

1. What are maven repositories?

1. Local repositories: It is located in your operating system.
2. Central repositories: It is managed by the Maven community. It contains a lot of common libraries.
3. Remote repositories: Apart from Maven repositories, some companies have their own repositories which contain their private libraries.

2. Maven life cycle?

1. Validate: validate the project is correct and all necessary information is available.
2. Compile: .java - .class
3. Test: Do unit test.
4. Package: package code to some format like jar.
5. Verify: Run checks on results of integration tests.
6. Install: install package to local repository.
7. Deploy: Copy the final package to remote repository.

3. Maven command?

1. mvn clean
2. mvn test
3. mvn install

4. Git basic Operation?

1. git add .
2. git commit
3. git push

5. Git working flow?

1. Create a public repository
2. Pull it to your local repository
3. Do modification
4. Add
5. Commit and add some comment
6. Push it to the remote repository

6. Java primitive type?

1. byte: 1 byte
2. Char: 2 bytes
3. Int: 4 bytes
4. long: 8 bytes
5. short: 2 bytes
6. float: 4 bytes
7. double: 8 bytes
8. boolean: 1 byte

VARIABLE

Oliver Li

- Primitive Data Types

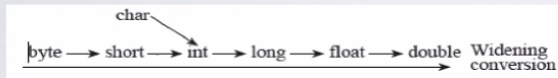
Data Type	Default Value	Size	Range
byte	0	8	-128 to 127 (inclusive)
short	0	16	-32,768 to 32,767 (inclusive)
int	0	32	-2,147,483,648 to 2,147,483,647 (inclusive)
long	0L	64	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807 (inclusive)
float	0.0F	32	1.401298464324817e-45f to 3.402823476638528860e+38f
double	0.0D	64	4.94065645841246544e-324 to 1.79769313486231570e+308
char	'\u0000'	16	0 to 65535
boolean	false	Not defined	true or false

CONVERSION AND CASTING

Oliver Li

- **Conversion**, also known as widening conversion, are performed automatically

- A smaller box can be placed in a bigger box and so on.



- **Casting** also known as narrowing conversion.

- A bigger box has to be placed in a small box.

- Casting is not implicit in nature.

- `int i = (int)(8.0/3.0);`

- Casting will lose precision



7. Wrapper Class?

Byte, Character, Integer, Long, Short, Float, Double, Boolean

8. What is AutoBoxing and unboxing?

Convert primitive type into corresponding wrapper class.

Unboxing is the reverse process.

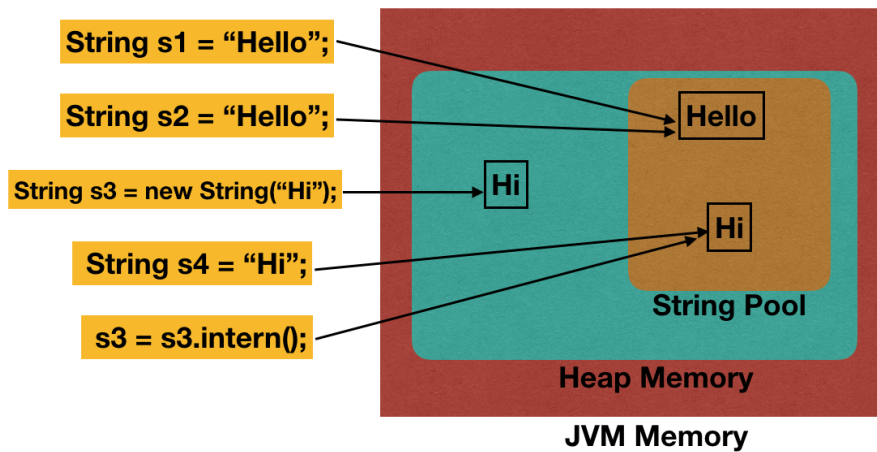
9. What is the difference between String/StringBuilder/StringBuffer?

1. String is an immutable class, while StringBuffer/StringBuilder are mutable .
2. String and StringBuffer are thread-safe.

10. String/Integer constant Pool

heap: new

string pool: "abc"



```
public static void main(String[] args) {  
    Integer a = 12;  
    Integer b = 12;  
    Integer c = new Integer(12);  
    Integer d = Integer.valueOf(12);  
    Integer e = 200;  
    Integer f = 200;  
  
    System.out.println(a == b); // true  
    System.out.println(a == c); // false  
    System.out.println(a == d); // true  
    System.out.println(e == f); // false    from -128 to 127  
}
```

11. Why do we need to override equals and hashCode together?

我来简单的说一下为什么要重写hashCode和equals。

在hashmap中是通过hashCode决定元素放入哪个索引（桶）中，然后通过equals判断和索引中对应链表中的每个元素是否相同，如果有相同 -> 不放，如果不相同 -> 放（jdk version > 8尾插法、< 8头插法）

如果不重写hashCode方法，hash值就是地址值；重写hashCode方法，hash值是根据对象中的成员变量值经过一系列的算法求得。

如果不重写equals方法，调用equals方法默认走地址值；重写equals方法后调用equals是通过成员变量值。

如果在hashmap中如果只重写了hashCode没重写equals，如果我们添加两个相同内容的User对象放入hashmap，添加完第一个User后，添加第二个User时，hash值和第一个User相同，会走equals到同一个索引中找是否存在相同的元素，此时我们因为没重写equals，比较的是地址值（两个虽然内容相同的对象地址值是不同的），因此第二个相同的User也插入到了同一个索引中。

如果在hashmap中如果只重写了equals没重写hashCode

这个最好理解，上面说到：两个虽然内容相同的对象地址值是不同的，因为没重写hashCode，hash值是地址值 ==> 两个相同内容的对象地址值不同，hash值也不同 ==> 他们两个被放到了hashmap不同的索引中。

Hashing retrieval is a two-step process:

1. Find the right bucket using (hashCode).
2. Search the bucket for the right element using (equals).

Default hashCode is generated by the address of an instance.

Default equals compares two instances according to their addresses.

Supposely, we just override hashCode without overriding equals. If we add two instances with the same values in a set. The size of the set will be 2.

Supposely, we just override equals without overriding hashCode. If we add two instances with the same values in a set. The size of the set will be 2.

12. What is the difference between list and set?

List allows duplicates while Set doesn't allow duplicates.

13. What is the difference between ArrayList and LinkedList?

ArrayList is faster in storing and accessing data.	LinkedList is faster in manipulation of data.
----------------------------------------------------	-----------------------------------------------

14. heap is implemented by priority_queue.

15. deque is implemented by ArrayDeque.

16. HashMap vs HashTable vs ConcurrentHashMap

1. HashMap is implemented by array and LinkedList and red-black tree; Before JDK 1.8, it was implemented by array and LinkedList.
2. The default capacity is 16.
3. When the length of LinkedList ≥ 8 . It will be transformed to red-black tree.

Differences:

1. HashTable and ConcurrentHashMap are thread-safe
2. ConcurrentHashMap uses bucket-level lock while HashTable uses object level lock.

17. HashSet vs TreeSet vs LinkedHashSet

LinkedHashSet keeps the insertion order while HashSet doesn't
TreeSet can sort elements using according to comparator.

18. HashMap VS TreeMap VS LinkedHashMap(LRU)

As 17

19. Stack and Queue

Stack is thread-safe.

BlockingQueue offers a simple **thread-safe** mechanism

```
Stack<Integer> st = new Stack<>();  
Deque<Integer> q = new ArrayDeque<>();
```

20.Array

The best way is to use a `List` within a `List`:

```
List<List<String>> listOfLists = new ArrayList<List<String>>();
```

```
int[][] arr = { { 1, 2 }, { 3, 4 } };
```

21 BinaryTree VS Balanced BinaryTree

Bi-Tree: the height of the left and right subtree of any node differ by not more than 1

22.Comparator VS Comparable

1. When we use a comparable interface, we need to override the `compareTo` method. It is used to define the class internal order.
2. When we use a comparator interface, we need to override the `compare` method. Comparator is used for sorting. And it is used by other classes like `TreeSet`, `priority_queue`.

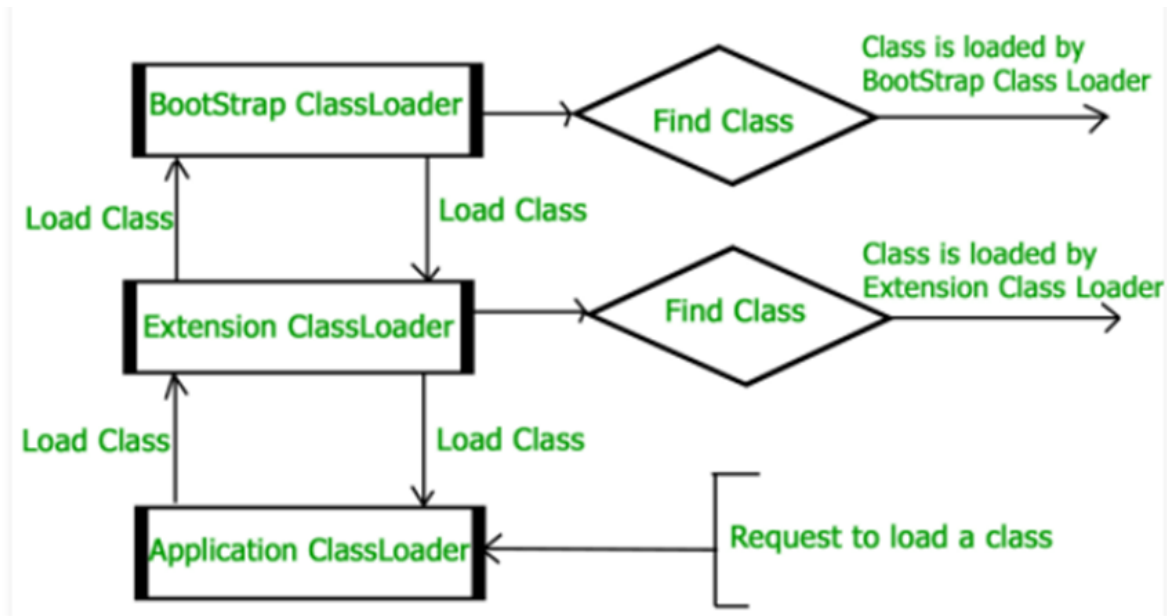
23 What does the class loading subsystem have?

It has three phases: Loading, linking , initialization

1. Loading phase has a bootstrap class loader, extension class loader, application class loader.
2. Linking phase has three steps. Verify, prepare and resolve.
Verify: check if it is a java binary code.
Prepare: Memory is allocated for static variables inside a class. And these variables are set to default value.
Resolve: replace symbolic references with direct references. Class not found exceptions happens in this step.

3. Initialization phase: initialize static variables in class. (static block)

24. How does the classloader load classes? What flow should we follow?



It is a recursive process.

25. What does the run-time area have?

Stack: It is local to each thread. And it stores local variables. And returns the addresses during method calls.

Heap: It is shared by thread. Objects are allocated here.

Method Area: It stores code, the structure of class, constant pool. It is shared by threads.

PC register: It is independent for each thread. It points to the addresses of JVM instructions which are currently running.

native method stack: it is used for non-java methods.

26.What does the Execution engine have?

Interpreter: convert java binary codes into machine codes.

JIT(just in-time) compiler: Help to improve performance of java program by compiling binary code into machine code at run-time.

Garbage Collector: release the memory space for those unreferenced objects.

Native method Interface: It is used to call native methods.

Native method libraries: it is implemented by native codes(C/C++).

27.What is the native keyword in java?

The native keyword in Java is applied **to a method to indicate that the method is implemented in native code**

28.what types does garbage collection have?

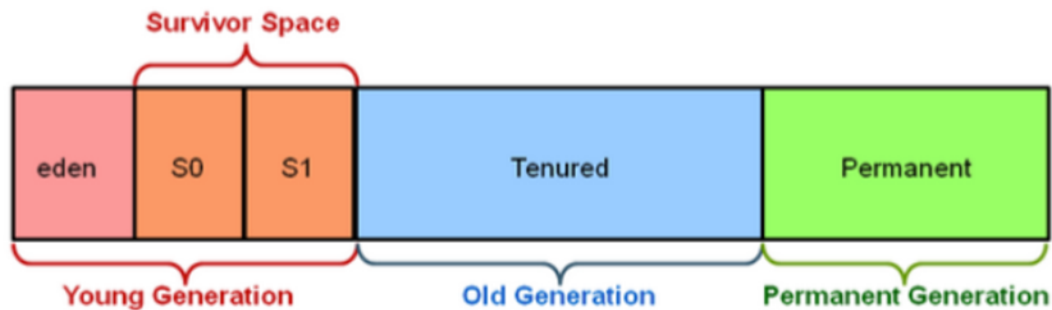
1. serial GC:
2. parallel GC:
3. G1 GC: Divide heap into equal sized chunks and rank these chunks according to the number of garbage they have.
4. CMS GC -> removed from java 14

29.What is the GC process?

Gc has three process:

1. Young generation:
2. Old generation
3. permanent generation:**Metadata such as classes and methods are stored** in the Permanent Generation.

Hotspot Heap Structure



New objects will be put in the Eden space. When the Eden space is full, all unreferenced objects will be removed from memory and referenced objects will be put into S0 space. Unreferenced objects will be removed while referenced objects will be put into S1. Next time the S1 and eden area are full. The unreferenced objects will be removed from eden and S1 while referenced objects will be put into S0. When the object lives a long time, minor gc will be triggered. These objects will be collected by Tenured space.

- **Major GC** is cleaning the Tenured space.
- **Full GC** is cleaning the entire Heap – both Young and Tenured spaces.

30. What is a stop-the-world event?

All minor garbage collections are "Stop the World" events. This means that all application threads are stopped until the operation completes.

31.access modifier

Modifier	Class	Package	Subclass	Global
Public	Yes	Yes	Yes	Yes
Protected	Yes	Yes	Yes	No
Default	Yes	Yes	No	No
Private	Yes	No	No	No

	default	private	protected	public
Same Class	Yes	Yes	Yes	Yes
Same package subclass	Yes	No	Yes	Yes
Same package non-subclass	Yes	No	Yes	Yes
Different package subclass	No	No	Yes	Yes
Different package non-subclass	No	No	No	Yes

32.What is native?

Native indicates that method is implemented by native code(C/C++)

33.What is static?

It means the member is belong to its type and not belongs to an instance

34. What is the synchronized keyword?

Synchronized blocks in Java are marked with the **synchronized keyword**

35. Strictfp

It means that floating points operation gives the same results in any platform.

36.transient

The transient keyword in Java is **used to avoid serialization**.

37. volatile keyword

The value marked by volatile keyword will be read directly from the main memory.

```
class Test
{
    static volatile int var =5;
}
```

38. Final vs finally vs finalize

Final indicates the variable is immutable .

Finally is the keyword in the try catch statement. The codes in finally block will be executed even though there are exceptions in the try block.

Finalize is the method in class. It is used to release resources. And it is executed before garbage collection process.

```
}
// defining the finalize method
protected void finalize()
{
    System.out.println("Called the finalize() method");
}
```

39.throw vs throws

Throw is used inside of a method.

Throws is behind the name of a method. The exception is thrown to the caller.

40.immutable class

It means the members in the class cannot be modified.

41. implements vs extends

extends: It is used for subclasses to inherit base-class. A subclass can inherit from only one base-class.

Implements:It is used for class to implement the method in an interface or multi-interfaces.

42.OOP

1. How can we achieve Abstraction: use abstract or interface
2. How do we achieve Encapsulation: declare private variables and use getter/setter methods to modify them.
3. Inheritance:extends and implements
Class can extend one class but can implement multiple interfaces.
4. How do we achieve polymorphism
override
overload

43. checked Exception vs Unchecked exception

Checked Exception: These exceptions will be checked during compiling process IO Exception, SQL exception, class not found.

Unchecked Exception: Compiler does not check them.Eg Null Pointer exception ,IndexOutOfBoundsException

44. Exception VS Error

Error represent serious problems which should be removed before compiling and executing. For example , JVM error -> stackoverflow and outofmemory. Exceptions can be handled by programs.

45.How to handle exceptions?

Try catch

46.How to customize exceptions?

We can define an exception by inheriting an Exception `class`.

47.How to handle multiple exceptions?

The order of handling multiple exceptions is from specific to general. That is, Catch for arithmetic exception must come before catch for Exception.

48.What is try with resources?


All the formal parameters in try method must implement the autocloseable interface and override the close method.

```
try{
    int a[]=new int[5];
    a[5]=30/0;
}
catch(ArithmeticException e)
{
    System.out.println("Arithmetic Exception occurs");
}
catch(ArrayIndexOutOfBoundsException e)
{
    System.out.println("ArrayIndexOutOfBoundsException occurs");
}
catch(Exception e)
{
    System.out.println("Parent Exception occurs");
}
System.out.println("rest of the code");
```

49. What is generics? Why do we use?

Generics mean **parameterized types**. Therefore, we can create a class which works with different data types.

50. How to use generics?



```
// A Simple Java program to show working of user defined
// Generic classes

// We use < > to specify Parameter type
class Test<T>
{
    // An object of type T is declared
    T obj;
    Test(T obj) { this.obj = obj; } // constructor
    public T getObject() { return this.obj; }
}

// A Generic method example
static <T> void genericDisplay (T element)
{
    System.out.println(element.getClass().getName() +
                        " = " + element);
}
```

We use <T,E> to specify the parameter types.

51. Adv and disadvantage of generics?

Adv:

1. To achieve code reusability.
2. Type Safety: We can hold only a single type of objects in generics
3. To avoid ClassCastException.
4. It is checked at compile time so problems will not occur at runtime.

Dis:

1. We cannot initialize generic types with primitive type .
2. We cannot create, catch, or throw **generic** types.
3. Due to type erasure, you cannot use *instanceof methods* with generic types.

52. Difference VS ?

<? extends E>: It requires the generic type is E or its subclass. There is an arraylist which type is ? extends E. We cannot add elements to the list. It is used in scenarios which have more read operation than write.

<? super T>: It requires the generic type is E or its base-classes. There is an arraylist which type is ? extends E. We can only get elements from the list with an object. It is used in scenarios which have more write operation than read.

53. What is type erasure?

Type erasure is a **process in which compiler replaces a generic parameter with Object**
Generate bridge methods to keep polymorphism in extended generic types.. (compile time)

Type erasure ensures that no new classes are created for parameterized types; consequently, generics incur no runtime overhead.

54. Bridge Method?

These are **methods that create an intermediate layer between the source and the target functions.**

```
public class MyComparator implements Comparator<Integer> {  
    public int compare(Integer a, Integer b) {  
        //  
    }  
  
    //THIS is a "bridge method"  
    public int compare(Object a, Object b) {  
        return compare((Integer)a, (Integer)b);  
    }  
}
```

55. [Why primitive datatypes are not allowed in java.util.ArrayList?](#)

That is because type erasure.

56. Difference between input/output Streams and Readers/Writers

The major difference between these is that the input/output stream classes read/write byte stream data. Whereas the Reader/Writer classes handle characters.

57. Byte Streams vs Character Streams

Byte streams: These handle data in bytes (8 bits) (FileInputStream / ObjectInputStream)
Using these you can store characters, videos, audios, images etc. All byte stream classes are descended from InputStream and OutputStream.

Character Stream: These handle data in 2 bytes. (FileReader/FileWriter)
Using these you can read and write text data only. All character stream classes are descended from Reader and Writer.

58. What is File

Give you access to underlying file systems.

59. How do we do serialization and deserialization?

We use `objectinputstream` and `objectoutputstream`. For each serialization class, we have a unique ID. If you want to serialize some classes, these classes need to implement `Serializable` interface.

60. What is serialization and deserialization?

serialization : **converting an object into a stream of bytes to store the object**
Deserialization is the reverse process.

61.transient

The transient keyword in Java is **used to avoid serialization**.

62.Is lambda thread-safe?

All the objects created by lambda are immutable objects so it is thread-safe.

63.Difference between abstract class VS interface.

Type of variables:: The interface has only static and final variables.

Implementation:Interface can't provide the implementation of an abstract class.

Abstract class is inherited using extends keyword

Interface is implemented using implement keyword

64. What is a functional interface?

A functional interface is **an interface that contains only one abstract method**.

65. Default method in interface.





It allows the interfaces to have methods with implementation without affecting the classes that implement the interface.

The reason why the Java 8 release included *default* methods is pretty obvious.

In a typical design based on abstractions, where an interface has one or multiple implementations, if one or more methods are added to the interface, all the implementations will be forced to implement them too. Otherwise, the design will just break down.

Default interface methods are an efficient way to deal with this issue. They **allow us to add new methods to an interface that are automatically available in the implementations**. Therefore, we don't need to modify the implementing classes.

In this way, **backward compatibility is neatly preserved** without having to refactor the implementers.

 // A simple program to Test Interface default
 // methods in java
 **interface** TestInterface
 {
 // abstract method
public void square(**int** a);

 // default method
default void show()
 {
 System.out.println("Default Method Executed");
 }
}

class TestClass **implements** TestInterface
{
 // implementation of square abstract method
public void square(**int** a)
 {
 System.out.println(a*a);
 }

public static void main(String args[])
 {
 TestClass d = **new** TestClass();
 d.square(4);

 // default method executed
 d.show();
 }
}

66. **predefine functional interface**

Consumer
 Predicate
 Supplier
 Function

Predicate

- public Boolean test(T t);

Function

- public R apply(T t);

Consumer

- public void accept(T t);

Supplier

- public R get();

```
Supplier<Double> generateRandomNumber = () -> Math.random();  
System.out.println(generateRandomNumber.get());
```

67. Optional class

It is used to solve null pointer exceptions.

Of method: is used to create an instance of optional. If the input parameter is null, then throw `NullPointerException`

```
Student st1 = new Student();  
Optional<Student> op = Optional.of(st1);
```

ofNullable method: is used to create an instance of optional. If the input parameter is null, then return a null optional instance.

```
Optional opNull = Optional.ofNullable(null);
```

orElse: If the instance has value then return the instance, otherwise return predefined message.

如果Optional实例有值则将其返回，否则返回orElse方法传入的参数。示例如下：

```
1 //如果值不为null, orElse方法返回Optional实例的值。
2 //如果为null, 返回传入的消息。
3 //输出: There is no value present!
4 System.out.println(empty.orElse("There is no value present!"));
5 //输出: Sanaula
6 System.out.println(name.orElse("There is some value!"));
```

orElseThrow: if the instance has value then return the value, otherwise throw an exception.

```
public class JavaOptional {
    public static void main(String[] args) {
        String str = null;

        if (str == null) {
            System.out.println("nothing here");
        } else {
            System.out.println(str);
        }

        Optional<String> opt = Optional.ofNullable(str);
        System.out.println(opt.orElse( other: "nothing here"));
    }
}
```

67.Two kinds of stream operations.

-Intermediate operation:

-terminal operation:

68. Difference between map and flatmap

Map: produce a stream of value.

Supposedly, there is a list stream.

When you pass one function to map, the function will be applied to each element in the list to generate a new value for each element.

FlatMap: produce a stream of stream value.

When you pass one function to map, the function will be applied to each element in the list to generate a new stream for each element. And concatenate all streams into one stream.

```
List<String> list = Arrays.asList("a,b,c", "1,2,3");
Stream<String> s1 = list.stream().map(s ->
s.replaceAll(",", ""));
s1.forEach(System.out::println);
```

```
Stream<String> s2 = list.stream().flatMap( s ->
{
    String[] splits = s.split(",");
    Stream<String> s3 = Arrays.stream(splits);
    return s3;
});
s2.forEach(System.out::println);
```




```
// Driver code
public static void main(String[] args)
{
    // Creating a list of Prime Numbers
    List<Integer> PrimeNumbers = Arrays.asList(5, 7, 11,13);

    // Creating a list of Odd Numbers
    List<Integer> OddNumbers = Arrays.asList(1, 3, 5);

    // Creating a list of Even Numbers
    List<Integer> EvenNumbers = Arrays.asList(2, 4, 6, 8);

    List<List<Integer>> listOfListofInts =
        Arrays.asList(PrimeNumbers, OddNumbers, EvenNumbers);

    System.out.println("The Structure before flattening is : " +
        listOfListofInts);

    // Using flatMap for transforming and flattening.
    List<Integer> listofInts = listOfListofInts.stream()
        .flatMap(list -> list.stream())
        .collect(Collectors.toList());

    System.out.println("The Structure after flattening is : " +
        listofInts);
}
```

Output :

```
The Structure before flattening is : [[5, 7, 11, 13], [1, 3, 5], [2, 4, 6, 8]]
The Structure after flattening is : [5, 7, 11, 13, 1, 3, 5, 2, 4, 6, 8]
```

69. Intermediate API VS terminated API

https://blog.csdn.net/y_k_y/article/details/84633001

`sorted()`: 自然排序, 流中元素需实现Comparable接口

`sorted(Comparator com)`: 定制排序, 自定义Comparator排序器

```
1 List<String> list = Arrays.asList("aa", "ff", "dd");
2 //String 类自身已实现Comparable接口
3 list.stream().sorted().forEach(System.out::println); // aa dd ff
4
5 Student s1 = new Student("aa", 10);
6 Student s2 = new Student("bb", 20);
7 Student s3 = new Student("aa", 30);
8 Student s4 = new Student("dd", 40);
9 List<Student> studentList = Arrays.asList(s1, s2, s3, s4);
10
11 // 自定义排序: 先按姓名升序, 姓名相同则按年龄升序
12 studentList.stream().sorted(
13     (o1, o2) -> {
14         if (o1.getName().equals(o2.getName())) {
15             return o1.getAge() - o2.getAge();
16         } else {
17             return o1.getName().compareTo(o2.getName());
18         }
19     }
20 ).forEach(System.out::println);
```

Intermediate API:

filter: 过滤流中的某些元素

limit(n): 获取n个元素

skip(n): 跳过n元素, 配合limit(n)可实现分页

distinct: 通过流中元素的 hashCode() 和 equals() 去除重复元素

Map:

Flatmap:

Sort:

Peek:

terminated API:

forEach:

Collect:

findFirst: 返回流中第一个元素

findAny: 返回流中的任意元素

count: 返回流中元素的总个数
max: 返回流中元素最大值
min: 返回流中元素最小值

```
List<String> list = Arrays.asList("a,b,c", "1,2,3");  
Stream<String> s1 = list.stream().map(s -> s.replaceAll(regex: ",", replacement: ""));  
List<String> list1 = s1.collect(Collectors.toList());  
list1.forEach(System.out::println);
```

70. How to convert list to map

Use toMap method in stream

2) JAVA 8 直接用流的方法:

```
1. @Test  
2. public void convert_list_to_map_with_java8_lambda () {  
3.  
4.     List<Movie> movies = new ArrayList<Movie>();  
5.     movies.add(new Movie(1, "The Shawshank Redemption"));  
6.     movies.add(new Movie(2, "The Godfather"));  
7.  
8.     Map<Integer, Movie> mappedMovies = movies.stream().collect(  
9.         Collectors.toMap(Movie::getRank, (p) -> p));  
10.  
11.     logger.info(mappedMovies);  
12.  
13.     assertTrue(mappedMovies.size() == 2);  
14.     assertEquals("The Shawshank Redemption", mappedMovies.get(1).getDescription());  
15. }
```

71. What is method reference?

Method reference is **used to refer method of functional interface.**

72. Process vs thread

Process has independent memory space(heap method area) and system resources while multi-thread share memory. Thread has an independent stack and PC register.

73. 6 Thread state

New:When an instance of the Thread class is created

Runnable:When the start() method is called on a the new instance of a thread, it enters into a runnable state.

Block: wait for a monitor lock to entry synchronized block or method

Waiting: A thread is waiting for another thread to perform an action.

wait()

join()

park()

Timed_waiting: A thread is waiting for another thread to perform an action for a specific waiting time.

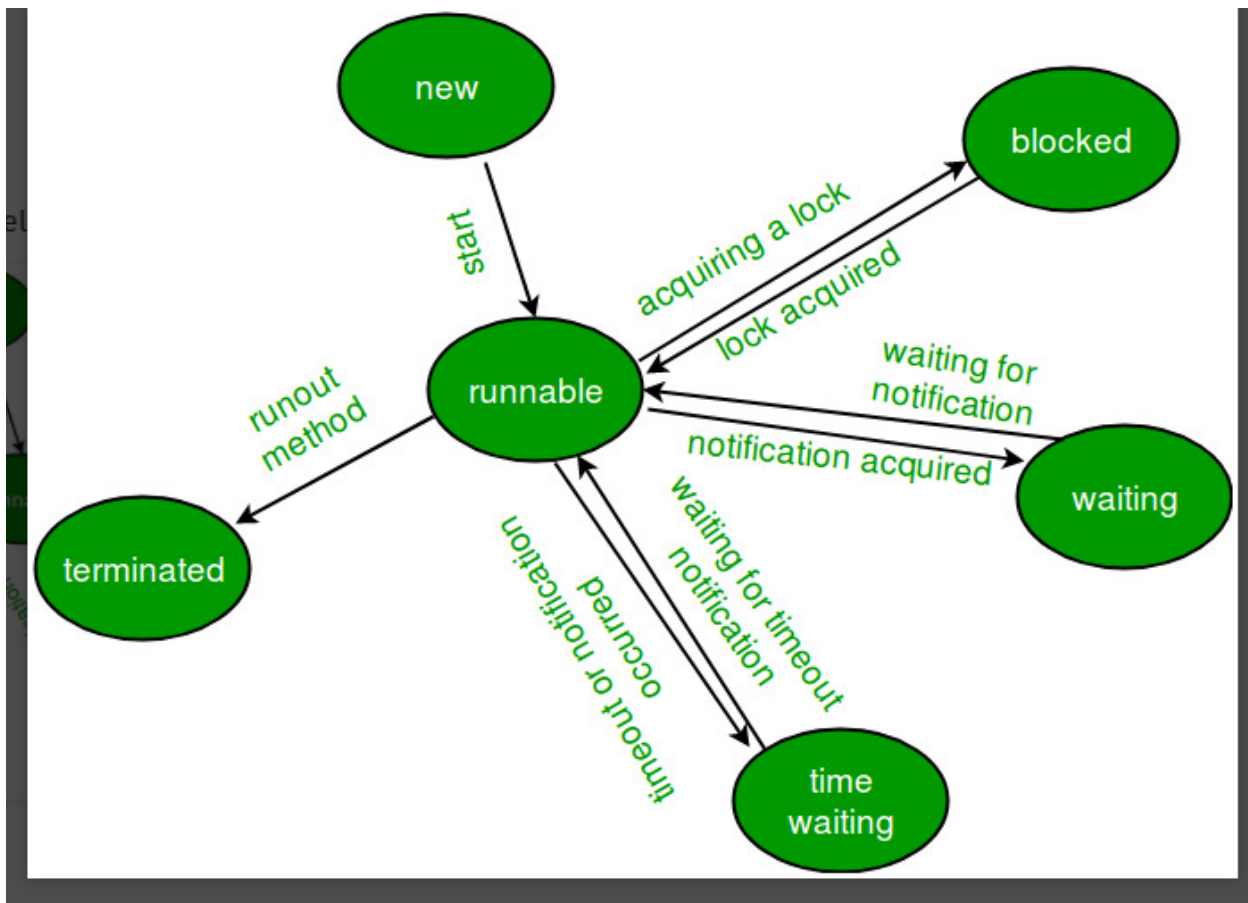
wait(5);

join(6);

park();

Terminated

A thread that has exited is in this state.



74. Four methods to create a thread.

1. Extend thread and override Run method
2. Implement runnable and override Run method
3. Implement callable and override call method (when we need get a value from a thread)
4. Using thread pool and override execute method(improve efficiency)

75.Callable VS Runnable

1. **Callable has call() method but Runnable has run() method.**
2. Callable has call method which returns value but Runnable has run method which doesn't return any value.

3. call method can throw checked exception but run method can't throw checked exception.

76.customized thread pool

```
Creates a new ThreadPoolExecutor with the given initial parameters.
Params: corePoolSize – the number of threads to keep in the pool, even if they are idle, unless
        allowCoreThreadTimeOut is set
        maximumPoolSize – the maximum number of threads to allow in the pool
        keepAliveTime – when the number of threads is greater than the core, this is the maximum
        time that excess idle threads will wait for new tasks before terminating.
        unit – the time unit for the keepAliveTime argument
        workQueue – the queue to use for holding tasks before they are executed. This queue will
        hold only the Runnable tasks submitted by the execute method.
        threadFactory – the factory to use when the executor creates a new thread
        handler – the handler to use when execution is blocked because the thread bounds and
        queue capacities are reached
Throws: IllegalArgumentException – if one of the following holds:
        corePoolSize < 0
        keepAliveTime < 0
        maximumPoolSize <= 0
        maximumPoolSize < corePoolSize
        NullPointerException – if workQueue or threadFactory or handler is null

public ThreadPoolExecutor( @Range(from = 0, to = java.lang.Integer.MAX_VALUE) int corePoolSize,
                           @Range(from = 1, to = java.lang.Integer.MAX_VALUE) int maximumPoolSize,
                           @Range(from = 0, to = java.lang.Long.MAX_VALUE) long keepAliveTime,
                           @NotNull TimeUnit unit,
                           @NotNull BlockingQueue<Runnable> workQueue,
                           @NotNull ThreadFactory threadFactory,
                           @NotNull RejectedExecutionHandler handler) {
```

ThreadPoolExecutor

- **corePoolSize**: the number of threads to keep in the pool
- **maximumPoolSize**: the maximum number of threads to allow in the pool
- **keepAliveTime**:
When the number of threads is greater than the corePool size, that is the maximum time the idle threads will wait before terminating.
- **Unit**: the time unit for keep alive time.
- **workQueue**: the queue used for new tasks before terminating.
- **threadFactory**: the factory used for creating new threads.
- **Handler**: the handler is used to handle exception when the thread bounds and queue capacities are reached. It is used to handle the reject policy.
 - **AbortPolicy**

- **CallerRunPolicy**
- **DiscardPolicy**
- **DiscardOldestPolicy**

core: 2
max: 5
queue 4

77.predefined(inbuild) thread pool

predefined Thread pool

multi

```
ExecutorService threadPool1 = Executors.newFixedThreadPool(5); // core == 5, max 5
ExecutorService threadPool2 = Executors.newSingleThreadExecutor(); // core 1, max 1
```

```
ExecutorService threadPool3 = Executors.newCachedThreadPool(); // core 0, max = Integer.MAX_VALUE;
ExecutorService threadPool4 = Executors.newScheduledThreadPool(3); // Creates a thread pool that can schedule commands to run after a given delay, or to execute periodically
```

78.Future vs completableFuture

<https://www.cnblogs.com/yaochunhui/p/15543298.html>

Future: is used for asynchronous programming. We submit a task to threadpool and we can get a Future instance. It is used to get the result at some point in the future.

when we use future to get asynchronous result, we have two methods:

Calling the block method `get()`.

Polling the check if the `isDone` method returns true or false.

These two methods are not good, because the calling thread will be forced to wait.

`CompletableFuture` improves the `Future`. When the asynchronous task ends, the method of an object will be automatically called back.

```
supplyAsync  
thenAccept  
exceptionally
```

79. Synchronized static method vs synchronized method

static synchronized method will lock the class instead of the object,
Class-level lock VS Object-level lock

80. Lock interface

`ReentrantLock` class implement `Lock` interface

- `lock()`, `unlock()`, `newCondition()`, `tryLock()`, `lockInterruptibly()`...

`ReentrantReadWriteLock` class implement `ReadWriteLock` Interface

```
-ReadLock  
-WriteLock
```

81.Enum

It is used to create some constant values.

```
package day5;  
  
public class EnumTest {  
    public static void main(String[] args) {  
  
        //      System.out.println(EColor.BLUE.ordinal());  
        //      for (EColor val: EColor.values()) {  
        //          System.out.println(val);  
        //      }  
    }  
}
```



```
        System.out.println(EColor.RED.getNum());  
    }  
  
}
```

Ordinal: get the index of that enum

Values: get all the values in that enum

DataBase:

1.database VS DBMS VS SQL

Database management system (DBMS) is a software tool to manage data in a database.

SQL: It is a query language not a database.

2. File System VS DataBase System

Manage file in storage medium/ manage data in database

Redundant data / no redundant data

No efficient query processing/ efficient

Less data consistency / consistency

Less security / security

Less cost/ high cost

3. What is database normalization

Normalization is used to eliminate redundant data and to ensure data is stored logically.

First normal form:

Each table cell should contain a single value.

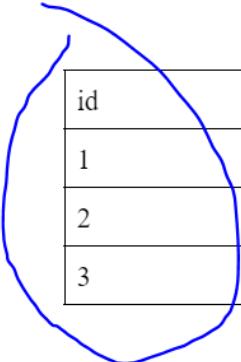
Each tuple should be unique.

| id | phones | address |
|----|------------------|---------|
| 1 | 1234567, 7654321 | |

Second normal form:

-be in 1NF

- have only a single column primary key.



| id | firstName | lastName | phone | deparment |
|----|-----------|----------|-------|-----------|
| 1 | A | B | 123 | Java |
| 2 | A | C | 123 | Java |
| 3 | H | C | | |

Third normal form:

-be in 2NF

-it has no transitive functional dependencies.

| id | Birth date |
|----|------------|
| 1 | 01/01/2001 |
| 2 | 01/01/1995 |
| 3 | 01/01/1990 |

| id | age |
|----|-----|
| 1 | 21 |
| 2 | 27 |
| 3 | 32 |

4.What is major categories of no-sql

1. Document datastore: **mongoDB**
2. Key value:**redis**
3. Graphs :**Neo4j**
4. Columnar:**cassandra**

5.What is CAP

C: consistency: all clients have the same view of the data

A: Availability : each client can always write and read data.

P: partition tolerance: the system work well despite physical network partition

DB in CHINA cannot visit DB in US

Databases can have only two of them.

CP:**MongoDB, Redis**

AP:DynamoDB

6. Sharding VS replica

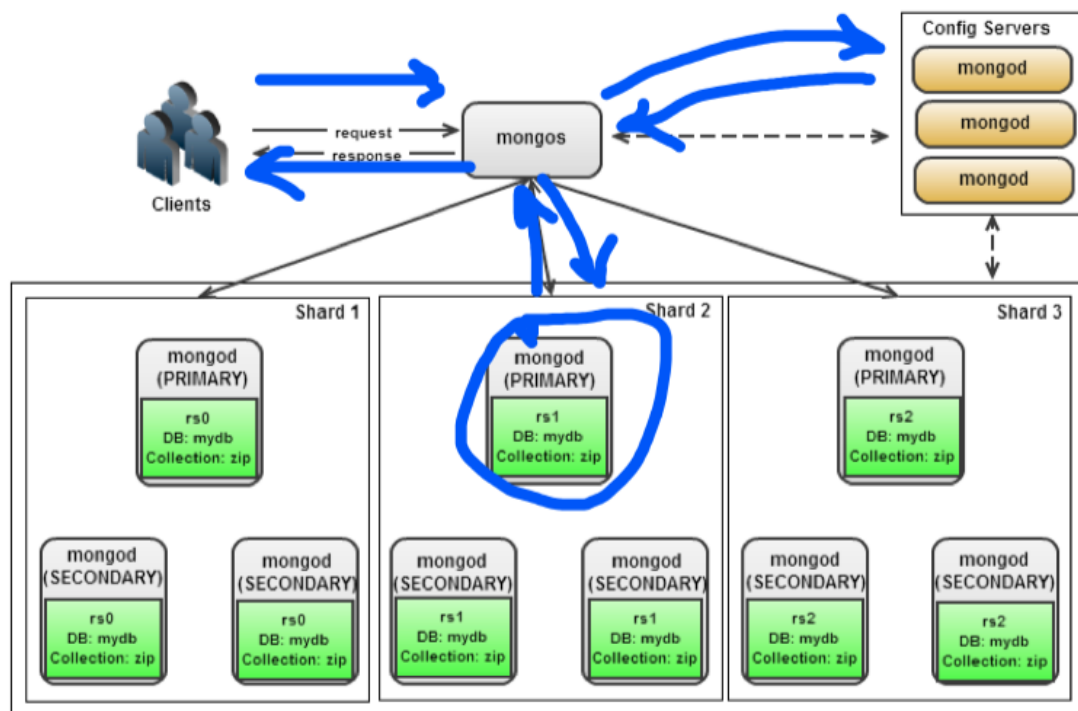
- **Sharding** partitions the data-set into discrete parts.
- **Replication** duplicates the data-set. It means redundancy and offer fault tolerance.

7.MongoDB architecture

-no-sql

-document data store

-written by C++(support API to many langs JAVA PYTHON)



Mongos: work as a router and
accepts all requests and
hit the config server to get the metadata and
Route the request to the corresponding shard.

Mongod : database instance (one primary node and two secondary nodes)

Mongos: database router

Process all requests based on the info from config server and route the request to
corresponding mongods

Mongo: interactive shell

Functionality of MongoDB

- dynamic schema
- follows CP
- built in horizontal scaling(sharding)
- secondary indexes

8.What is redis and how does it use cache?

Redis (remote directory server)

- key-value store
- it is used as a cache
- it is in memory database
- CP
- support different kinds of data structure(String,List,sorted set,hashes)

Cache hit:

If data is in cache, read it from cache.

If not, we read it from DB and write to cache.

9.memachache VS cache

| | Memcached | Redis |
|--------------------------------------------------|-----------|-------|
| Sub-millisecond latency | Yes | Yes |
| Developer ease of use | Yes | Yes |
| Data partitioning | Yes | Yes |
| Support for a broad set of programming languages | Yes | Yes |
| Advanced data structures | - | Yes |
| Multithreaded architecture | Yes | - |
| Snapshots | - | Yes |
| Replication | - | Yes |
| Transactions | - | Yes |
| Pub/Sub | - | Yes |
| Lua scripting | - | Yes |
| Geospatial support | - | Yes |

Redis uses a **single core** and shows better performance than Memcached in storing small datasets when measured in terms of cores. Memcached implements a **multi-threaded architecture** by utilizing multiple cores. Therefore, for storing larger datasets, Memcached can perform better than Redis

ElastiCache cluster support these two engines.

10. Functionality of Redis

Has persistence mechanism

1. Support point-in-time snapshot.
2. AOF(append only file) logs every write operation.

11. Sql VS no-sql

| sql | no-sql |
|----------------------------------------|------------------------------------|
| relational database | non-relational database |
| pre-defined schema | dynamic schema |
| vertical scaling | horizontal scaling |
| ACID | CAP |
| not suited for hierarchical data store | suited for hierarchical data store |

12.What is the index?

indexing is a way to optimize the performance of database by minimizing the number of the disk access

There are two types of index.

Clustered index:

One table has only one clustered index.

It sort according to the primary key.

Non-clustered index:

One table can have many non-clustered indexes

We can create an non-clustered for specified column.

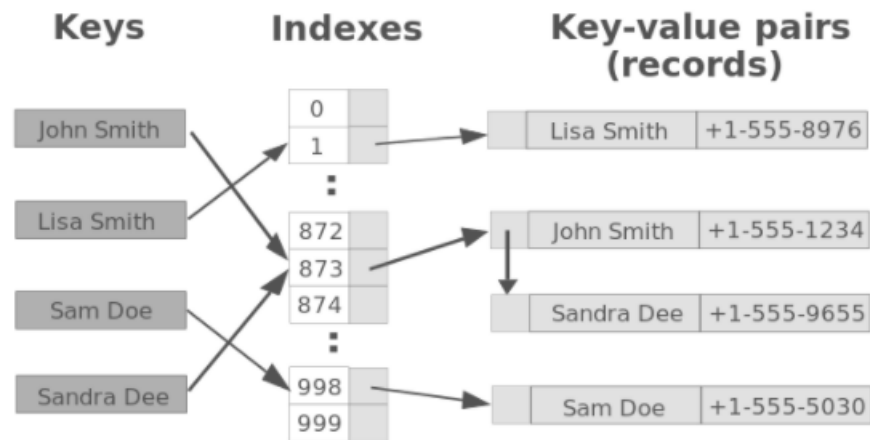
The data structure of non-clustered index :

B+ tree

Hash map

Bit map

...



```
select *
```

```
from table
```

```
where id < 100 and id > 50
```

```
50 - 100 O(n)
```

```
n -> 2n O(n)
```

```
two id of 50
```

```
row id of 100
```

```
O(logn)
```

```
CREATE BITMAP INDEX index_name  
ON table_name(columns);
```

13.View vs stored procedure

A Stored Procedure:

- Accepts parameters
- Can **NOT** be used as building block in a larger query
- Can contain several statements, loops, IF ELSE, etc.
- Can perform modifications to one or several tables
- Can NOT be used as the target of an INSERT, UPDATE or DELETE statement.

A View:

- Does **NOT** accept parameters
- Can be used as building block in a larger query
- Can contain only one single SELECT query
- Can **NOT** perform modifications to any table
- But can (sometimes) be used as the target of an INSERT, UPDATE or DELETE statement.

14. View VS materiali view

A view uses a query to pull data from the underlying tables.

A materialized view is a table on disk that contains the result set of a query.

15.SQL/Application tuning

SQL tuning

1. using **execution plan** to identify the cause of slowness
2. try to reduce joins, remove unused join and join conditions
3. use the index to improve the performance
4. union all instead of union
5. limit
6. view or stored procedure

application tuning

- check the db query - do the sql tuning
- DB connection usage -> connection pool
- do JVM tuning -> Jstack, JMap, JConsole
- server side: CPU, Memory usage by using commands like top, ps
- code review
- check networking, firewall, load balancer

16.What is ACID

Atomicity:

All changes to data are performed as if they are a single operation.

Consistency

Data is in a consistent state when a transaction starts and when it ends.

For example, in an application that transfers funds from one account to another, the consistency property ensures that the total value of funds in both the accounts is the same at the start and end of each transaction.

Isolation

The intermediate state of a transaction is invisible to other transactions.

For example, in an application that transfers funds from one account to another, the isolation property ensures that another transaction sees the transferred funds in one account or the other, but not in both, nor in neither.

Durability

After a transaction successfully completes, changes to data persist and are not undone, even in the event of a system failure.

Transaction: account A -> accountB

read A
A = A - 100;
write A
read B
B = B + 100
write B

A: shouldn't take money from A without giving to B
C: A + B is the same, money isn't lost or gained
I: other queries shouldn't see A or B change until completion
D: the money doesn't go back to A

17. What is dirty read, non-repeatable read, phantom read?

Dirty read: read uncommitted data from another transaction.

Non-repeatable read: read committed data from an update query from another transaction, therefore it causes data inconsistency in the transaction.

phantom read: read committed data from an insert or delete query from another transaction.

18. Which kind of isolation level prevents which kind of problems?

Isolation Level

| Isolation Level | Dirty Reads | Unrepeatable Reads | Phantom Reads |
|------------------|-------------|--------------------|---------------|
| Read uncommitted | Y | Y | Y |
| Read committed | N | Y | Y |
| Repeatable read | N | N | Y |
| Serializable | N | N | N |

19. What is dead lock?

a **deadlock** is a state in which each member of a group waits for another member, including itself, releasing a [lock](#).^[1]

How to detect: **Wait-for-graph** is one of the methods for detecting the deadlock situation. This method is suitable for smaller databases.

How to handle: The only way to resolve a SQL Server deadlock is **to terminate one of the processes and free up the locked resource so the process can complete.**

How to prevent:

1. Do not allow any user input during transactions.
 2. Keep transactions as short as possible.
- Ensure the database design is properly normalized.
 - Develop applications to access server objects in the same order each time.
 - Do not allow any user input during transactions.
 - Avoid cursors.
 - Keep transactions as short as possible.
 - Reduce the number of round trips between your application and SQL Server by using stored procedures or by keeping transactions within a single batch.
 - Reduce the number of reads. If you do need to read the same data more than once, cache it by storing it in a variable or an array, and then re-reading it from there.
 - Reduce lock time. Develop applications that obtain locks at the latest possible time, and release them at the earliest possible time.
 - If appropriate, reduce lock escalation by using `ROWLOCK` or `PAGLOCK`.
 - If the data being locked is not modified very frequently, consider using `NOLOCK` to prevent locking.
 - If appropriate, use the lowest possible isolation level for the user connection running the transaction.
 - Consider using bound connections.

20. What is binary lock?

it is either locked or unlocked.

21. shared and exclusive locks

Shared lock: read lock

Exclusive lock: write lock

22 Optimistic locking vs Pessimistic locking

Optimistic locking , **where a record is locked only when changes are committed to the database.**

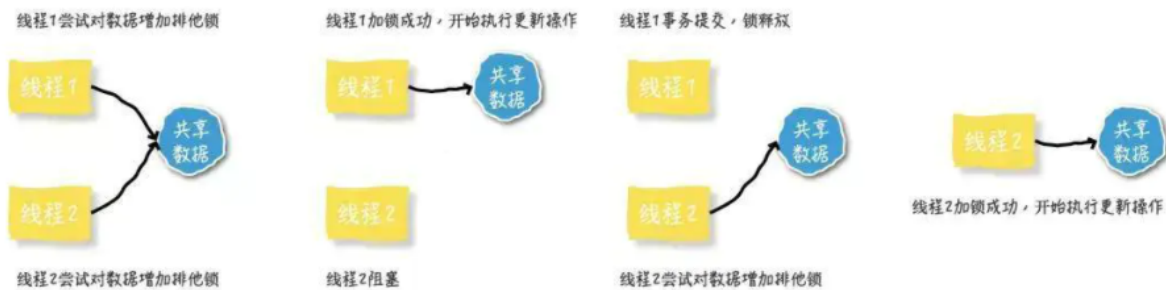
1 理解

乐观锁是相对悲观锁而言的，乐观锁假设数据一般情况不会造成冲突，所以在数据进行提交更新的时候，才会正式对数据的冲突与否进行检测，如果冲突，则返回给用户异常信息，让用户决定如何去做。乐观锁适用于读多写少的场景，这样可以提高程序的吞吐量。



2. 版本号控制：一般是在数据表中加上一个数据版本号 version 字段，表示数据被修改的次数。当数据被修改时，version 值会 +1。当线程 A 要更新数据时，在读取数据的同时也会读取 version 值，在提交更新时，若刚才读取到的 version 值与当前数据库中的 version 值相等时才更新，否则重试更新操作，直到更新成功。

Pessimistic locking , where a record is locked while it is edited.

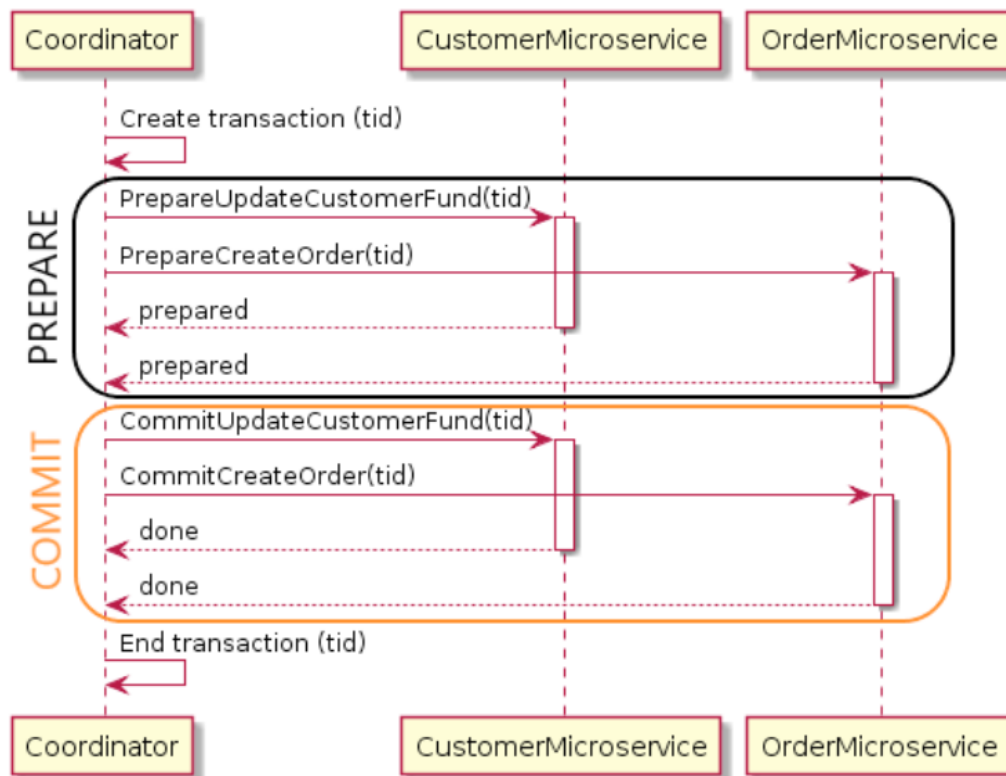


1 悲观锁实现方式

悲观锁的实现，往往依靠数据库提供的锁机制。在数据库中，悲观锁的流程如下：

1. 在对记录进行修改前，先尝试为该记录加上排他锁(exclusive locks)。
2. 如果加锁失败，说明该记录正在被修改，那么当前查询可能要等待或者抛出异常。具体响应方式由开发者根据实际需要决定。
3. 如果成功加锁，那么就可以对记录做修改，事务完成后就会解锁了。
4. 期间如果有其他对该记录做修改或加排他锁的操作，都会等待解锁或直接抛出异常。

23. What is 2PC(two phase commit)?



A two-phase commit is a **standardized protocol that ensures that a database commit is implementing in the situation where a commit operation must be broken into two separate parts.**

Prepare phase:

Commit phase:

24. 2PC vs Saga

Saga: **Saga simply sacrifices atomicity and relies on eventual consistency.**

- Typically, 2PC is for *immediate* transactions.
- Typically, Sagas are for *long running* transactions.

25. SQL Structured Query Language

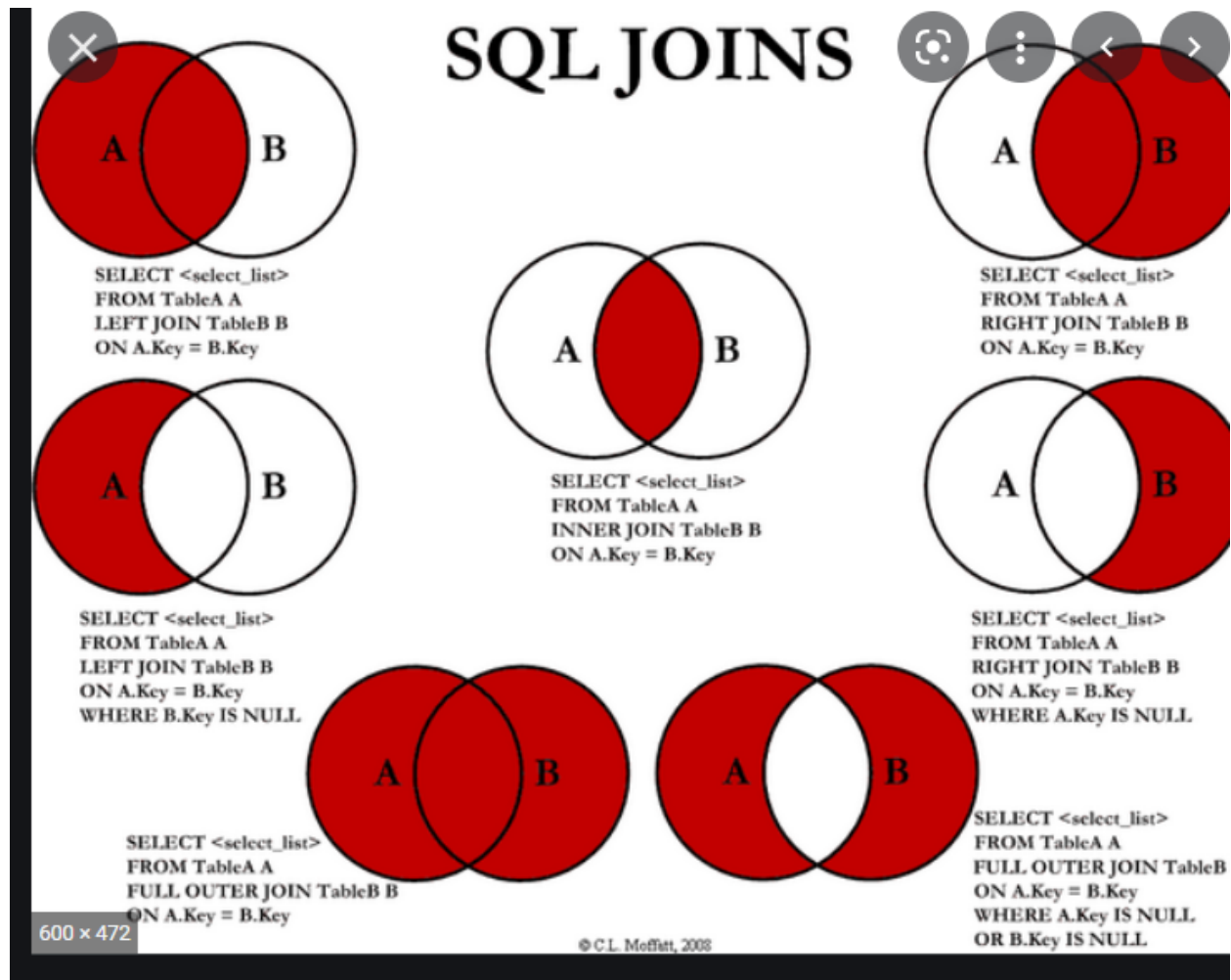
- DDL (data definition language)
 - create, drop, alter, truncate,
- DQL(data query language)
 - select ...
- DML (data manipulation language)
 - insert, update, delete
- DCL (data control language)
 - grant, revoke
- DTL (data transaction language)
 - commit, rollback

26.union vs join

Difference between JOIN and UNION in SQL :

| JOIN | UNION |
|--------------------------------------------------------------------------------|-----------------------------------------------------------------------------|
| JOIN combines data from many tables based on a matched condition between them. | SQL combines the result-set of two or more SELECT statements. |
| It combines data into new columns. | It combines data into new rows |
| Number of columns selected from each table may not be same. | Number of columns selected from each table should be same. |
| Datatypes of corresponding columns selected from each table can be different. | Datatypes of corresponding columns selected from each table should be same. |
| It may not return distinct columns. | It returns distinct rows. |

27. left join vs outer join



28. union vs union all

Union all includes duplicate data.

29. having vs where

the Where clause acts as a **pre filter** where as Having as a **post filter**.

The **where** clause works on row's data, not on aggregated data.

The **having** clause works on aggregated data.

