[redacted] [redacted]

#1. String is immutable. It can't be changed.
StringBuilder and StringBuffer both are mutable.
But StringBuffer is thread safe while StringBuilder is faster and more efficient.

#2. ① S1 == S2 : False
② S1. equals.(S2) : True.

#3.
```
Public static void main (String[] args){
   ...
 }
```

#4. Multiple inheritance is one class inherits (extends) from multiple different class.
Java does not support multiple inheritance (extends), but it supports multiple implements from interface.

#5. ① ~~multiy~~ Abstract class ~~doesn~~ does not support multiple inheritance.
But interface can do multiple implements.
② Abstract class has its constructor while interface doesn't have it.
③ abstract class can have abstract method and non-abstract method, while interface can only have abstract method.

#6. Maker interface in java is an interface which does not have any method.
For Example: serialization.

#7. Static polymorphism: overload
```
public class Person {
     public void getInfo() {
          system.out.println("person exists!");
 }
//overload
public void getInfo (String name){
          system.out.println("person" + name );
 }
```
same class. same name. different argument.

dynamic polymorphism: override.
```
public class people extends person{
   //override.
   public void getInfo(){
        system.out.println("people here")
   }

 }
```
in Subclass. same signature.

#8.
```
public class Singleton{
    private static Singleton obj;              // static variable single-instance of singleton type
    private Singleton(){};                     // private constructor
    public static synchronized getInstance(){  // static getInstance() method
                                               // synchronized.
        if (obj == null){
            obj = new Singleton();
        }
        return obj;
    }

    @override
    public Object clone() throws CloneNotSupportException(){   // clone override.
        throw CloneNotSupportException;

    }
```

#9.
```
public abstract class Foxconn{
    public abstract makePhone(){};

}
public class Iphone extends Foxconn{
    ...
    // make Iphone
}
public class Huawei extends Foxconn{
    ...
    // make Huawei
}
// factory class
public class PhoneFactory{
    public static Foxconn makePhone(String brand){
        if ("Iphone". equals(brand)){
            return new Iphone();
        }
        if ("Huawei". equals(brand)){
            return new Huawei();

        }
    }
}
```

#10 −0.25

ArrayList : based on array. it is not thread safe. insert = O(n), read: O(n) index ?

Linked List : ~~they a~~ it is doubly linkedlist. then it is two way.
it is thread safe. insert (O(1)) read: O(n)
not thread safe

[redacted]

#11. we need to override hashcode() and equals().

#12. there are many buckets. use hashcode() to differtent from them and store data into different buckets. ~~The~~ HashMap use key-value pair.

-0.5 ? equals()

#13. we can't use for loop. But we can use Iterator.

#14. ~~checked~~
They are two types of exception.
checked exception is compile time . ~~we use try . catch . finally to achieve it.~~ SQLException.
-0.75 For example:
unchecked exception is runtime. . For example: null pointer exception.
NullPointerException.

#15. ① extends Thread
② implements Runnable.

#16. ExecutorSerivce helps to maintain thread pool.
ExecutorService excutor = Executor. newCachedThreadPool();
use invokeAll() to start all the threads. return a future type. use shutdown() to shut down manully.

#17.
```
public class USstates{
    public static void main (string[] args){
        List<States> states = Arrays. asList (
                new States ("New York"),
                new States ("New Jersey"),
                new States ("Virginia")
        };
        States result = states.stream()
                .filter((s)→ "N". equals (s. getName() [0]     )
                .findAny()
                .orElse (null);
        system. out. println("result:   " + result);
    }
}.
```

#18. Functional Interface has one single abstract method. It has an annotation @ functionalInterfa

#19. SUM(salary) as Totalsalary

Select Department, TotalSalary from Employee.
group by Department
having ~~sal~~ Totalsalary > 2000
order by Department desc.

—

#20. Final → ① method can't be overrided, ② class can't be extended ③ field : it is a constant.

Finally → try, catch, finally. it will definitelly be excuted.

Finalize → garbage collections.

#21. git rebase : change the master branch to a feature branch. No need commit.

-0.5 git merge : merge a feature branch to the master branch. need commit.

#22.

$$\begin{tikzcd}
 & & & \text{Model} \arrow[dl] \arrow[d] \arrow[dr] & & \\
\text{FrontEnd} \arrow[r] & \text{Security} \arrow[r] & \text{controller} \arrow[r] & \text{service} \arrow[r] & \text{DAO} \arrow[r] & \text{Database.}
\end{tikzcd}$$

#23.
```
public String reverseString (String s) {
    char[] chars = ~~static~~ s.toCharArray();
    int i=0 ;
    int j = s.length() - 1 ;
    while (i<j) {
        Char tmp = chars[i];
        Chars[i] = chars[j];
        Chars[j] = tmp;
        i++; j-- ;
    }
    return new String(chars);
}}
```