(12) Static

a) A static variable belongs to the entire class, not just to a specific obj.

→ Single copy of var created and stored among all obj of the class. Memory allocation only happens once.

b) Syntax → private static float deliveryCharge = 1.5f;

(OR)

Static Block → static {
deliveryCharge = 1.5f;
}

when multiple static var are used.

c) Since static var shared by whole class they are not called by obj name but by classname.

System.out.println("DCharge = " + customer1.deliveryCh ;

obj name.

Use Customer ← i.e class name.

Getter & Setter are also static functions which changing value of static variables. (this. - not used, Customer. → used)

d) Good example shown

(13) Aggregation.

a) → Till now we only looked at independent objs (OR) objs which don't interact with objs of other classes.

→ Relationship like these in Java are covered by

→ Inheritance, Aggregation & Association.

6) Aggregation → Also called Has-a relationship.

Employee class & Dept class in a company.
Dept Has-a Employee as manager.

Library class & book class → Lib has a books.

```
public class Order {

    private int orderId;
    private Food orderedFoods;
    private double totalPrice;
    private string status;

    public Order (int orderId, Food orderedFood)

        this.orderId = orderId;
        this.orderedFoods = orderedFoods;
        this.status = 'ordered';
    }
}

public class Food{

    private String foodName;
    private int quantity;
    private double price;
    private String cuisine;
}
```

class. Used as a datatype in Order.

---

(14) Association.

→ Uses 'uses-a' relationship.

Man uses-a car to reach the office.
Patient uses-a appointment to meet doctor.
Teacher uses-a stick to - - -

Eg:-
→ generateBill method uses Order object (instance of order class) to access & display order details.

extra:
: if all constructors private, getInstance will be used.

Constructor chaining.

Only static → static overload can happen.

(15) Inheritance.

a) → Three levels in Hotel → Regular
(or) four. Premium
Premium Plus
Guest.

We make a common customer class that all of the levels inherit.

So, Guest → customer + extra delivery
Regular → Customer
Premium → Regular + benefits
Premium plus → (Premium + benefits) + extra benefits.

'Is-a' relationship followed → So, Premium 'is-a' regular
→ Dog 'is a' animal.
etc.

b) Syntax

→ Class customer {
. . . . .
}
Class Regular extends Customer { (used for inheriting.)
. . . . .
}

Parent constructor called before the child constructor.

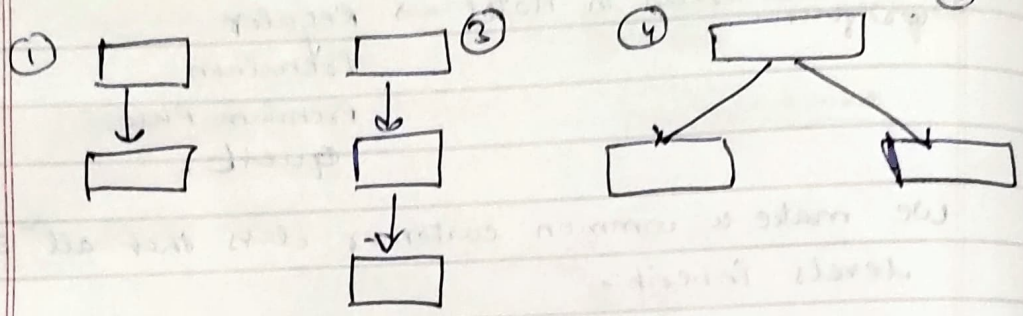
c) → super() keyword → used to execute child constructor before parent (needed in certain cases). → wrong

for default values to parent. → Used to give values to parent class var in const which need parameters.

super(custId, custName); → first thing in child constructor.

(OR) to call overridden superclass (parent class).

d) Types of Inheritance → ① Single, ② Multiple, ③ Multi-lvl, ④ Hierarchical



Multiple inheritance (many to one) not supported by Java.

---

(16) Polymorphism.

→ means 'many forms'.

a) Bank loans → different rates for diff customers
different types.

b) 2 types → dynamic.
→ static. (compile time).

c) For Static

Method overloading → multiple methods with same name & in same class with signature changed.

→ signature can differ by:-

i) No: of parameters
ii) data types of "
iii) order of parameters.

Constructor overloading is also possible.

---

d) for dynamic Polymorphism.

↓ At runtime.

→ Method overidding is used.

→ to override, signature must be same.

* Also to override, access modifier should be stronger i.e for eg:- private classes/methods cannot be overidden.

→ Static methods cannot be overridden.

→ See more examples for clarity.

Dynamic Binding.

---

e) Annotations & Super (Optinal).

@override → annotation used to make code more readable by java itself.

many types of diff annotations in java.