

Pillars of OOPS

- ↳ ① Inheritance
- ② Polymorphism
- ③ Abstraction
- ④ Encapsulation

① Inheritance → same as inheriting the properties of parents

Parent
int a;



Child
int b

Child class inherit properties from parent

Syntax

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

ans

extends Parent

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

```
Child c = new ChildC()
```

```
c.b = 10;
```

```
c.printB();
```

```
c.a = 20;
```

```
c.printAC();
```

Types of inheritance

① Single level \Rightarrow



② Multilevel \Rightarrow

```
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;

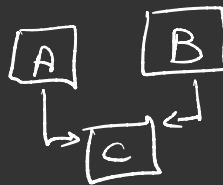
obj.a;

has properties of
C, B and A

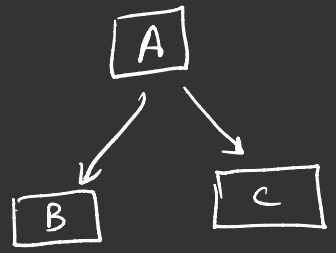
③ Multiple Inheritance

not allowed in
java.

\Rightarrow

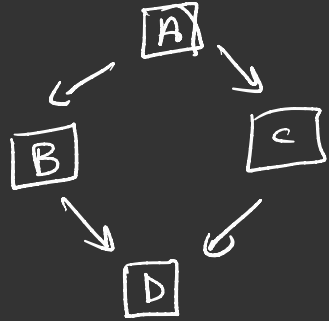


④ Hierarchy Inheritance



⑤ Hybrid Inheritance

not allowed



super

↳ access parent of current class

↳ call the constructor of parent class

A

B

② Poly morphism

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(1, 2, 3)  
    Sum(2, 3);  
    Sum(2, 3, 4);  
}
```

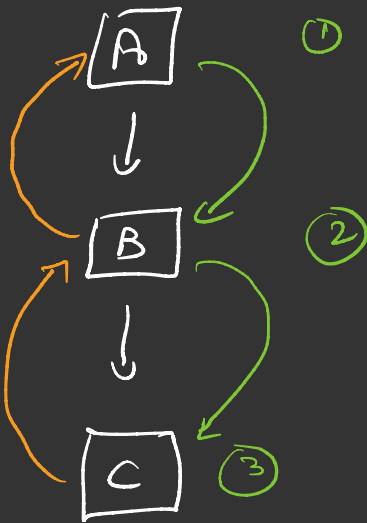
sum(2.4, 3.1)

→ Either

→ no of parameters is different

→ datatype of parameters are different

Constructor Call order



C obj = new C()