```
Customer{
id,
name,
mobile,
}
```

```
Product{
id
name
price

}
```

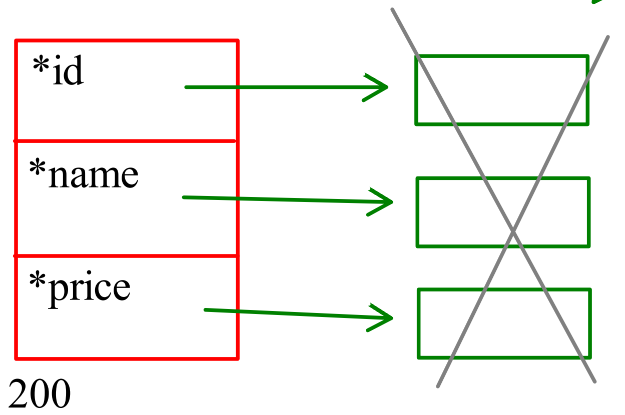
```
vector<Product> v1;
Product p;
v1.push_back(p);
```

```
addElement(T element){
v1[i] = element;
}

push(T element)
```

```
vector<Product*>p1
```

Rohan
1,2,3,4

Nilesh
1,2



200

Person

Employee Student

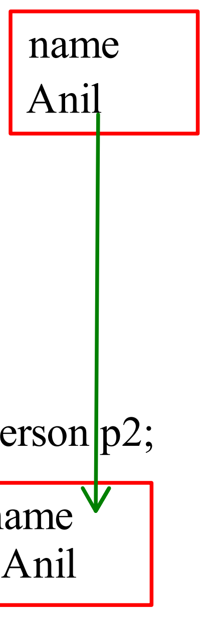
```
class Person
{
name
}

class Employee{
empid
salary
}

class Student{
rollno
marks
}
```

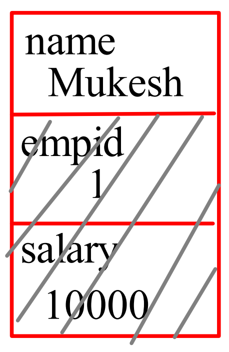
```
vector<Person*> v;
Person p;
v.push_back(p);
Employee e;
v.push_back(e);
```

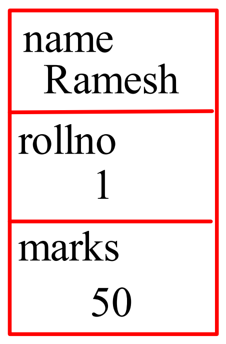
Person p1;

Employee e1;

Student s1;

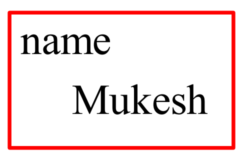






Person p2;

Person p3 = e1;



p2=p1

Employee e3 = p3;

generaated_id.txt

```
fout<<id;
```

1,2,34

4

```
loadEmployee(){
Employee *e;
for(){

e=list[i];
}
e->getId;
Employee::gid = e->getId;
}
```

```
Customer{
id,
name,
mobile,
vector<Product*> pp;
}
```

```
Product{
id
name
price

}
```

```
vector<Product*> p;
```

```
class Person{
name
}

class Employee{
id,
salary
}

class Student{
rollno
marks
}
```

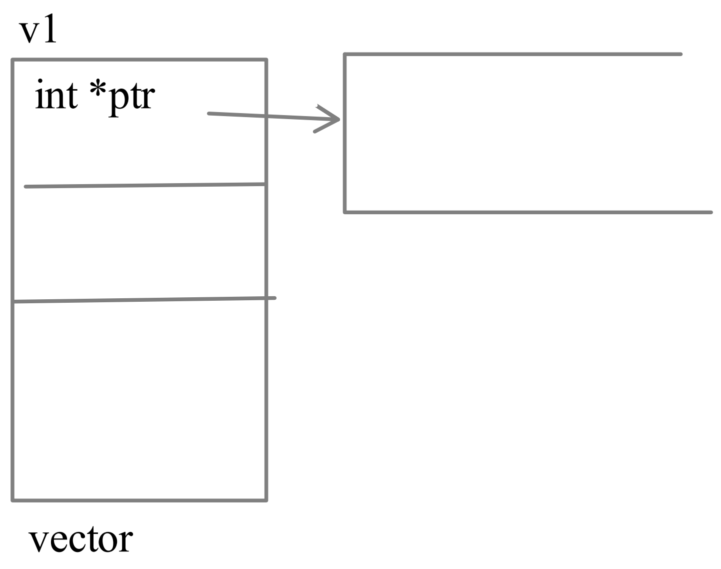
```
vector<Person *> v1;
v1.push_back(new Employee(1,"Anil",10000));
v1.push_back(new Student(1,"Mukesh",60));
```

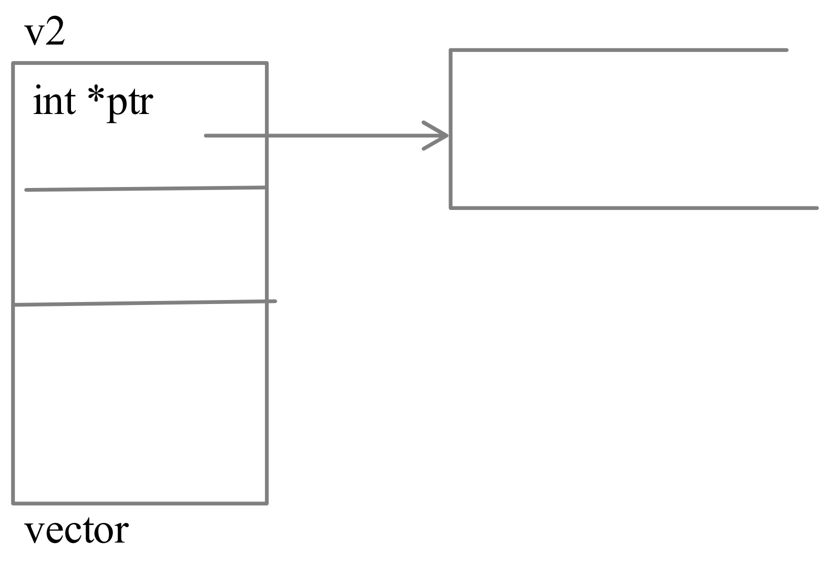
```
person.txt
E,1,"Anil",10000
S,1,"Mukesh",60
```

```
if(flag == 'E')
el.pushBack(new Employee(s1,s2,s3));
else
sl.push_back(new Student(s1,s2,s3));
```

```
stringstream(line);
getline(data,flag,',')
String s1,s2,s3;
```

```
vector<Employee *> el ;
vector<Student *> sl ;
```



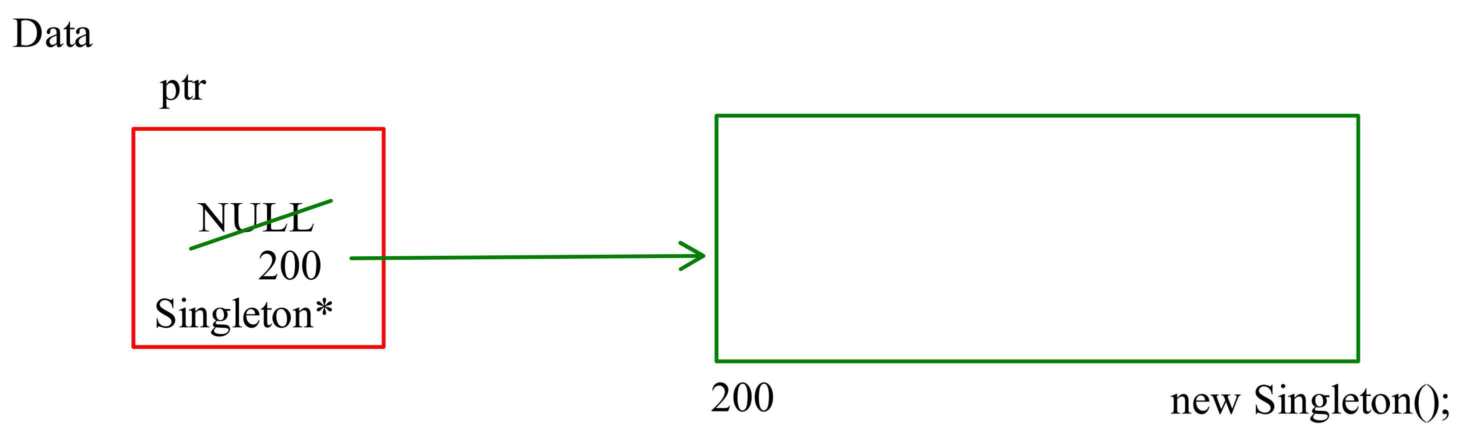


```
class Play{
display
sensors
button click

}
```

```
Play p1;
Play p2;
Play p3;
```

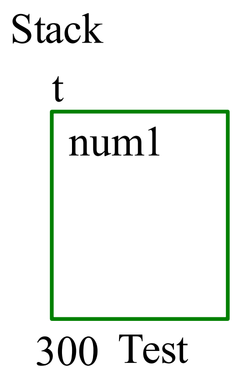
Retrofit
// Apis

Flipkart -> request
Amazon
Meesho

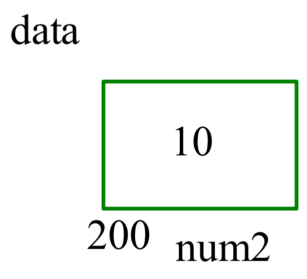


```
class Test{

int num1;
static int num2;

void f1(){
}

static void f2()
{
Test t;
t.num1;
num2;
}

};
```

```
int main(){
Test::f2();
}
```



300.num1;
t.num1;



C++
OOP-> Concurrency
class, object
static
const
Inheritance,virtual, upcasting, downcasting, runtime polymorphism
Exception handling
Operator overloading
Dynamic Memory managment
Reference