- finalize()
- final vs finally vs finalize
- try with resources
- Autocloseable

- Comparable
- Comparators
- Diff b/w comparable vs comparator

- Iterable

- Semaphore Code

- Functional Interface and lambda

⇒ Finalize

⇒ finalize() ⇒ method, that is invoked GC, when GC comes to destroy the object it calls finalize()

↓

present in object and we need to override

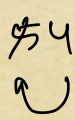
↓

used to free resources being used by obj, that GC is cleaning

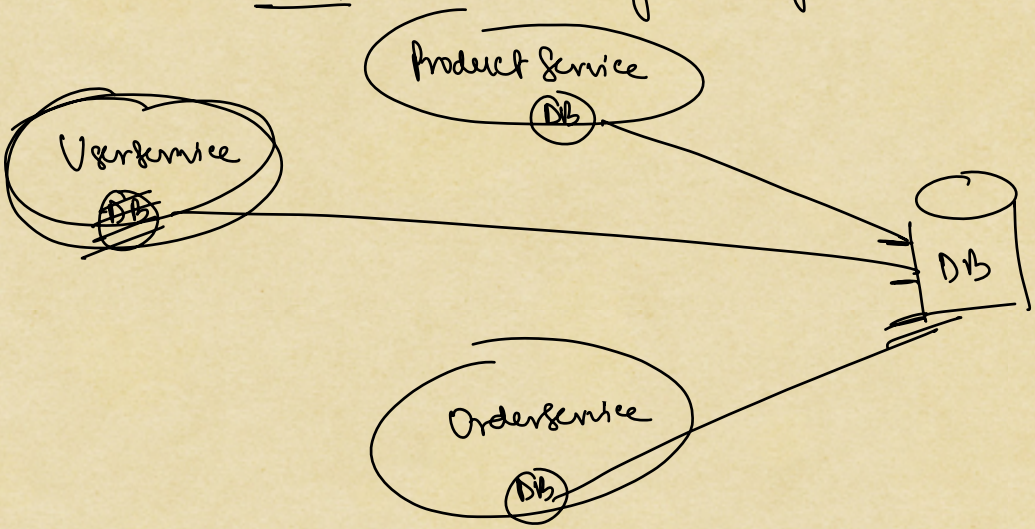
Person


UserService

DBConn ← used (x--)



→ destructor → destroys the Object




finalize ⇒ Object

↓

class Person {

@Override → non-mandatory / good to have

public void finalize() {

}

}

→ finalize() ⇒ not used in industry anymore

↓

deprecated

: finally ⇒ block

↓

use for clearing up resources

→ works with try | catch | both and definitely executes.

: final ⇒ keyword

↓

→ variables ⇒ can't post object creation

→ methods ⇒ can't override

→ classes ⇒ can't inherit.

⇒ try with resources :-

Resources → implement "AutoCloseable" are only usable in try with resources.

try {

Scanner sc = new ————————

sc.nextInt();

}

finally { sc.close(); }

try ( Scanner sc = new Scanner( System.in) ) {

sc.nextInt();

}

→ moment the code block completes it automatically closes and cleans up the resources.

* Heavy use in Industry

→ DB Connections

→ Msg queue connections

→ File I/p or File O/p Streams.

⇒ Autocloseable

close()

↓

resource should implement AutoCloseable

for which we want auto-cleanup

» Comparables :

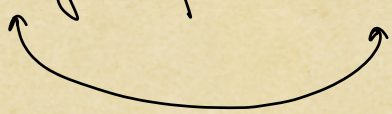
Comparable interface

```
compareTo(T o) {
    == comparing logic
}
```

Car implement Comparable

↓

```
car1.compareTo(car2);
```



eCommerce

Product

price
Sales
name
margin

compareTo( )