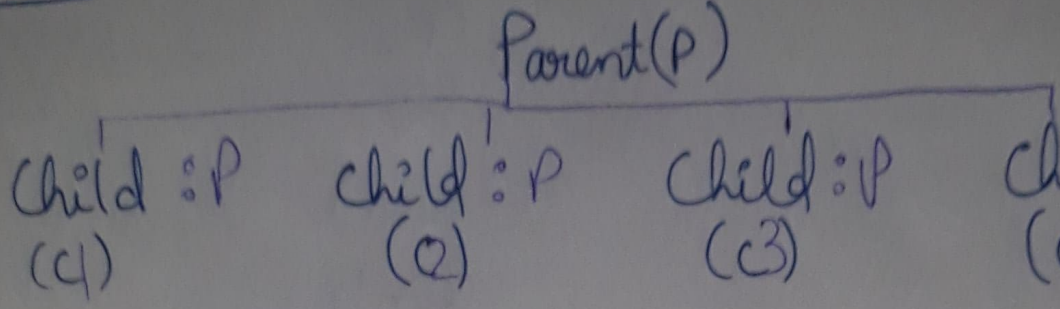
Polymorphism




```
P[] arr = { new C1(), new C2(), new C3(), new C4()};
arr[0].fun();        arr[1].fun();
arr[2].fun();        arr[3].fun();
```

↑ example of Dynamic polymorphism (Method Overriding) The fn to be run not known at compile time (Late Binding)

(Dropdown)

Compile time Polymorphism

1. Function Overloading
2. Generics

(Fn to be called determined by Compiler)

eg:-

```
fun(int a, int b) {
    return a+b;
}
```

```
fun(double a, double b) {
    return (a+b)/2;
}
```

Called via: fun(1, 2)

Example of method overloading

Called via: fun(1.0, 2.0)

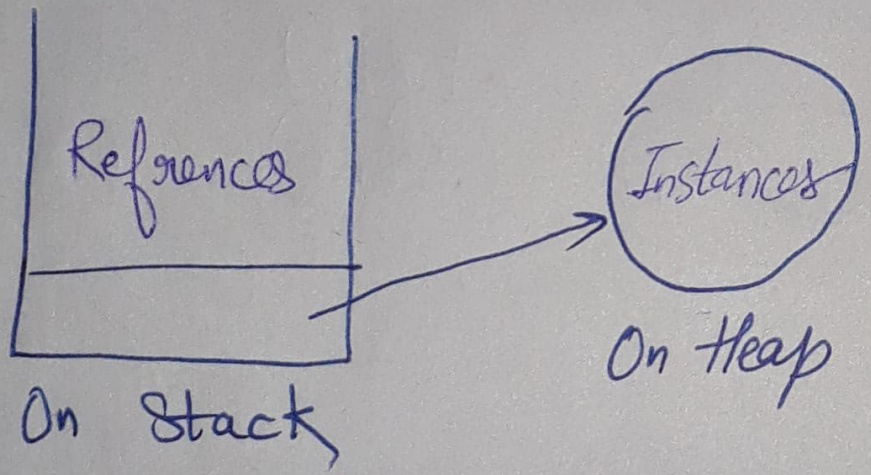
⇒ No operator overloading in Java But a special case for strings exist where append() functionality is overloaded using + operator.

→ Complex refrences (resolved by typecast)

→ data-members, non-virtual] Resolved by Refrences (LHS)

virtual] Resolved by Instances (RHS)

→ In Java all functions by default are virtual.




```
P o1 = new P()
P o2 = new C()
C o3 = new P()   Not allowed
C o4 = new C()
```

At Compile time refrences are resolved.

At Runtime instances are resolved.