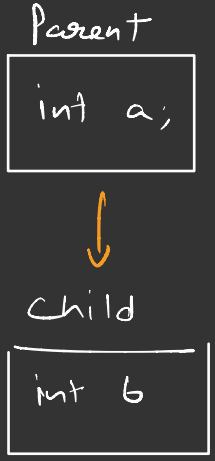
Pillars of OOPs

1. Inheritance
2. Polymorphism
3. Abstraction
4. Encapsulation

(1) Inheritance → same as inheriting the properties of parents



Child class inherit properties from parent

Syntax

```
class Parent {
  int a;
  void printA(){
    syso(a);
  }
}
```

extends Parent

```
class Child {
  int b;
  void printB(){
    syso(b);
  }
}
```

```
Child c = new Child()
c.b = 10;
c.printB();
c.a = 20;
c.printA();
```

Types of inheritance

(1) Single level ⇒

(2) Multilevel ⇒

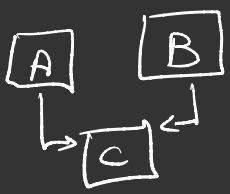
```
class A {
  int a;
}
class B extends A {
  int b;
}
class C extends B {
  int c;
}
```


```
main() {
C obj = new C();
obj.c;
obj.b;
obj.a;
```

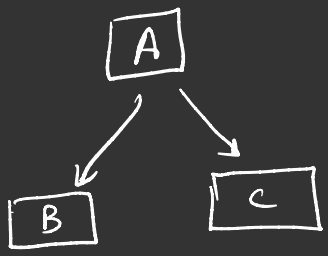
has properties of C, B and A

(3) Multiple Inheritence

not allowed in java. ⇒

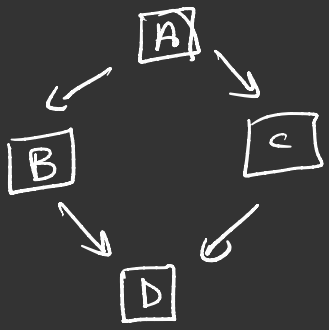


④ Hierarchy Inheritence



⑤ Hybrid Inheritence

not allowed



super

↳ access parent of current class

↳ call the constructor of Parent class


```
class A {
    int a
}

class B extends A {
    int a
}
```

② Polymorphism

many — forms

① Compile time / Static

② Run time / Dynamic

Function Overloading

```
void Sum(int a, int b) {
    syso(a+b);
}

void Sum(int a, int b, int c) {
    syso(a+b+c);
}

void Sum(double a, double b) {
    syso(a+b);
}

main() {
    sum(2,3);
    Sum(2,3,4);
}
```

Sum(1,2,3)

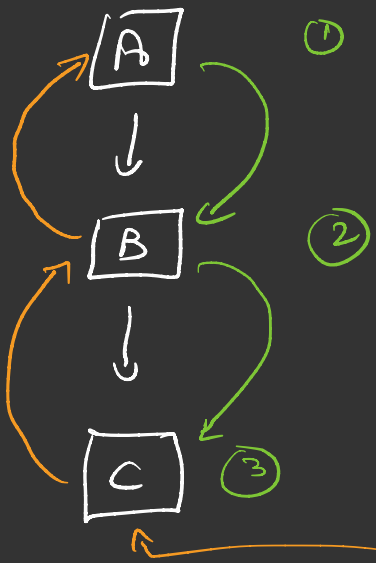
sum(2.4, 3.1)

→ Either

→ no of parameters is different

→ datatype of parameters are different

Constructor Call order



C obj = new C();