Hirerachical Inheritance

```
class Product{
int id;
string title;
double price

virtual void accept(){
id,title,price
}
}
```

10% discount

```
class Book:public Product{
string author;

void accept(){
author
Product::accept();
getPrice -> 10% discount
setPrice
}
}
```

5% discount

```
class Tape:public Product{
string artist;

void accept(){
artist
Product::accept();
getPrice -> 5% discount
setPrice
}
}
```

```
main();
Product *arr[3];
```

1. Add Book -> arr[index] = new Book();
arr[index]->accept();
2. Add Tape -> arr[index] = new Tape();
arr[index]->accept();
3. Calculate Total Bill
double total = 0;
for(int=0;i<3;i++){
    total = total + arr[i]->getPrice();
}

```
3. Calculate Total Bill
double total = 0;
for(int=0;i<3;i++){
    if(typeid(*arr[i])==typeid(Book)){
    double newPrice = 10% discounted price
    total = total+newPrice;
}
else{
    double newPrice = 5% discounted price
    total = total+newPrice;
}

}
```

Virtual Destructor

Q. When to make the base class dtor as virtual?

- When upcasting is done and we have dynamic memory allocation in the derived class, so to free/deallocate the dynamic memory in the derived class we need to call the derived class dtor However because of upcasting only the base class dtor gets called when the object goes out of scope To give the call to the dtor of derived class first and then the base class we must declare the base class dtor as virtual.

Advanced Casting Operators

1. dynamic_cast -> classes are polymorphic
2. static_cast -> classes are not polymorphis, but inheritance exists
3. reinterpret_cast -> no relationship between the entities
4. const_cast -> To remove the const qualifier for the types

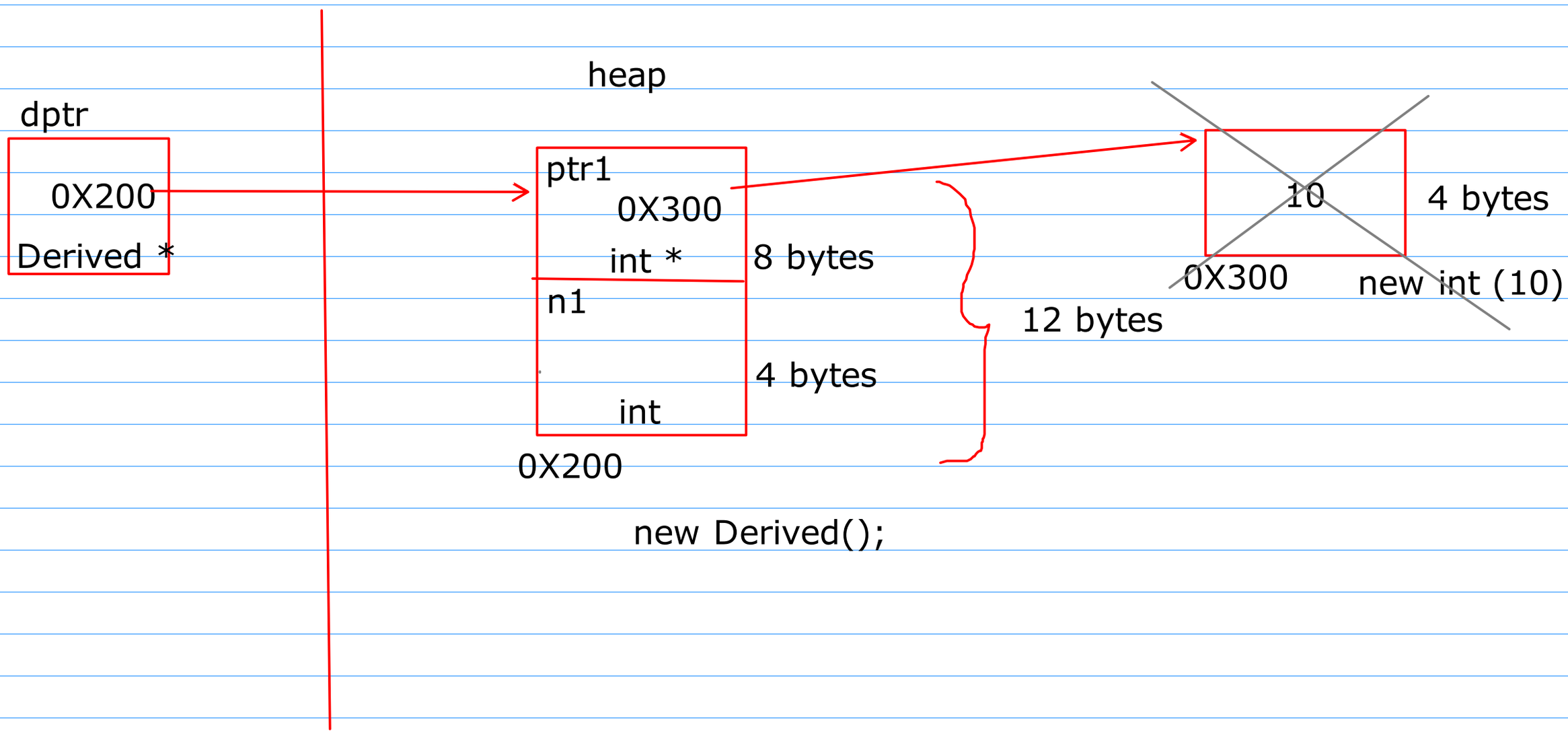
```
Employee e;
Manager m;
Salesman s;
Salesmanager sm;
```

```
e.accept();
e.display();
```

```
Employee *e;
Manager *m;
Salesman *s;
Salesmanager *sm;
```

```
Employee * e = new Manager();
Employee * e = new Salesman();
Employee * e = new Salesmanager();
```

```
Manager * m = new Manager();
Manager * m = new Salesman();
Manager * m = new Salesmanager();
```



Templates

Java -> Generics

1,4,5,2,3
44.55,11.22,55.66,77.88,22.33
E,C,A,B,D

```
swap(v1, v2){
    temp = v1;
    v1 = v2;
    v2 = temp;
}
```

vector

```
class Array{
T *ptr;

Array(int size=5){
ptr = new T[size];
}
void addElement(T ele){
}
int getIndex(T ele){

}
}
```

Exception Handling

- Why?
    - To seperate business logic from the error handling logic
- How?
    - To perform exception handling use try,catch and throw keywords