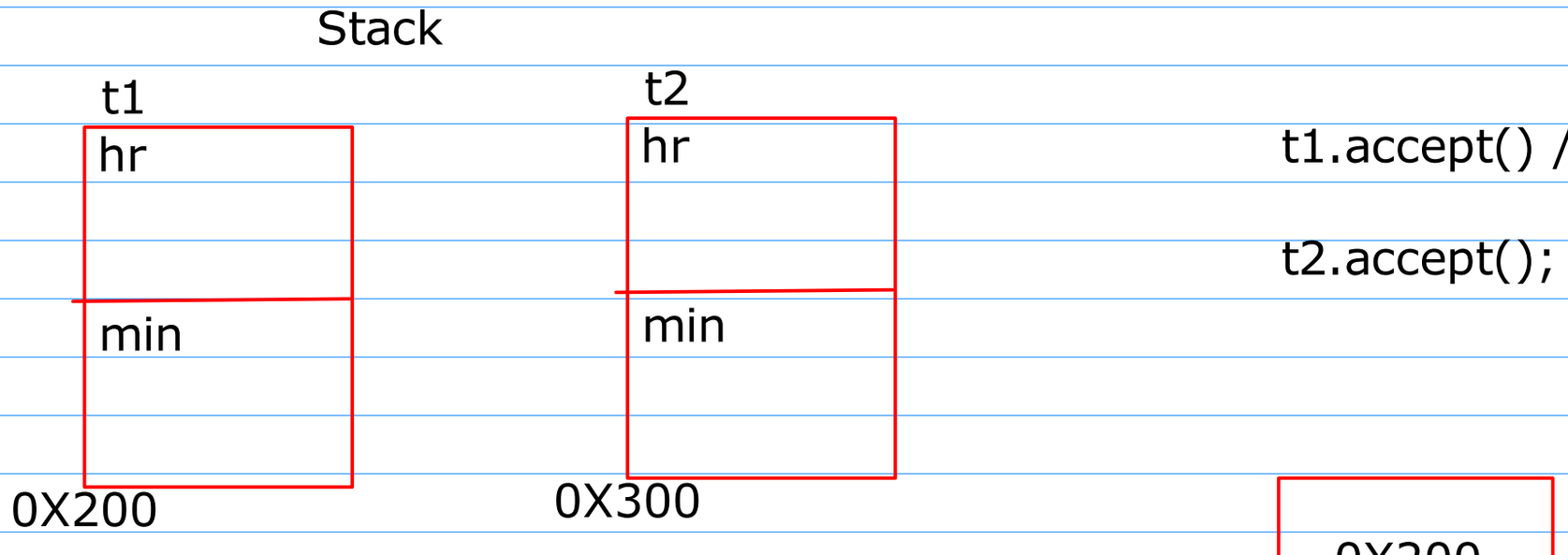
this Pointer

- It is a constant pointer passed internally to all the non static member functions of the class
- This pointer points to the current calling object



0X200.hr = 10;
t1.min = 20

Types of member functions

1. Constructor
2. Destructor
3. Mutator (Setter)
4. Inspector
5. Facilitator

Constructor

- It is a special Member Function
1. The name of Ctor is same as that of the class name
2. Ctor do not have any return type
3. the ctor gets called automatically when the object is created

Types of Constructor

1. Default/Parameterless
2. Parameterized Constructor
3. Copy Constructor

Its own Syntax
Its own Rules

int n = 10;
int *ptr = &n;

ptr -> &n
&ptr -> address of the ptr
*ptr -> value of the n

Ctor members initializer list

- It is a list that is used to initialize the state of an object in the same way in which the data memebrs are declared
- We cannot use this in the ctor members initializer list
- First the ctor members initializr list will be executed first and then the ctor body

Constructor Delegation (Java->Ctor Chaining)

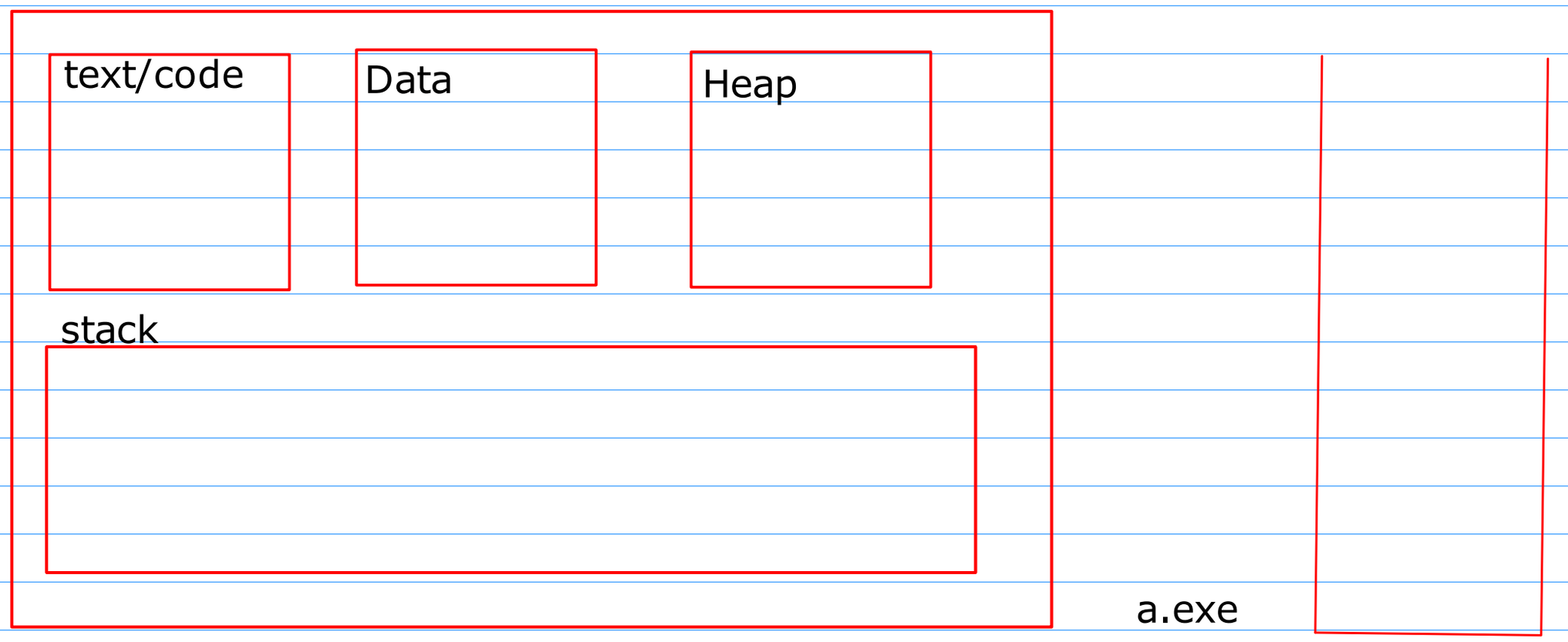
Destructor

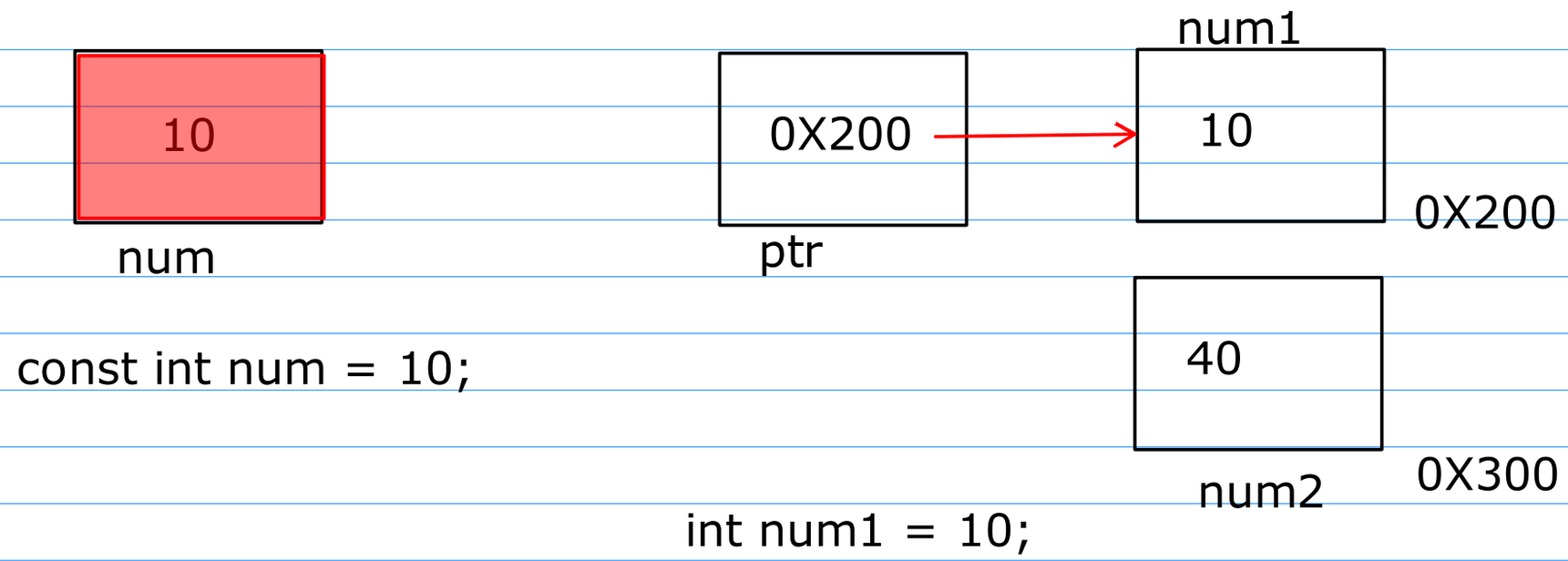
- It is a special member function of the class
- The name of dtor is same as that of class name but with a tild sign (~)
- It does not have any return type
- It gets called when the object goes out of scope

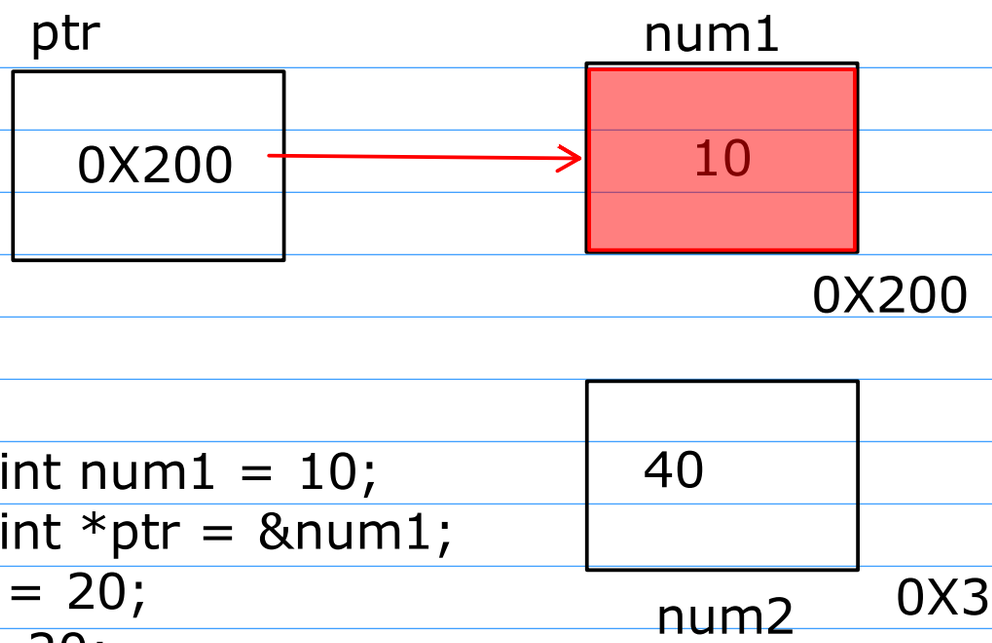
- The dtor calling sequence is exactly opposite to that of ctor calling sequence

Ctor Delegtion

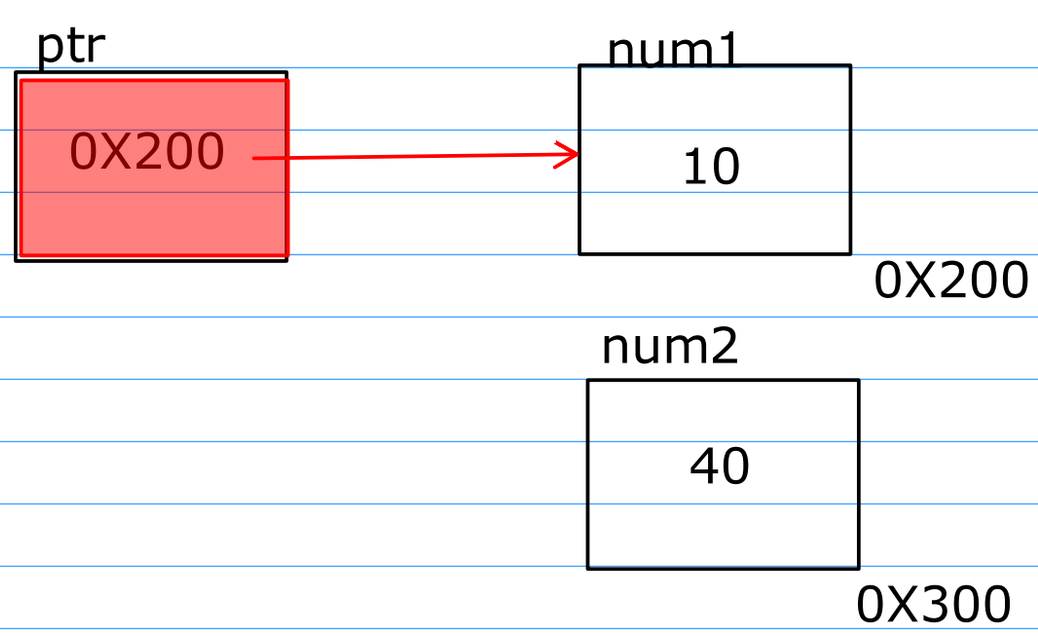
Task -> Login & Registration(Team Lead)



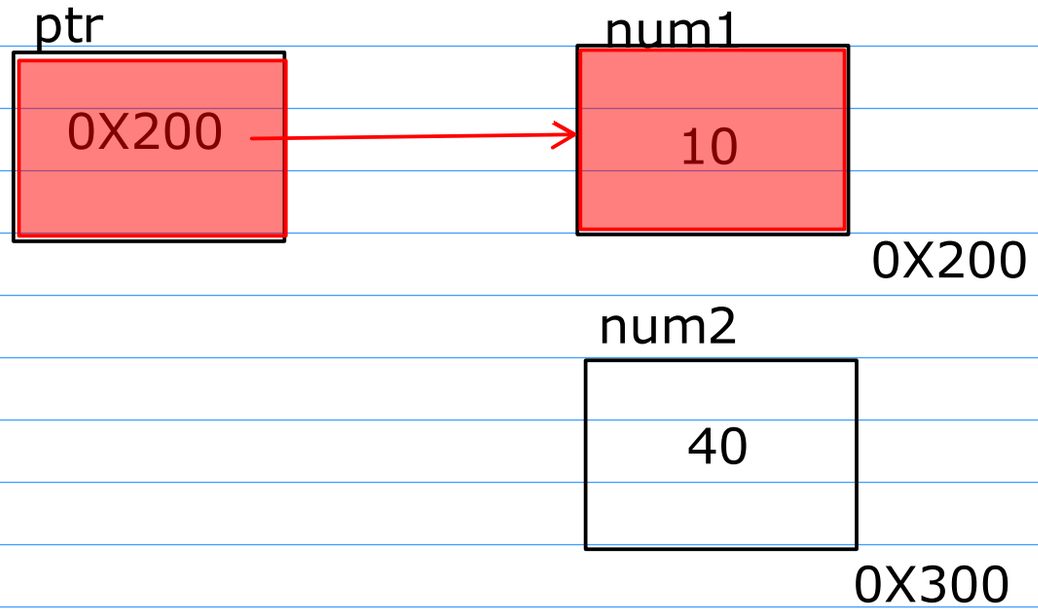




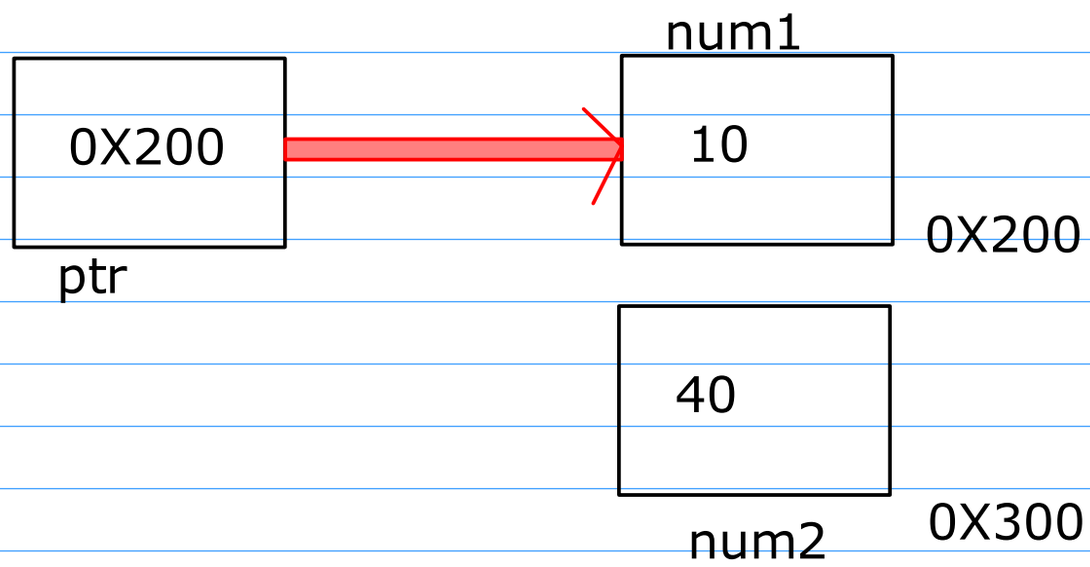

```
const int num1 = 10;
const int *ptr = &num1;
num1 = 20;
*ptr = 20;
int num2 = 40;
ptr = &num2;
```




```
int num1 = 10;
int num2 = 20;
int * const ptr = &num1;
//ptr = &num2; // NOT OK
```

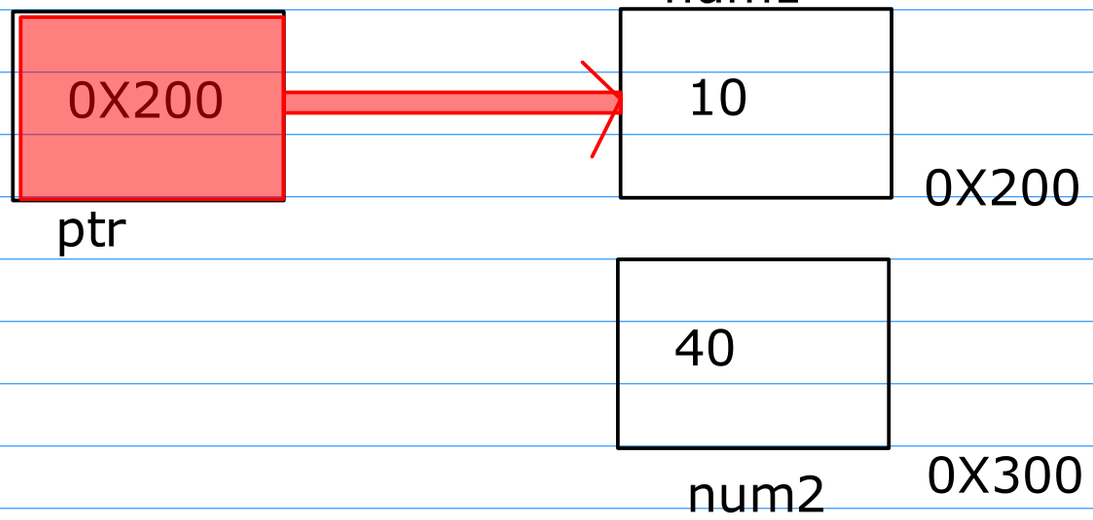



```
const int num1 = 10;
int num2 = 20;
const int * const ptr = &num1;
//ptr = &num2; // NOT OK
//*ptr = 20; // NOT OK
//num1 = 20;//NOT OK
```




```
int num1 = 10;
const int *ptr = &num1;
num1 = 20;
//*ptr = 20; // NOT OK
int num2 = 40;
ptr = &num2;
```

num1




```
int num1 = 10;
int num2 = 40;
const int*const ptr = &num1;
num1 = 20;
// *ptr = 20;// NOT OK
// ptr = &num2; // NOT OK
```

```
void accept(){ // Time * const this
Time t1;
this = &t1; // NOT OK
}
```