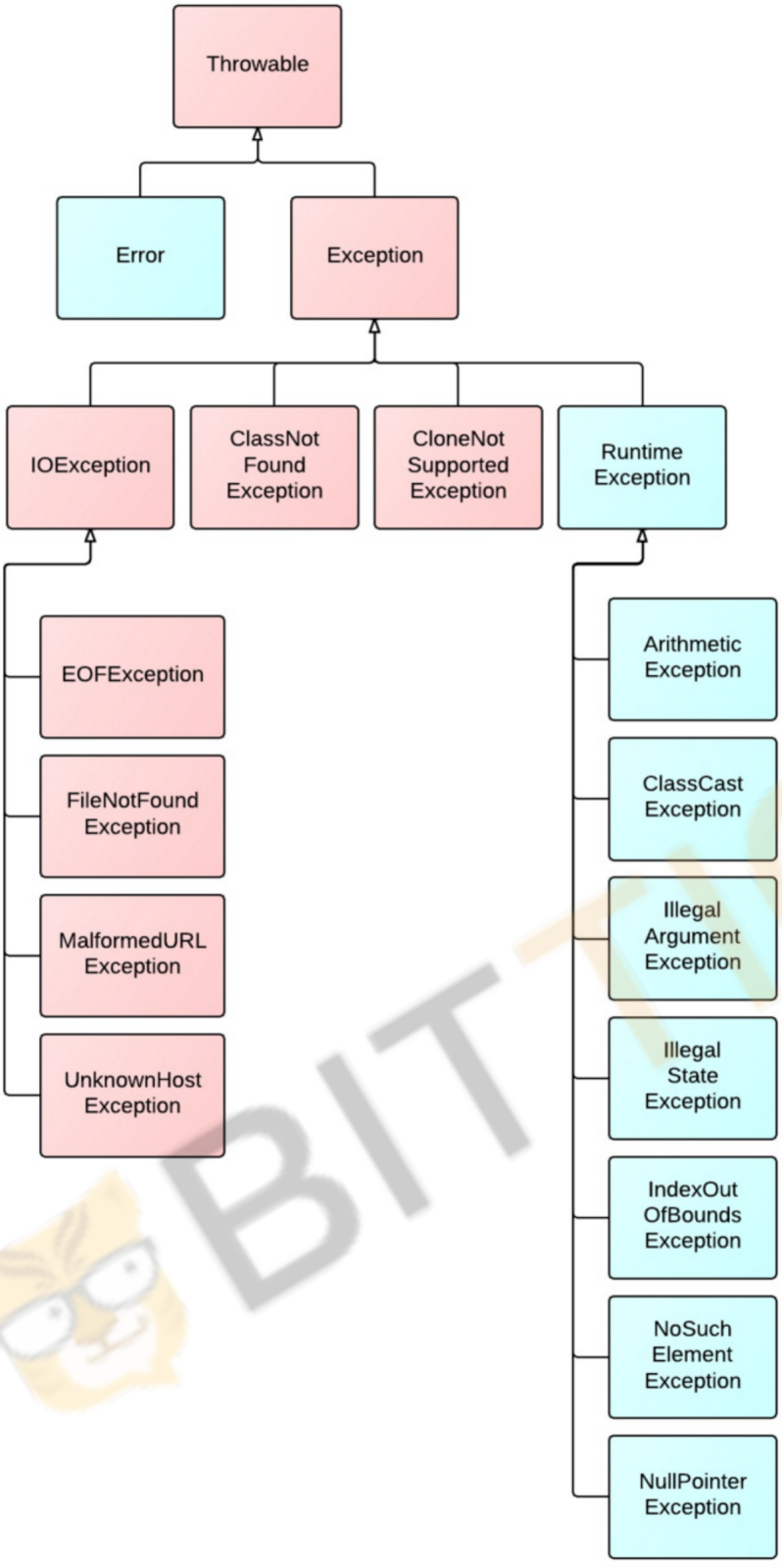
Finalize is a method, finalize is used to perform clean up processing just before object is garbage collected.

```
class FinalizeExample{
public void finalize(){System.out.println("finalize called");}
public static void main(String[] args){
FinalizeExample f1=new FinalizeExample();
FinalizeExample f2=new FinalizeExample();
f1=null;
f2=null;
System.gc();
}}
```

7. Checked / unchecked exception

Checked exceptions are the exceptions that are checked at compile time. If some code within a method throws a checked exception, then the method must either handle the exception or it must specify the exception using throws keyword.

Unchecked exceptions are the exceptions that are not checked at compiled time. In C++, all exceptions are unchecked, so it is not forced by the compiler to either handle or specify the exception. It is up to the programmers to be civilized, and specify or catch the exceptions. In Java exceptions under Error and RuntimeException classes are unchecked exceptions, everything else under throwable is checked.



8. Primitive types

Byte: 8-bit signed two's complement integer, [-128, 127]. The byte data type can be useful for saving memory in large arrays, where the memory savings actually matters.

Short: 16-bit signed two's complement integer. Save memory in large arrays of floating point numbers.

Int: 32-bit signed two's complement integer, [-2^{31}, 2^{31}-1]. In Java SE 8 and later, you can use the int data type to

represent an unsigned 32-bit integer, which has a minimum value of 0 and a maximum value of 2^{32}-1. Static methods like compareUnsigned, divideUnsigned etc have been added to the Integer class to support the arithmetic operations for unsigned integers.
Long: 64-bit two's complement integer, [-2^{63}, 2^{63}-1]. In Java SE 8 and later, you can use the long data type to represent an unsigned 64-bit long, which has a minimum value of 0 and a maximum value of 2^{64}-1.
Float: single-precision 32-bit IEEE 754 floating point. Save memory in large arrays of floating point numbers. This data type should never be used for precise values, such as currency. For that, you will need to use the java.math.BigDecimal class instead. Numbers and Strings covers BigDecimal and other useful classes provided by the Java platform.
Double: double-precision 64-bit IEEE 754 floating point. For decimal values, this data type is generally the default choice. As mentioned above, this data type should never be used for precise values, such as currency.
Boolean: true / false. Use this data type for simple flags that track true/false conditions. This data type represents one bit of information, but its "size" isn't something that's precisely defined.
Char: single 16-bit Unicode character. It has a minimum value of '\u0000' (or 0) and a maximum value of '\uffff' (or 65,535 inclusive).

9. Overriding vs. overloading

Overriding means having two methods with the same method name and parameters (i.e., method signature). One of the methods is in the parent class and the other is in the child class. Overriding allows a child class to provide a specific implementation of a method that is already provided its parent class.
The real object type in the run-time, not the reference variable's type, determines which overridden method is used at runtime. In contrast, reference type determines which overloaded method will be used at compile time.
Polymorphism applies to overriding, not to overloading.
Overriding is a run-time concept while overloading is a compile-time concept.

10. Public static void main(String[] args)

Public is the visibility. This can be public, private, protected or default.

	Class	Package	Subclass	World
public	y	y	y	y
protected	y	y	y	n
no modifier	y	y	n	n
private	y	n	n	n

Static indicates that this method can be called without creating an instance of this class. Without it, you have to instantiate this class and call this method from the resulting object.
Void is the return type of this method, indicating that this method doesn't return anything.
Main is the name of a function. main() is special because it is the start of the program.
String[] args: In Java args contains the supplied command-line arguments as an array of String objects. In other words, if you run your program as java MyProgram one two then args will contain ["one", "two"].

11. Stack (LIFO) vs. Queue (FIFO)
public class **Stack<E>** extends Vector<E>

boolean	Empty()
E	Push(E item)
E	Pop()
E	Peek()
int	Search(Object o) Returns the 1-based position where an object is on this stack.

(**Vector:**
The Vector class implements a growable array of objects. Like an array, it contains components that can be accessed using an integer index. However, the size of a Vector can grow or shrink as needed to accommodate adding and removing items after the Vector has been created. Each vector tries to optimize storage management by maintaining a capacity and a capacityIncrement. The capacity is always at least as large as the vector size; it is usually larger because as components are added to the vector, the vector's storage increases in chunks the size of capacityIncrement. An application can increase the capacity of a vector before inserting a large number of components; this reduces the amount of incremental reallocation.)
(**Vector vs. ArrayList:**
Vector is synchronized, ArrayList is not.
A Vector defaults to doubling the size of its array, while the ArrayList increases its array size by 50 percent.)

public interface **Queue<E>** extends Collection<E>
A collection designed for holding elements prior to processing.

	Throws exception	Returns special value
insert	Add(e) unchecked exception	Offer(e) return boolean
remove	Remove(e) NoSuchElementException	Poll(e) return E

examine	Element() NoSuchElementException	Peek() return E

12. HashSet vs. TreeSet vs. LinkedHashSet

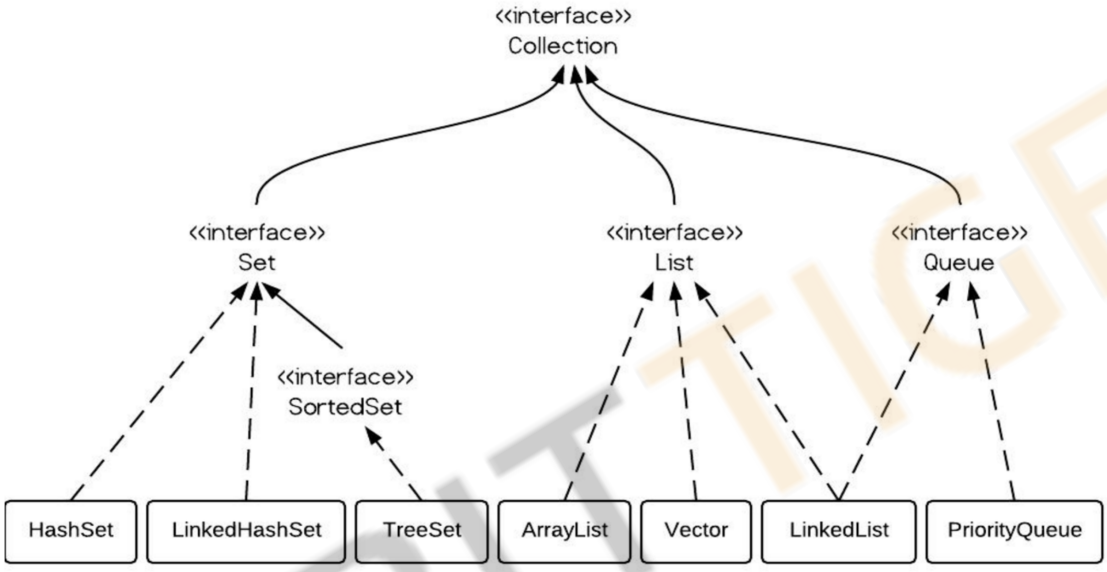
Set interface: A Set contains no duplicate elements. That is one of the major reasons to use a set.

When and which to use is an important question. In brief, if you need a fast set, you should use HashSet; if you need a sorted set, then TreeSet should be used; if you need a set that can be store the insertion order, LinkedHashSet should be used.

HashSet is Implemented using a hash table. Elements are not ordered. The add, remove, and contains methods have constant time complexity O(1).

TreeSet is implemented using a tree structure(red-black tree in algorithm book). The elements in a set are sorted, but the add, remove, and contains methods has time complexity of O(log (n)). It offers several methods to deal with the ordered set like first(), last(), headSet(), tailSet(), etc.

LinkedHashSet is between HashSet and TreeSet. It is implemented as a hash table with a linked list running through it, so it provides the order of insertion. The time complexity of basic methods is O(1).



13. Stirng vs. StringbBuffer

String class is used to manipulate character strings that cannot be changed. Objects of type String are read only and immutable.

The StringBuffer class is used to represent characters that can be modified.

The significant performance difference between these two classes is that StringBuffer is faster than String when performing simple concatenations.

StringBuffer(int capacity)
StringBuffer(String str)
Append(E e)
charAt(int index)
Delete(int start, int end)
Insert(int offset, E e)
Length()

Substring(int start) substring(int start, int end)

14. HashMap vs. TreeMap vs. HashTable vs. LinkedHashMap vs. ConcurrentHashMap

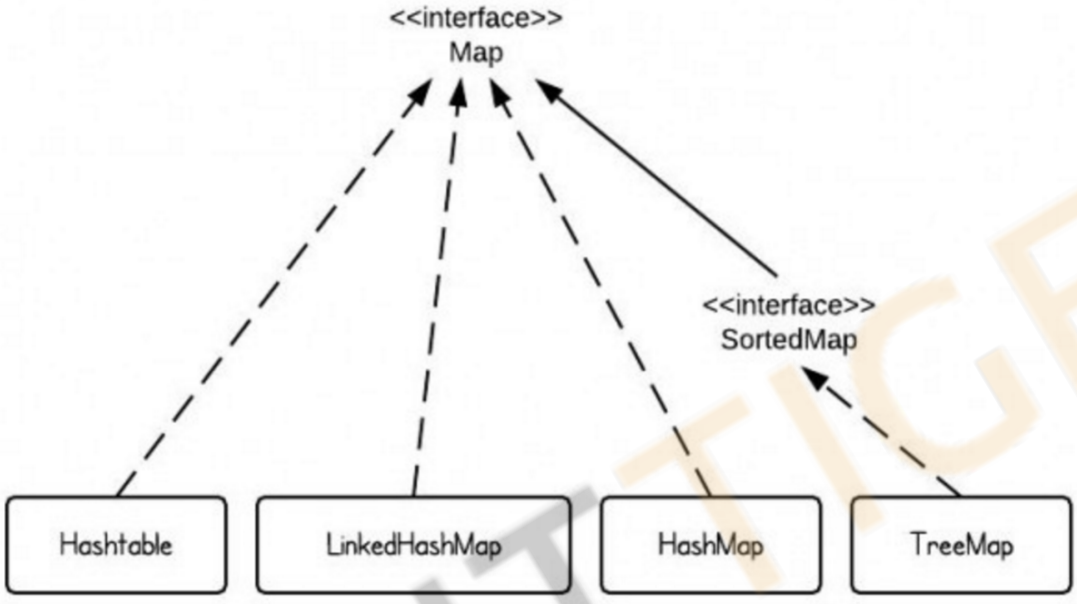
HashMap is implemented as a hash table, and there is no ordering on keys or values.

TreeMap is implemented based on red-black tree structure, and it is ordered by the key.

LinkedHashMap preserves the insertion order

Hashtable is synchronized, in contrast to HashMap. It has an overhead for synchronization. All methods of Hashtable are synchronized which makes them quite slow due to contention if a number of thread increases.

ConcurrentHashMap is thread safe without synchronizing the whole map. Reads can happen very fast while write is done with a lock. There is no locking at the object level. Unlike Hashtable and Synchronized Map, it never locks whole Map, instead, it divides the map into segments and locking is done on those. Though it performs better if a number of reader threads are greater than the number of writer threads.



15. Array vs. ArrayList vs. LinkedList

Array is a container object that holds a fixed number of values of a single type.

Array has fix size, can contain primitive types, can be multi-dimensional, while ArrayList is resizable, cannot contain primitive types, only one-dimensional.

ArrayList and LinkedList both implements List interface, maintain the insertion order, non-synchronized (can use Collections.synchronizedList to synchronize). Arraylist is implemented by array. Linked list is implemented by doubly linked list.

Search: ArrayList search operation is pretty fast compared to the LinkedList search operation. get(int index) in ArrayList gives the performance of O(1) while LinkedList performance is O(n).

Deletion: LinkedList remove operation gives O(1) performance while ArrayList gives variable performance: O(n) in worst case (while removing first element) and O(1) in best case (While removing last element).

Inserts Performance: LinkedList add method gives O(1) performance while ArrayList gives O(n) in worst case.

Memory Overhead: ArrayList maintains indexes and element data while LinkedList maintains element data and two pointers for neighbor nodes hence the memory consumption is high in LinkedList comparatively.

16. PriorityQueue(heap)

An unbounded priority queue based on a priority heap.

Not synchronized, otherwise use the thread-safe PriorityBlockingQueue.

Implementation note: this implementation provides O(log(n)) time for the enqueing and dequeing methods (offer, poll, remove() and add); linear time for the remove(Object) and contains(Object) methods; and constant time for the retrieval methods (peek, element, and size).

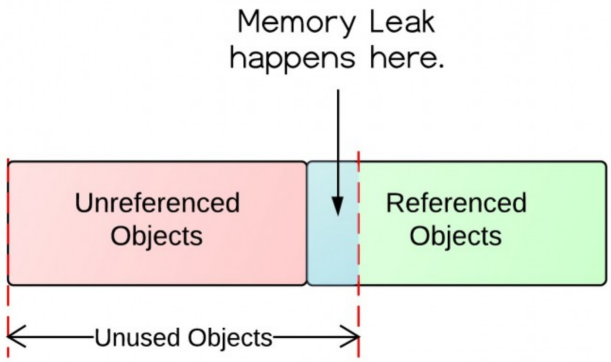
17. Comparable vs. Comparator

Comparable is implemented by a class in order to be able to comparing object of itself with some other objects. First, the class needs to implement Comparable, and in the class need to override compareTo method. (public int compareTo(E e))

In some situations, you may not want to change a class and make it comparable. In such cases, Comparator can be used if you want to compare objects based on certain attributes/fields. We need to write a MyComparator class that implements Comparator interface, and then override compare method. (public int compare(E e1, E e2))

18. Java memory leak

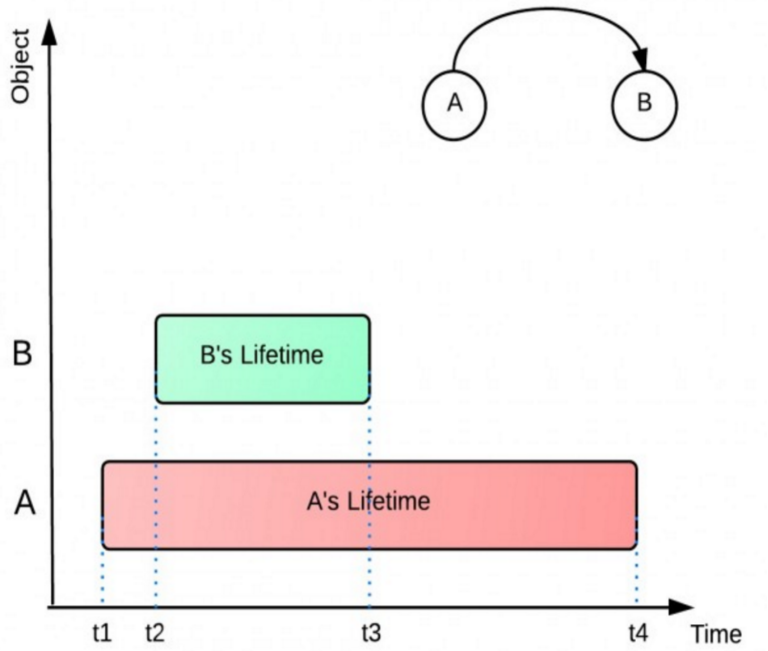
Objects are no longer being used by the application, but Garbage Collector can not remove them because they are being referenced.



Why Memory Leaks Happen?

Let's take a look at the following example and see why memory leaks happen. In the example below, object A refers to object B. A's lifetime (t1 - t4) is much longer than B's (t2 - t3). When B is no longer being used in the application, A still holds a reference to it. In this way, Garbage Collector can not remove B from memory. This would possibly cause out of memory problem, because if A does the same thing for more objects, then there would be a lot of objects that are uncollected and consume memory space.

It is also possible that B hold a bunch of references of other objects. Those objects referenced by B will not get collected either. All those unused objects will consume precious memory space.



substring() method in JDK 6 can cause memory leaks

```
//JDK 6
String(int offset, int count, char value[]) {
        this.value = value;
        this.offset = offset;
        this.count = count;
}

public String substring(int beginIndex, int endIndex) {
        //check boundary
        return  new String(offset + beginIndex, endIndex - beginIndex, value);
}
```

If you have a VERY long string, but you only need a small part each time by using substring(). This will cause a performance problem since you need only a small part, you keep the whole thing.

```
x = x.substring(x, y) + ""
```

19. JVM

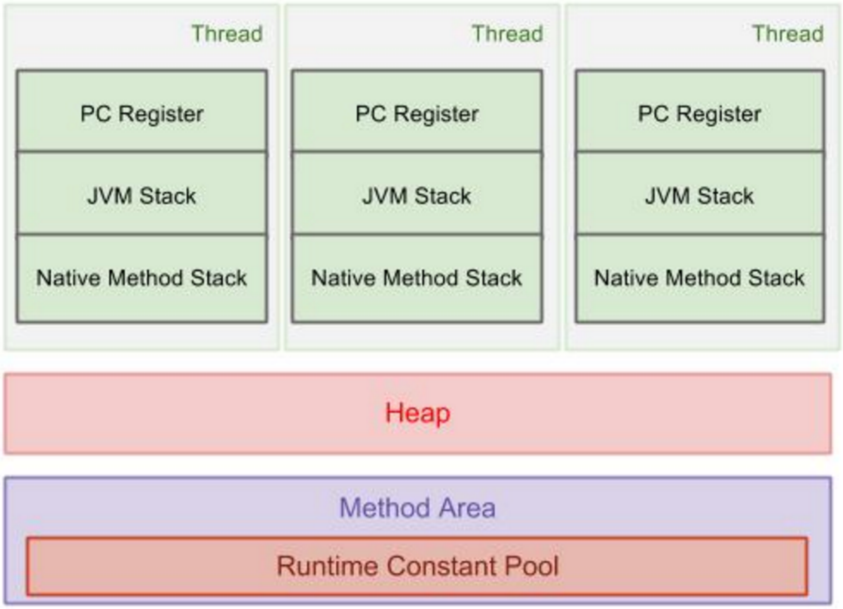
When you download Java, you get the Java Runtime Environment **(JRE)**. The JRE consists of the Java Virtual Machine (JVM), Java platform core classes, and supporting Java platform libraries. All three are required to run Java applications on your computer.

The Java Development Kit **(JDK)** is a collection of tools for developing Java applications. With the JDK, you can compile programs written in the Java Programming language and run them in a JVM. In addition, the JDK provides tools for packaging and distributing your applications.

The JDK and the JRE share the Java Application Programming Interfaces **(Java API)**. The Java API is a collection of prepackaged libraries developers use to create Java applications. The Java API makes development easier by providing the tools to complete many common programming tasks including string manipulation, date/time processing, networking, and implementing data structures (e.g., lists, maps, stacks, and queues).

The Java Virtual Machine **(JVM)** is an abstract computing machine. The JVM is a program that looks like a machine to the programs written to execute in it. This way, Java programs are written to the same set of interfaces and libraries. Each JVM implementation for a specific operating system, translates the Java programming instructions into instructions and commands that run on the local operating system. This way,

Java programs achieve platform independence.



1. Data Areas for Each Individual Thread (not shared)
Data Areas for each individual thread include program counter register, JVM Stack, and Native Method Stack. They are all created when a new thread is created.
Program Counter Register - used to control each execution of each thread.
JVM Stack - contains frames which is demonstrated in the diagram below.
Native Method Stack - used to support native methods, i.e., non-Java language methods.
2. Data Areas Shared by All Threads
All threads share Heap and Method Area.
Heap: it is the area that we most frequently deal with. It stores arrays and objects, created when JVM starts up. Garbage Collection works in this area.
If a computation requires more heap than can be made available by the automatic storage management system, the Java Virtual Machine throws an OutOfMemoryError.
Method Area: it stores run-time constant pool, field and method data, and methods and constructors code.
If memory in the method area cannot be made available to satisfy an allocation request, the Java Virtual Machine throws an OutOfMemoryError.
Runtime Constant Pool: It is a per-class or per-interface run-time representation of the constant_pool table in a class file. It contains several kinds of constants, ranging from numeric literals known at compile-time to method and field references that must be resolved at run-time.

20. Garbage collection

Live objects are tracked and everything else designated garbage.
There is a term that you should know before learning about GC. The term is "stop-the-world." Stop-the-world will occur no matter which GC algorithm you choose. Stop-the-world means that the JVM is stopping the application from running to execute a GC. When stop-the-world occurs, every thread except for the threads needed for the GC will stop their tasks. The interrupted tasks will resume only after the GC task has completed. GC tuning often means reducing this stop-the-world time.
This garbage collector was created based on the following two hypotheses:

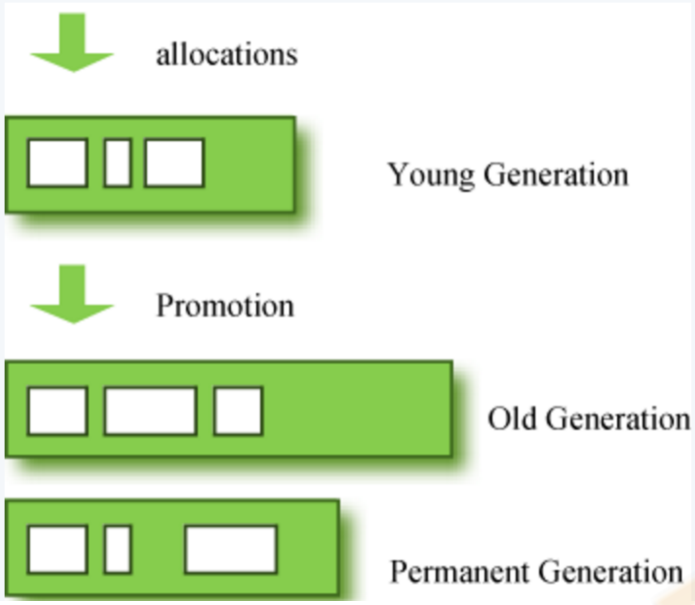
Most objects soon become unreachable.
References from old objects to young objects only exist in small numbers.

These hypotheses are called the weak generational hypothesis. So in order to preserve the strengths of this hypothesis, it is physically divided into two - young generation and old generation.

Young generation: Most of the newly created objects are located here. Since most objects soon become unreachable, many objects are created in the young generation, then disappear. When objects disappear from this area, we say a **"minor GC"** has occurred.

Old generation: The objects that did not become unreachable and survived from the young generation are copied here. It is generally larger than the young generation. As it is bigger in size, the GC occurs less frequently than in the young generation. When objects disappear from the old generation, we say a **"major GC"** (or a **"full GC"**) has occurred.

The permanent generation from the chart above is also called the "method area," and it stores classes or interned character strings. So, this area is definitely not for objects that survived from the old generation to stay permanently. A GC may occur in this area. The GC that took place here is still counted as a major GC.



What if an object in the old generation need to reference an object in the young generation?

To handle these cases, there is something called the a "card table" in the old generation, which is a 512 byte chunk. Whenever an object in the old generation references an object in the young generation, it is recorded in this table. When a GC is executed for the young generation, only this card table is searched to determine whether or not it is subject for GC, instead of checking the reference of all the objects in the old generation. This card table is managed with write barrier. This write barrier is a device that allows a faster performance for minor GC. Though a bit of overhead occurs because of this, the overall GC time is reduced.

Composition of the Young Generation

In order to understand GC, let's learn about the young generation, where the objects are created for the first time. The young generation is divided into 3 spaces.

One Eden space

Two Survivor spaces

There are 3 spaces in total, two of which are Survivor spaces. The order of execution process of each space is as below:

The majority of newly created objects are located in the Eden space.

After one GC in the Eden space, the surviving objects are moved to one of the Survivor spaces.

After a GC in the Eden space, the objects are piled up into the Survivor space, where other surviving objects already exist.

Once a Survivor space is full, surviving objects are moved to the other Survivor space. Then, the Survivor space that is full will be changed to a state where there is no data at all.

The objects that survived these steps that have been repeated a number of times are moved to the old generation.

As you can see by checking these steps, one of the Survivor spaces must remain empty. If data exists in both Survivor spaces, or the usage is 0 for both spaces, then take that as a sign that something is wrong with your system.

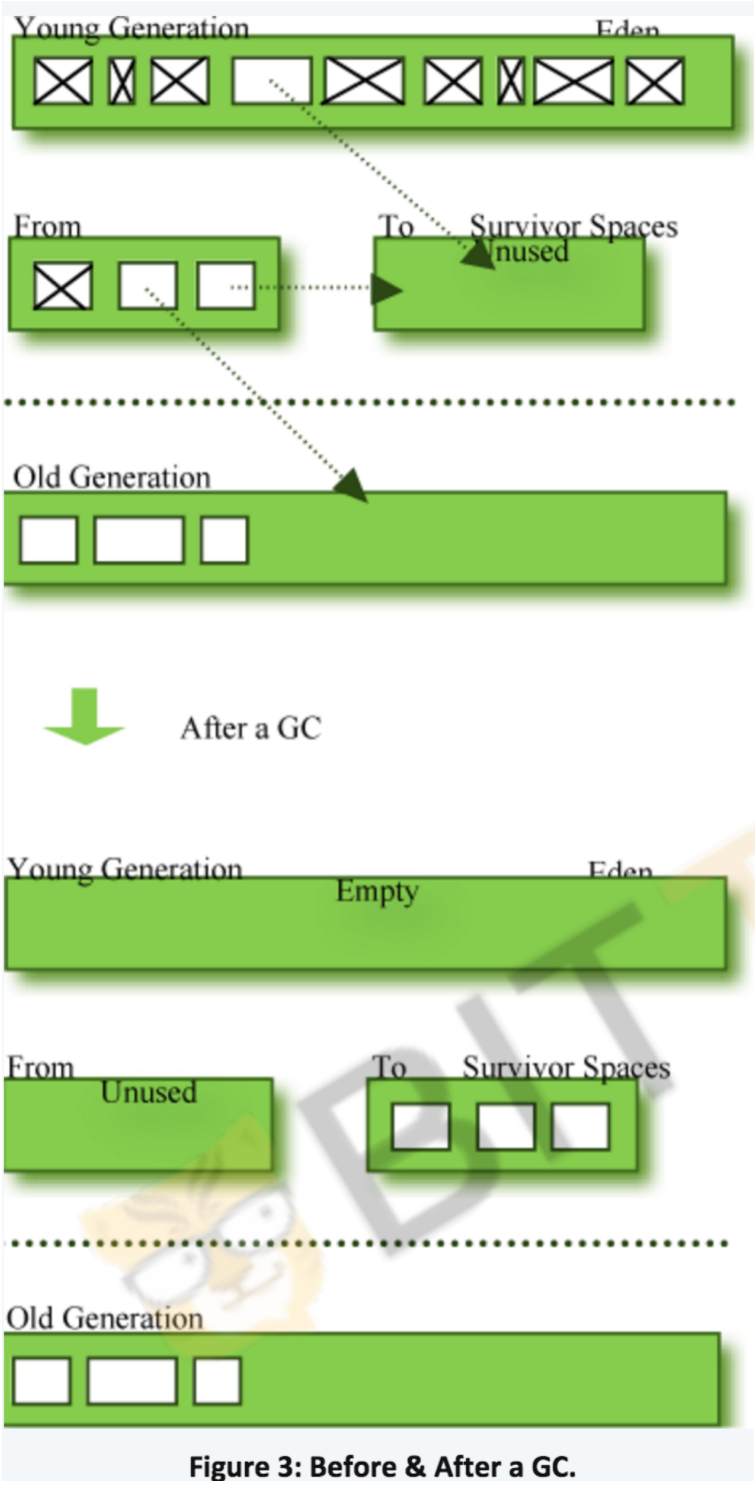


Figure 3: Before & After a GC.

21. Object class method

Object class is the superclass of all Java classes. All Java classes inherited from this class.

method	r e t u r n type	description

clone()	Protected Obejcet	Creates and returns a copy of this object.
equals(Object obj)	boolean	Indicates whether some other object is "equal to" this one.
finalize()	Protected void	Called by the garbage collector on an object when garbage collection determines that there are no more references to the object.
getClass()	Class<?>	Returns the runtime class of this Object.
hashcode()	int	Returns a hash code value for the object.
notify()	void	Wakes up a single thread that is waiting on this object's monitor.
notifyAll()	void	Wakes up all threads that are waiting on this object's monitor.
toString()	String	Returns a string representation of the object.
wait() throws InterruptedException	void	Causes the current thread to wait until another thread invokes the notify() method or the notifyAll() method for this object.
wait(long timeout) throws InterruptedException	void	Causes the current thread to wait until either another thread invokes the notify() method or the notifyAll() method for this object, or a specified amount of time has elapsed.
wait(long timeout, int nanos) throws InterruptedException	void	Causes the current thread to wait until another thread invokes the notify() method or the notifyAll() method for this object, or some other thread interrupts the current thread, or a certain amount of real time has elapsed.

22. Java heap/stack

The main difference between heap and stack is that stack memory is used to store local variables and function call while heap memory is used to store objects in Java. No matter, where the object is created in code e.g. as a member variable, local variable or class variable, they are always created inside heap space in Java.

Each Thread in Java has their own stack which can be specified using -Xss JVM parameter, similarly, you can also specify heap size of Java program using JVM option -Xms and -Xmx where -Xms is starting size of the heap and -Xmx is a maximum size of java heap.

If there is no memory left in the stack for storing function call or local variable, JVM will throw java.lang.StackOverFlowError, while if there is no more heap space for creating an object, JVM will throw java.lang.OutOfMemoryError: Java Heap Space.

4) If you are using Recursion, on which method calls itself, You can quickly fill up stack memory. Another difference between stack and heap is that size of stack memory is a lot lesser than the size of heap memory in Java.

5) Variables stored in stacks are only visible to the owner Thread while objects created in the heap are visible to all thread. In other words, stack memory is kind of private memory of Java Threads while heap memory is

shared among all threads.

23. Java 8 vs. java7

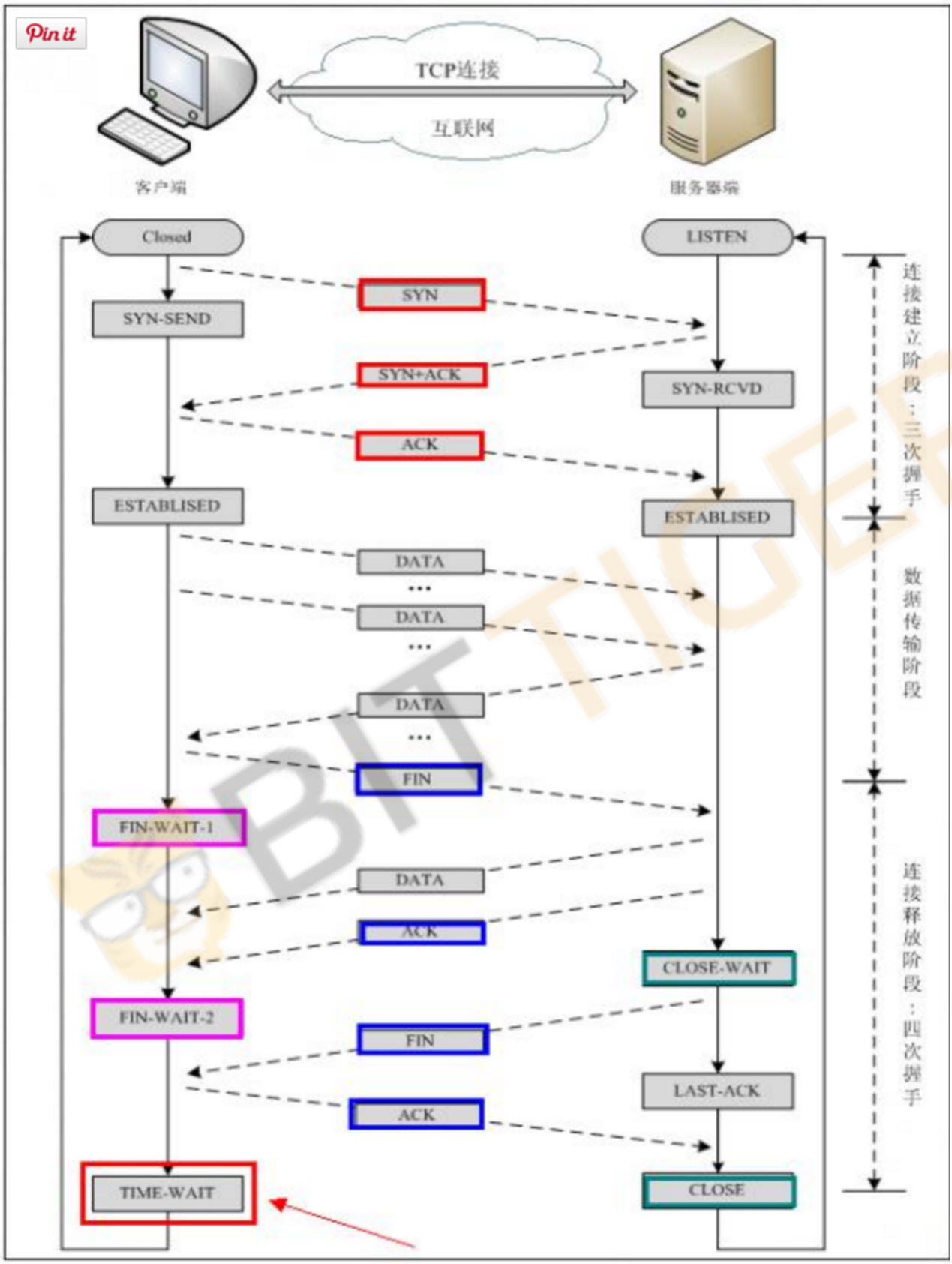
Language-level support for lambda expressions; unofficially under Project Lambda
Default methods (virtual extension methods) which make multiple inheritance possible in Java.
a JavaScript runtime which allows developers to embed JavaScript code within applications.

24. volatile

25.

网络

建立TCP需要三次握手才能建立，而断开连接则需要四次握手。整个过程如下图所示：



Difference between TCP and UDP

TCP	UDP
Reliable: TCP is **connection-oriented protocol**. When a file or message send it will get delivered unless connections fails. If connection lost, the server will request the lost part. There is **no corruption** while transferring a message.	***Not Reliable***: UDP is **connectionless** protocol. When you a send a data or message, you don't know if it'll get there, it could get lost on the way. There may be **corruption** while transferring a message.
Ordered: If you send two messages along a connection, one after the other, you know the first message will get there first. You don't have to worry about data arriving in the wrong order.	***Not Ordered***: If you send two messages out, you don't know what order they'll arrive in i.e. **no ordered**
Heavyweight: – when the low level parts of the TCP "stream" arrive in the wrong order, resend requests have to be sent, and all the out of sequence parts have to be put back together, so requires a bit of work to piece together.	***Lightweight***: No ordering of messages, no tracking connections, etc. It's just fire and forget! This means it's **a lot quicker**, and the network card / OS have to do very little work to translate the data back from the packets.
Streaming: Data is read as a "stream," with nothing distinguishing where one packet ends and another begins. There may be multiple packets per read call.	***Datagrams***: Packets are sent individually and are guaranteed to be whole if they arrive. One packet per one read call.
Examples: World Wide Web (Apache TCP port 80), e-mail (SMTP TCP port 25 Postfix MTA), File Transfer Protocol (FTP port 21) and Secure Shell (OpenSSH port 22) etc.	*Examples*: Domain Name System (DNS UDP port 53), streaming media applications such as IPTV or movies, Voice over IP (VoIP), Trivial File Transfer Protocol (TFTP) and online multiplayer games etc

Difference between HTTP and HTTPS

HTTPS stands for Hypertext Transfer Protocol Secure. HTTPS is a protocol which uses an encrypted HTTP connection by transport-layer security.

Using HTTPS, the computers agree on a "code" between them, and then they scramble the messages using that "code" so that no one in between can read them. This keeps your information safe from hackers.

They use the "code" on a Secure Sockets Layer (SSL), sometimes called Transport Layer Security (TLS) to send the information back and forth.

不使用SSL/TLS的HTTP通信，就是不加密的通信。所有信息明文传播，带来了三大风险。

> (1) **窃听风险**（eavesdropping）：第三方可以获知通信内容。
>
> (2) **篡改风险**（tampering）：第三方可以修改通信内容。
>
> (3) **冒充风险**（pretending）：第三方可以冒充他人身份参与通信。

SSL/TLS协议是为了解决这三大风险而设计的，希望达到：

> (1) 所有信息都是**加密传播**，第三方无法窃听。
>
> (2) 具有**校验机制**，一旦被篡改，通信双方会立刻发现。
>
> (3) 配备**身份证书**，防止身份被冒充。

SSL/TLS协议的基本思路是采用公钥加密法，也就是说，客户端先向服务器端索要公钥，然后用公钥加密信息，服务器收到密文后，用自己的私钥解密。

但是，这里有两个问题。

(1) 如何保证公钥不被篡改?

> 解决方法：将公钥放在数字证书中。只要证书是可信的，公钥就是可信的。

(2) 公钥加密计算量太大，如何减少耗用的时间?

> 解决方法：每一次对话（session），客户端和服务器端都生成一个"对话密钥"（session key），用它来加密信息。由于"对话密钥"是对称加密，所以运算速度非常快，而服务器公钥只用于加密"对话密钥"本身，这样就减少了加密运算的消耗时间。

因此，SSL/TLS协议的基本过程是这样的：

> (1) 客户端向服务器端索要并验证公钥。
>
> (2) 双方协商生成"对话密钥"。
>
> (3) 双方采用"对话密钥"进行加密通信。

What's HTTP

The Hypertext Transfer Protocol (HTTP) is designed to enable communications between clients and servers.

HTTP works as a request-response protocol between a client and server.

A web browser may be the client, and an application on a computer that hosts a web site may be the server.

GET and POST
GET - Requests data from a specified resource
POST - Submits data to be processed to a specified resource
GET:

```
GET /path HTTP/1.1
```

GET requests can be cached
GET requests remain in the browser history
GET requests can be bookmarked
GET requests should never be used when dealing with sensitive data
GET requests have length restrictions
GET requests should be used only to retrieve data
POST:

```
POST /test/demo_form.asp HTTP/1.1
Host: w3schools.com
name1=value1&name2=value2
```

POST requests are never cached
POST requests do not remain in the browser history
POST requests cannot be bookmarked
POST requests have no restrictions on data length

	GET	POST
BACK button/Reload	Harmless	Data will be re-submitted (the browser should alert the user that the data are about to be re-submitted)
Bookmarked	Can be bookmarked	Cannot be bookmarked
Cached	Can be cached	Not cached
Encoding type	application/x-www-form-urlencoded	application/x-www-form-urlencoded or multipart/form-data. Use multipart encoding for binary data
History	Parameters remain in browser history	Parameters are not saved in browser history
Restrictions on data length	Yes, when sending data, the GET method adds the data to the URL; and the length of a URL is limited (maximum URL length is 2048 characters)	No restrictions
Restrictions on data type	Only ASCII characters allowed	No restrictions. Binary data is also allowed
Security	GET is less secure compared to POST because data sent is part of the URL Never use GET when sending passwords or other sensitive information!	POST is a little safer than GET because the parameters are not stored in browser history or in web server logs
Visibility	Data is visible to everyone in the URL	Data is not displayed in the URL

Ports
20: FTP
22: SSH
53: DNS
80: HTTP
输入www.google.com会发生什么

1. The browser extracts the domain name from the URL.
2. The browser queries DNS for the IP address of the URL. Generally, the browser will have cached domains previously visited, and the operating system will have cached queries from any number of applications. If neither the browser nor the OS have a cached copy of the IP address, then a

request is sent off to the system's configured DNS server. The client machine knows the IP address for the DNS server, so no lookup is necessary.

3. The request sent to the DNS server is almost always smaller than the maximum packet size, and is thus sent off as a single packet. In addition to the content of the request, the packet includes the IP address it is destined for in its header. Except in the simplest of cases (network hubs), as the packet reaches each piece of network equipment between the client and server, that equipment uses a routing table to figure out what node it is connected to that is most likely to be part of the fastest route to the destination. The process of determining which path is the best choice differs between equipment and can be very complicated.
4. The is either lost (in which case the request fails or is reiterated), or makes it to its destination, the DNS server.
5. If that DNS server has the address for that domain, it will return it. Otherwise, it will forward the query along to DNS server it is configured to defer to. This happens recursively until the request is fulfilled or it reaches an authoritative name server and can go no further. (If the authoritative name server doesn't recognize the domain, the response indicates failure and the browser generally gives an error like "Can't find the server at www.lkliejafadh.com".) The response makes its way back to the client machine much like the request traveled to the DNS server.
6. Assuming the DNS request is successful, the client machine now has an IP address that uniquely identifies a machine on the Internet. The web browser then assembles an HTTP request, which consists of a header and optional content. The header includes things like the specific path being requested from the web server, the HTTP version, any relevant browser cookies, etc. In the case in question (hitting Enter in the address bar), the content will be empty. In other cases, it may include form data like a username and password (or the content of an image file being uploaded, etc.)
7. This HTTP request is sent off to the web server host as some number of packets, each of which is routed in the same was as the earlier DNS query. (The packets have sequence numbers that allow them to be reassembled in order even if they take different paths.) Once the request arrives at the webserver, it generates a response (this may be a static page, served as-is, or a more dynamic response, generated in any number of ways.) The web server software sends the generated page back to the client.
8. Assuming the response HTML and not an image or data file, then the browser parses the HTML to render the page. Part of this parsing and rendering process may be the discovery that the web page includes images or other embedded content that is not part of the HTML document. The browser will then send off further requests (either to the original web server or different ones, as appropriate) to fetch the embedded content, which will then be rendered into the document as well.

Public key and private key
MD5 and SHA-1 algorithms

Public-key cryptography, or **asymmetric cryptography**, is any cryptographic system that uses two kinds of keys: public keys may be disseminated widely, while private keys are known only to the owner. In a public-key encryption system, any person can encrypt a message using the public key (better imagined as a lock) of the receiver and leave it on a public server or transmit it on a public network. Such a message can be decrypted only with the receiver's private key.

(Symmetric-key algorithms are algorithms for cryptography that use the same cryptographic keys for both encryption of plaintext and decryption of ciphertext. The keys may be identical or there may be a simple transformation to go between the two keys.)

Public key algorithms, unlike symmetric key algorithms, do not require a secure channel for the initial exchange of one (or more) secret keys between the parties.

Because of the computational complexity of asymmetric encryption, it is usually used only for small blocks of

data, typically the transfer of a symmetric encryption key (e.g. a session key). This symmetric key is then used to encrypt the rest of the potentially long message sequence. The symmetric encryption/decryption is based on simpler algorithms and is much faster.

HTTP Error

The first digit of the HTTP Status Code specifies one of five classes of response. The bare minimum for an HTTP client is that it recognizes these five classes. A first digit of 1, 2, or 3 represents a fully functional request. A first digit of 4 represents a client—side error, with the most common codes in the range of 400 to 404. A first digit of 5 represents a server—side error, with the most common codes in the range of 500 to 510. Because the codes in 400 and 500 range represent errors, they are also referred to as HTTP Error Codes.

400 Bad Request

This error indicates that the user's request contains incorrect syntax.

401 Unauthorized

This error indicates that the requested file requires authentication (a username and password).

403 Forbidden

This error indicates that the server will not allow the visitor to access the requested file. If a visitor receives this code unexpectedly, you should check the file's permission settings, or check whether the file has been protected.

404 Not Found

This error indicates that the server could not find the file that the visitor requested. This commonly occurs when a URL is mistyped.

5xx Errors

These errors are caused by the server being unable to fulfill an apparently valid request from a visitor. Often, you will need the help of a server administrator to investigate them.

502 Bad Gateway

This error is usually due to improperly configured proxy servers. However, the problem may also arise when there is poor IP communication between back—end computers, when the client's server is overloaded, or when a firewall is functioning improperly.

The first step in resolving the issue is to clear the client's cache. This action should result in a different proxy being used to resolve the web server's content.

503 Service Unavailable

This error occurs when the server is unable to handle requests due to a temporary overload or due to the server being temporarily closed for maintenance. The error indicates that the server will only temporarily be down. It is possible to receive other errors in place of 503.

504 Gateway Timeout

This error occurs when a server somewhere along the chain does not receive a timely response from a server further up the chain. The problem is caused entirely by slow communication between upstream computers.

Client/Server Model

The client–server model of computing is a distributed application structure that partitions tasks or workloads between the providers of a resource or service, called servers, and service requesters, called clients.[1] Often clients and servers communicate over a computer network on separate hardware, but both client and server may reside in the same system. A server host runs one or more server programs which share their resources with clients. A client does not share any of its resources, but requests a server's content or service function. Clients therefore initiate communication sessions with servers which await incoming requests.

Examples of computer applications that use the client–server model are Email, network printing, and the World Wide Web.

Clients and servers exchange messages in a request–response messaging pattern: The client sends a request, and the server returns a response. This exchange of messages is an example of inter-process communication. To

communicate, the computers must have a common language, and they must follow rules so that both the client and the server know what to expect. The language and rules of communication are defined in a communications protocol. All client-server protocols operate in the application layer. The application-layer protocol defines the basic patterns of the dialogue. To formalize the data exchange even further, the server may implement an API (such as a web service). The API is an abstraction layer for such resources as databases and custom software. By restricting communication to a specific content format, it facilitates parsing. By abstracting access, it facilitates cross-platform data exchange.

RSA-MD5

RSA is a public-key cryptosystem for both encryption and authentication, RSA is combined with the MD5 hashing function to sign a message in this signature suite.

Message authentication involves hashing the message to produce a "digest," and encrypting the digest with the private key to produce a digital signature. Thereafter anyone can verify this signature by

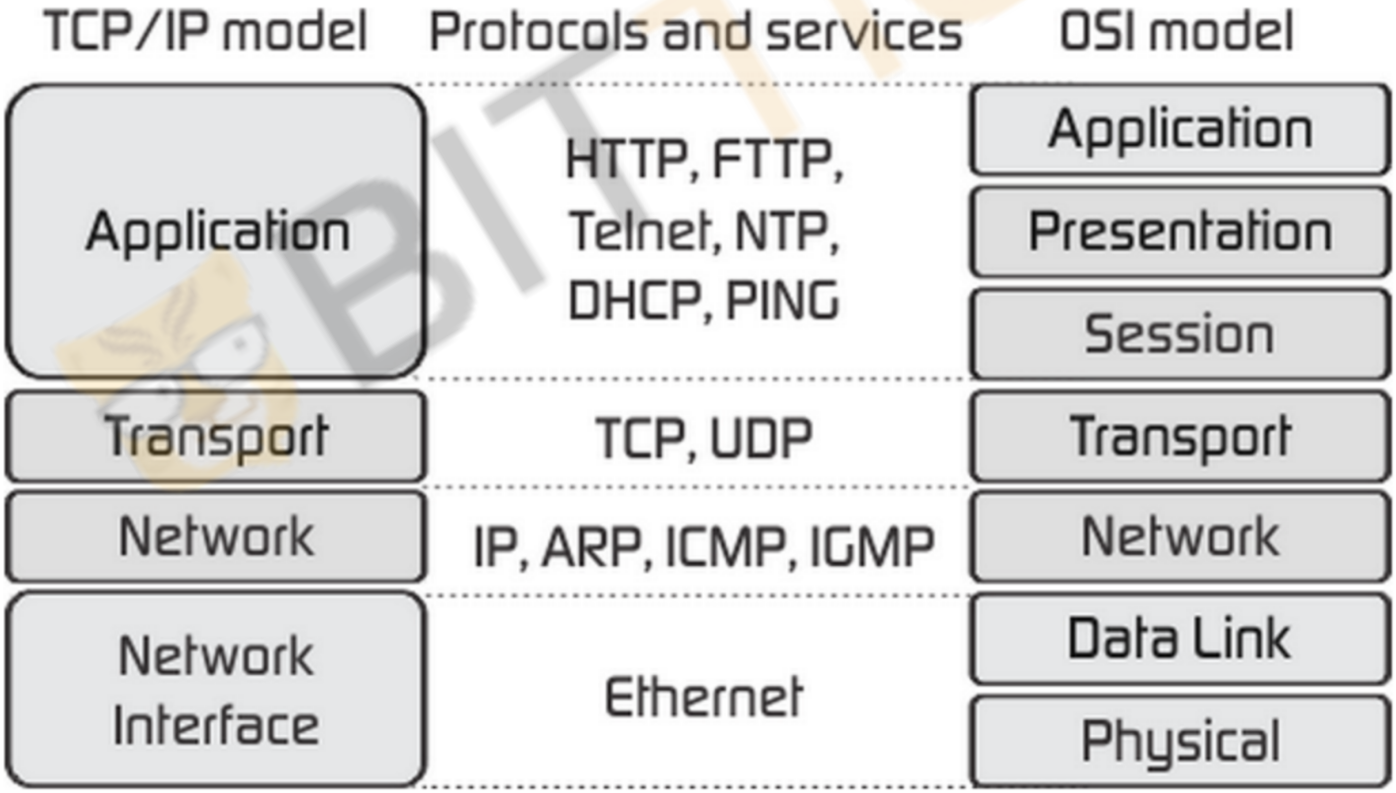
(1) computing the hash of the message,

(2) decrypting the signature with the signer's public key, and

(3) comparing the computed digest with the decrypted digest.

Equality between the digests confirms the message is unmodified since it was signed, and that the signer, and no one else, intentionally performed the signature operation — presuming the signer's private key has remained secret. The security of such procedure depends on a hash algorithm of such quality that it is computationally impossible to alter or find a substitute message that produces the same digest - but studies have shown that even with the MD5 and SHA-1 algorithms, producing an altered or substitute message is not impossible. The current hashing standard for encryption is SHA-2. The message itself can also be used in place of the digest.

OSI and TCP/IP

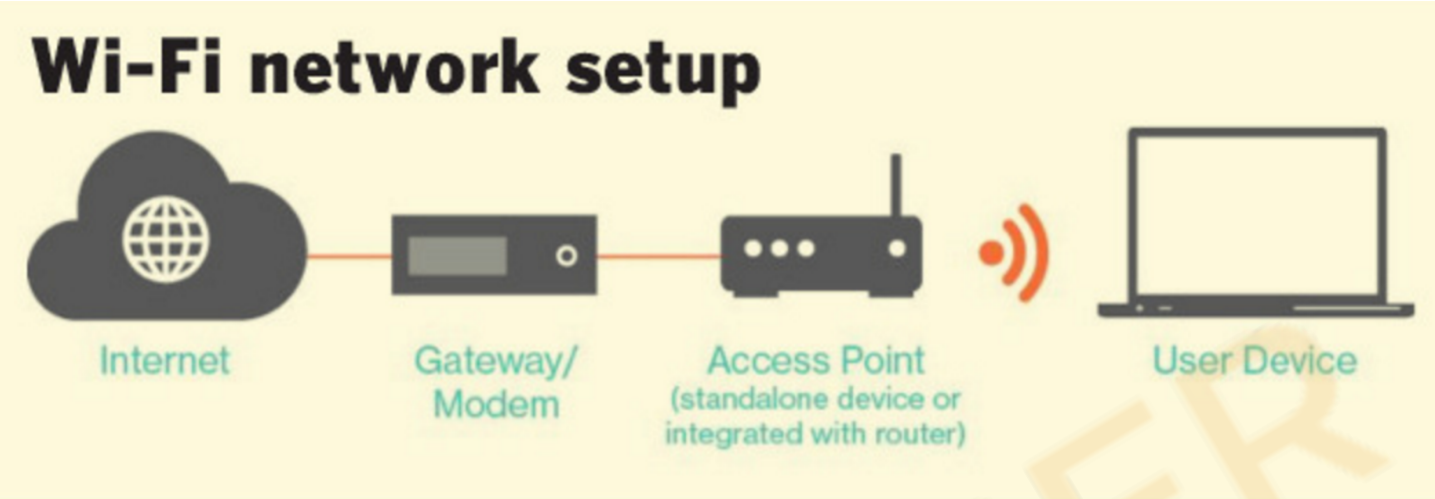


Is it safe to use public WIFI networks?

Wi-Fi is a type of wireless local area network (WLAN) technology that enables an electronic device such as a

laptop or smartphone to exchange data or connect to the Internet using radio waves. The core technology behind Wi-Fi is a device called an access point, which acts like a bridge between the wired network and the Wi-Fi network. The access point, in turn, typically connects to the Internet via a network router.

http://www.networkworld.com/article/2904439/wi-fi/is-it-safe-to-use-public-wi-fi-networks.html



database

设计数据库

```
class User {
private int userID;                    //primary key
private char[] username;               //char[10]
private char[] password;               //char[10], hidden
int state;
}

class UserTable {
private vector<User> table;
public … insert(…);
public … delete(…);
public … update(…);
public … select(…);
}
```

```
class User{
   private:
      int userID; //Primary Key
      char name[10];
      char hiddenPassword[10];
      int state;
      vector<Session> sessionList;
};
```

```
class Session{
   private:
      int sessionID;
      int userID;
      int deviceCode;
      time_t timeOut;
};
```

userID (Primary key)	name	hiddenPassword	state
4	Sangpo	A1V2F2G2F1	1
3	Steve	F4H1G1H7G1	2
0	Jobs	F1G4J5H1K1	3
21	Killer	M2J2J3N4M1	1
19	Me	M3J1B3N2N1	1

sessionID (Primary key)	userID (Foreign key)	deviceCode	timeOut
1010111101	4	1	22:12:13 15 03 2015
1010101010	4	3	10:12:33 16 03 2015
0101010001	4	1	12:22:23 16 03 2015
1101001010	0	1	17:21:05 16 03 2015
1101001001	19	1	12:05:51 16 03 2015
1000101001	19	2	14:33:33 16 03 2015

Challenge of lookup

- How to find the name of the user whose userID = 21?

- Algorithm
  - Check 4 == 21 (No)
  - Check 3 == 21 (No)
  - Check 0 == 21 (No)
  - Check 21== 21 (Yes)

  - Time Complexity = O(n)

userID	name	hiddenPassword	state
4	Sangpo	A1V2F2G2F1	1
3	Steve	F4H1G1H7G1	2
0	Jobs	F1G4J5H1K1	3
21	Killer	M2J2J3N4M1	1
19	Me	M3J1B3N2N1	1

Index with hash

```
class UserTable{
   private:
      vector<Row*> table;
      Hash index;
   public:
      ... Insert(...);
      ... Delete(...);
      ... Update(...);
      ... Select(...);
};
```

hashID	userID	pointer
0	21	
1	4	
2	3	
3	-	
4	0	
5	-	
6	19	

Lookup with hash

hashID	userID	pointer
0	21	
1	4	
2	3	
3	-	
4	0	
5	-	
6	19	

userID	name	hiddenPassword	state
4	Sangpo	IFFDIFSLAF	1
3	Steve	JIJEFBIAFN	2
0	Jobs	DSFSDIENFA	3
21	Killer	FIDNFIANFD	1
19	Me	FDIAANFIDD	1

Time Complexity: O(1)

Challenge of range query

- How to find the users whose userIDs are within [4,20]?

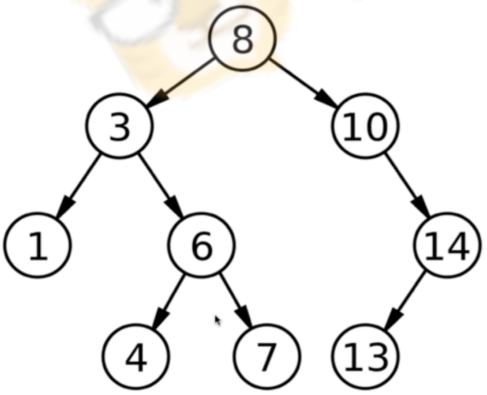
- Algorithm
  - Check 4 in [4,20] (Yes)
  - Check 3 in [4,20] (No)
  - Check 0 in [4,20] (No)
  - Check 21 in [4,20] (No)
  - Check 19 in [4,20] (Yes)

  - Time Complexity = O(n)

hash	userID	Pointer
0	21	
1	4	
2	3	
3	-	
4	0	
5	-	
6	19	

userID	...
4	...
3	...
0	...
21	...
19	...

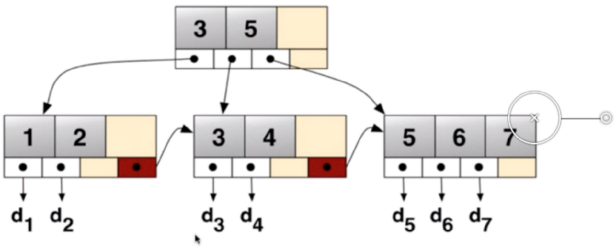
Index with Binary Search Tree



- A node's
  - Left sub-tree is smaller
  - Right sub-tree is larger

- Complexity
  - Space: O(n)
  - Time of insert: $O(\log_2 n)$
  - Time of delete: $O(\log_2 n)$
  - Time of lookup: $O(\log_2 n)$
  - Time of range query: $O(\log_2 n+k)$

Index with B+Tree



- Complexity
  - Space: O(n)
  - Time of insert: $O(\log_b n)$
  - Time of delete: $O(\log_b n)$
  - Time of lookup: $O(\log_b n)$
  - Time of range query: $O(\log_b n+k)$
- Advantage: disk friendly with continuous reading

Data - Membership

userID	money	endTime
3	20	16_07_2016

```
class Membership{
  private:
    int userID;
    double money;
    time_t endTime;
  public:
    ... addMoney(...);
    ... buyMember(...);
};
```

One more step: ACID

- Q5.1: Atomicity, all or nothing
- Q5.2: Consistency, valid according to all defined rules
- Q5.3: Isolation, a kind of independency between transactions
- Q5.4: Durability, stored permanently

Database
SQL stands for Structured Query Language.

To build a web site that shows data from a database, you will need:
An RDBMS database program (i.e. MS Access, SQL Server, MySQL)
To use a server-side scripting language, like PHP or ASP
To use SQL to get the data you want

To use HTML / CSS

RDBMS
RDBMS stands for Relational Database Management System.
RDBMS is the basis for SQL, and for all modern database systems such as MS SQL Server, IBM DB2, Oracle, MySQL, and Microsoft Access.
The data in RDBMS is stored in database objects called tables.
A table is a collection of related data entries and it consists of columns and rows.

A database most often contains one or more tables. Each table is identified by a name (e.g. "Customers" or "Orders"). Tables contain records (rows) with data.
SQL is NOT case sensitive: select is the same as SELECT

SELECT - extracts data from a database
UPDATE - updates data in a database
DELETE - deletes data from a database
INSERT INTO - inserts new data into a database
CREATE DATABASE - creates a new database
ALTER DATABASE - modifies a database
CREATE TABLE - creates a new table
ALTER TABLE - modifies a table
DROP TABLE - deletes a table
CREATE INDEX - creates an index (search key)
DROP INDEX - deletes an index

SELECT

The SELECT statement is used to select data from a database.
The result is stored in a result table, called the result-set.

```
SELECT column_name,column_name FROM table_name;
SELECT * FROM table_name; (selects all the columns from the table)
```

SELECT DISTINCT

The SELECT DISTINCT statement is used to return only distinct (different) values.

```
SELECT DISTINCT column_name,column_name FROM table_name;
```

WHERE

The WHERE clause is used to filter records. The WHERE clause is used to extract only those records that fulfill a specified criterion.

```
SELECT column_name,column_name
FROM table_name
WHERE column_name operator value;
```

WHERE Country = ‘Mexico’ or WHERE CustomerID = 1
Operators:

=	Equal
<>	Not equal. Note: In some versions of SQL this operator may be written as !=
>	Greater than
<	Less than
>=	Greater than or equal
<=	Less than or equal
BETWEEN	Between an inclusive range

LIKE Search for a pattern
IN To specify multiple possible values for a column

AND & OR

The AND & OR operators are used to filter records based on more than one condition.

```
SELECT * FROM Customers
WHERE Country='Germany'
AND (City='Berlin' OR City='München');
```

ORDER BY

The ORDER BY keyword is used to sort the result-set by one or more columns.

The ORDER BY keyword sorts the records in ascending order by default. To sort the records in a descending order, you can use the DESC keyword.

```
SELECT column_name, column_name
FROM table_name
ORDER BY column_name ASC|DESC, column_name ASC|DESC;
```

INSERT

he CustomerID column is automatically updated with a unique number for each record in the table.

```
INSERT INTO table_name
VALUES (value1,value2,value3,...);

INSERT INTO table_name (column1,column2,column3,...)
VALUES (value1,value2,value3,...);
```

UPDATE

WARNING! The WHERE clause specifies which record or records that should be updated. If you omit the WHERE clause, all records will be updated!

```
UPDATE table_name
SET column1=value1,column2=value2,...
WHERE some_column=some_value;
```

DELETE

WARNING! The WHERE clause specifies which record or records that should be deleted. If you omit the WHERE clause, all records will be deleted!

```
DELETE FROM table_name
WHERE some_column=some_value;
```

It is possible to delete all rows in a table without deleting the table. This means that the table structure, attributes, and indexes will be intact:

WARNING! Be very careful when deleting records. You cannot undo this statement!

```
DELETE FROM table_name;
DELETE * FROM table_name;
```

```
uName = getRequestString("UserName");
uPass = getRequestString("UserPass");
sql = "SELECT * FROM Users WHERE Name ='" + uName + "' AND Pass ='" + uPass + "'"
```

if someone put in " or ""=" into the user name or password text box.

It will becomes `SELECT * FROM Users WHERE Name ="" or ""="" AND Pass ="" or ""=""`

WHERE ""="" is always true.

Most databases support batched SQL statement, separated by semicolon.

```
SELECT * FROM Users; DROP TABLE Suppliers
```

The SQL above will return all rows in the Users table, and then delete the table called Suppliers.

The only proven way to protect a web site from SQL injection attacks, is to use SQL parameters.

```
txtUserId = getRequestString("UserId");
txtSQL = "SELECT * FROM Users WHERE UserId = @0";
db.Execute(txtSQL,txtUserId);
```

SELECT TOP

The SELECT TOP clause is used to specify the number of records to return.

```
SELECT TOP number|percent column_name(s)
FROM table_name;
SELECT TOP 2 * FROM Customers;
```

IN MYSQL:

```
SELECT column_name(s)
FROM table_name
LIMIT number;
```

AS (Aliases)

SQL aliases are used to give a database table, or a column in a table, a temporary name.

Basically aliases are created to make column names more readable.

```
SELECT column_name AS alias_name
FROM table_name;
SELECT column_name(s)
FROM table_name AS alias_name;
```

Tip: It requires double quotation marks or square brackets if the column name contains spaces:

```
SELECT CustomerName AS Customer, ContactName AS [Contact Person]
FROM Customers;
SELECT CustomerName, Address+', '+City+', '+PostalCode+', '+Country AS Address
FROM Customers;
```

In MySQL:

```
SELECT CustomerName, CONCAT(Address,', ',City,', ',PostalCode,', ',Country) AS
Address
FROM Customers;
```

Aliases can be useful when:

There are more than one table involved in a query

Functions are used in the query

Column names are big or not very readable

Two or more columns are combined together

JOIN

An SQL JOIN clause is used to combine rows from two or more tables, based on a common field between them.

The most common type of join is: SQL INNER JOIN (simple join). An SQL INNER JOIN returns all rows from multiple tables where the join condition is met.

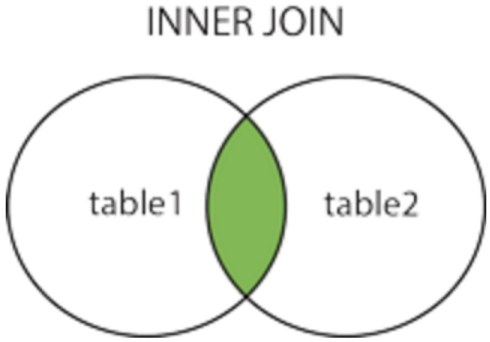
```
SELECT Orders.OrderID, Customers.CustomerName, Orders.OrderDate
FROM Orders
```

```
INNER JOIN Customers
ON Orders.CustomerID=Customers.CustomerID;
```

Types:

INNER JOIN: Returns all rows when there is at least one match in BOTH tables
LEFT JOIN: Return all rows from the left table, and the matched rows from the right table
RIGHT JOIN: Return all rows from the right table, and the matched rows from the left table
FULL JOIN: Return all rows when there is a match in ONE of the tables

INNER JOIN

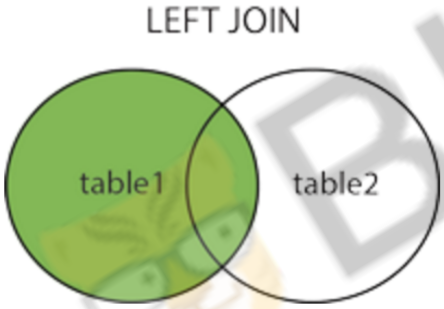


The INNER JOIN keyword selects all rows from both tables as long as there is a match between the columns in both tables.

```
SELECT column_name(s)
FROM table1
INNER JOIN table2
ON table1.column_name=table2.column_name;
```

INNER JOIN is the same as JOIN.

LEFT JOIN



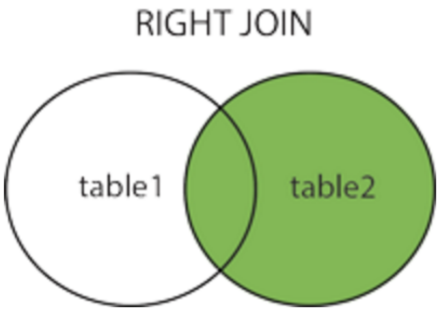
The LEFT JOIN keyword returns all rows from the left table (table1), with the matching rows in the right table (table2). The result is NULL in the right side when there is no match.

The LEFT JOIN keyword returns all the rows from the left table (Customers), even if there are no matches in the right table (Orders).

```
SELECT column_name(s)
FROM table1
LEFT JOIN table2
ON table1.column_name=table2.column_name;
```

In some databases LEFT JOIN is called LEFT OUTER JOIN.

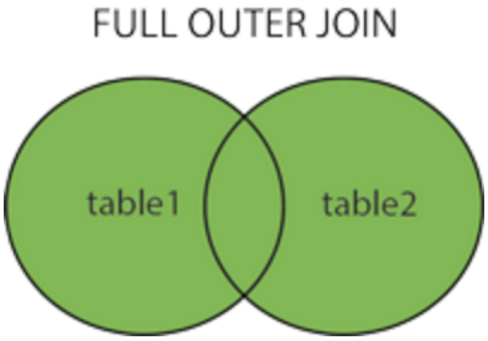
RIGHT JOIN



The RIGHT JOIN keyword returns all rows from the right table (table2), with the matching rows in the left table (table1). The result is NULL in the left side when there is no match.

```
SELECT column_name(s)
FROM table1
RIGHT JOIN table2
ON table1.column_name=table2.column_name;
```

FULL JOIN



The FULL OUTER JOIN keyword returns all rows from the left table (table1) and from the right table (table2).
The FULL OUTER JOIN keyword combines the result of both LEFT and RIGHT joins.
The FULL OUTER JOIN keyword returns all the rows from the left table (Customers), and all the rows from the right table (Orders). If there are rows in "Customers" that do not have matches in "Orders", or if there are rows in "Orders" that do not have matches in "Customers", those rows will be listed as well.