Java^{TM} Education & Technology Services

Object Oriented programming Using

C++

Friend Function

```
class Complex {

                :

    };

Complex addTo ( float v , Complex c ) {

    Complex b;

    b.setReal ( c.getReal () + v ) ;

    b.real = c.real + v ;          ✗

  return b;

}

int main() {

    Complex a (2, 5);

    a = addTo( 4 , a) ;
```

Stand alone function

Friend Function

```
class Complex {:

    friend  Complex addTo ( float v , Complex c ) ;

    };
```

friend function

```
Complex addTo ( float v , Complex c ) {

    Complex b;

    b.setReal ( c.getReal () + v ) ;

    b.real = c.real + v ;

  return b;

}
```

Stand alone function

```
int main() {

    Complex a (2, 5);

    a = addTo( 4 , a) ;
```

Friend Function

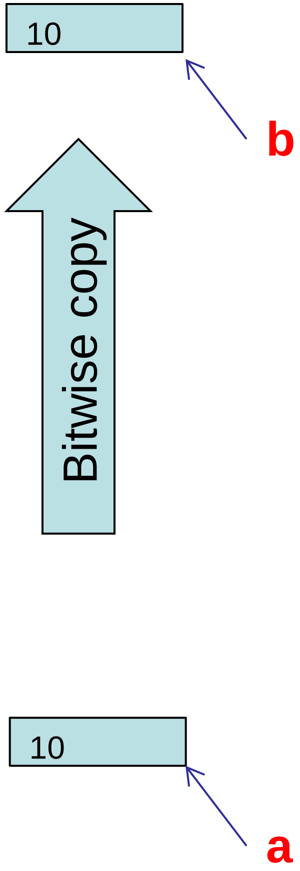
1. is a stand alone function or another class member function.
2. is a nonmember function that can deal with private and public members in the class.
3. violate the encapsulation concept.
4. its prototype should be declared inside the class.
5. one function can be friend for many classes.

Can a class be a friend for another class? and How?

Passing an object to a function

```
void myFunc (int b ) {

        :

}
```

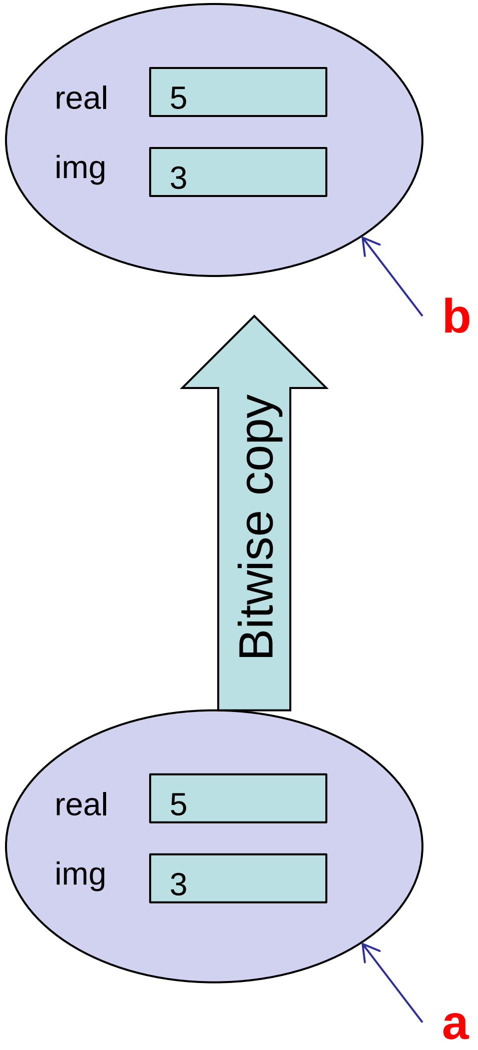
```
int main() {

        int a = 10 ;

        myFunc( a) ;

}
```



Passing an object to a function

```
void myFunc ( Complex b ) {

        :

}
```

```
int main() {

        Complex a ( 5, 3);

        myFunc( a) ;

}
```



Passing an object to a function

1. b is a local variable in myFunc and myFunc uses call by value.
2. the constructor for complex class will run only once for a .
3. and a will be bitwise copied into b when myFunc is called.
4. at the end of myFunc the destructor for complex class will run for b.
5. at the end of main the destructor for complex class will run for a.

Constructor	Destructor
a //main	b //myfun
	a //main

Passing an object to a function

1. Make a stand alone friend function to view the stack content.

```
class Stack{
        :
        friend  void viewContent ( Stack x ) ;
        };
```

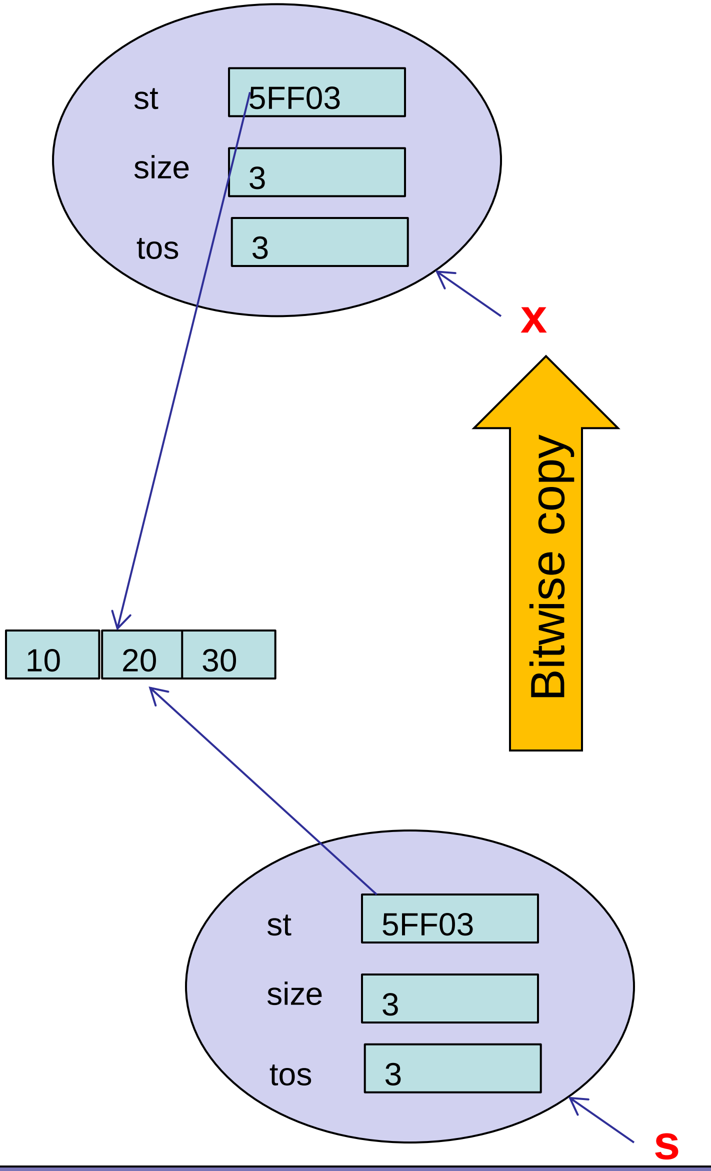
```
void viewContent ( Stack x ) {
int t = x.tos;
while ( t != 0 )
        cout<< x.st [--t]  << endl;
}
int main() {
```

```
        Stack s(3) ;
        :
        viewContent ( s) ;
```

Passing an object to a function

```
void viewContent ( Stack x ) {
        :
}
```

```
int main() {

        Stack s(3) ;

        :

        viewContent( s) ;
```

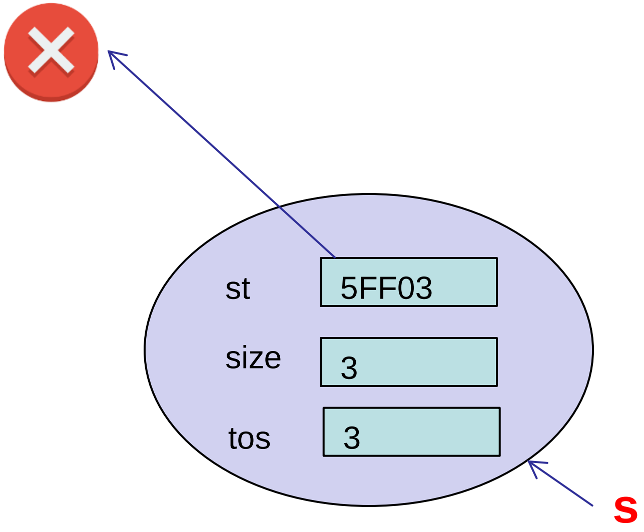


Passing an object to a function

1. **Problems ☹**

   1. shared area between two objects.
   2. at the end of **viewContent** Function the destructor for object **x** will free the pointer **st** so the object **s** at the main couldn't continue work correctly.

Dynamic Area Problem



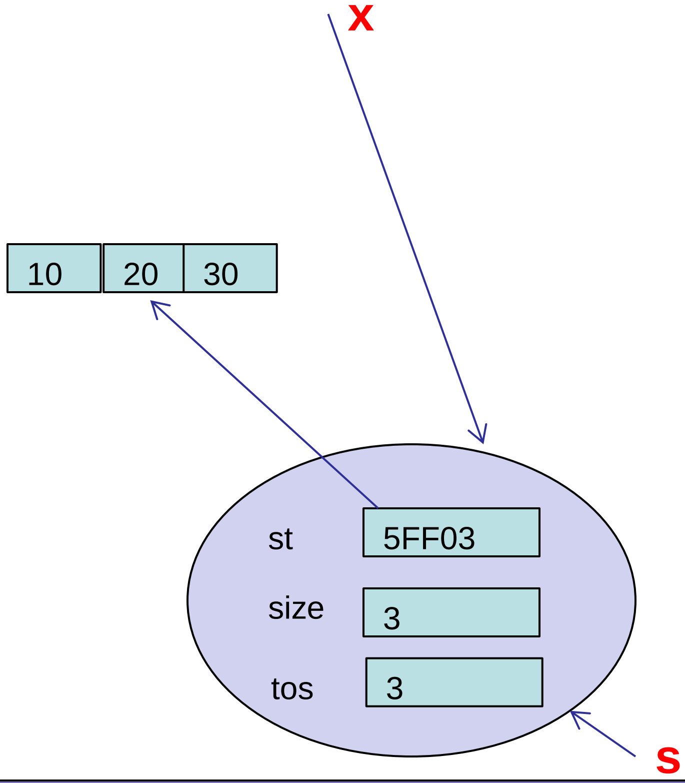
Passing an object to a function

1. Solutions for dynamic area problem☺
    1. using Call by Reference :

```
class Stack{
      :
      friend  void viewContent ( Stack &x ) ;
      };
```

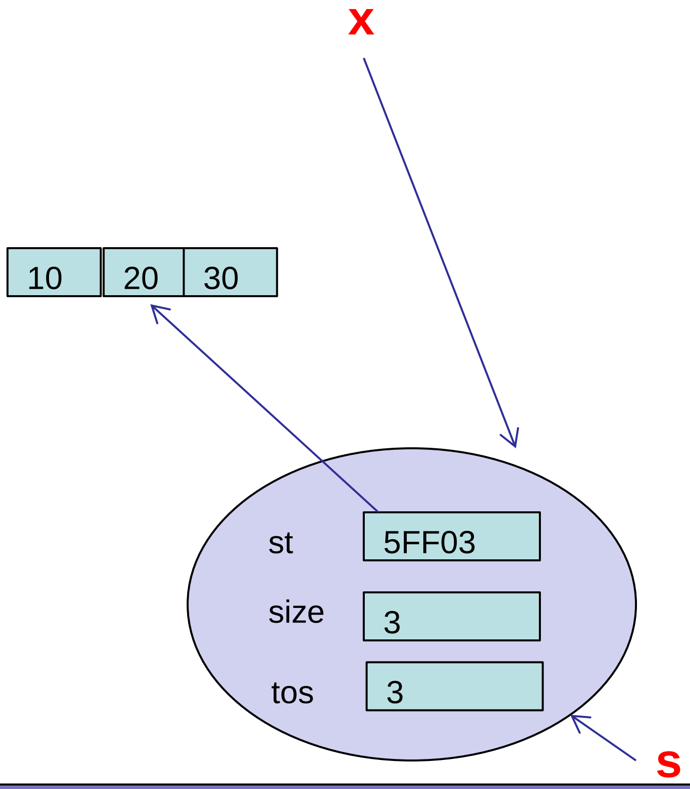
```
void viewContent ( Stack &x ) {
:
}
```

```
int main() {
      Stack s(3) ; ............................
      viewContent( s) ;
```



Passing an object to a function

1. Solutions for dynamic area problem☺
   1. using Call by Reference :
      - » Here we have 2 **references** not 2 **objects**
      - » at the end of **viewContent** function there is no destructor will run .



Passing an object to a function

1. Solutions for dynamic area problem☺
   2. using Copy Constructor :
      - » If the class you make has a pointer attribute and destructor free this pointer, add a copy constructor to your class.
      - » copy constructor does not apply bitwise copy so always new object is created.
      - » no need here to update viewContent function

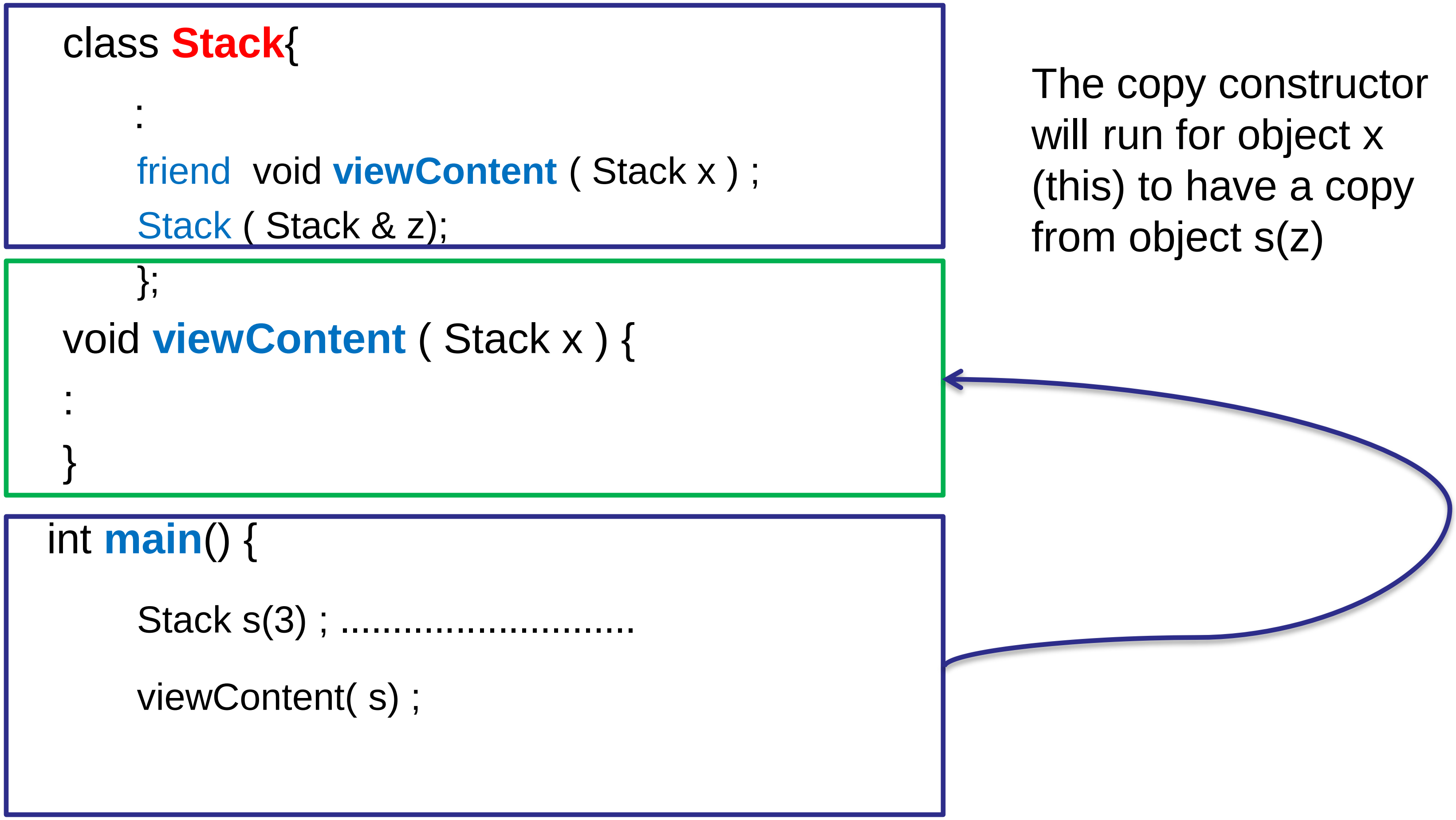
Passing an object to a function

1. **Solutions for dynamic area problem☺**
   2. **using Copy Constructor :**

```
class Stack{
        :
        friend  void viewContent ( Stack x ) ;
        Stack ( Stack & z);
        };

Stack :: Stack ( Stack  & z) {     // passing 2 params (this : new one , z : old one)
        tos = z.tos;
        size = z.size;
        st = new int [size];    // new allocation
        for(int i=0 ;  i< tos; i++ )
                        st [i] = z.st [i] ;     //  only copy values

        counter++
}
```

Passing an object to a function

1. Solutions for dynamic area problem☺
   2. using Copy Constructor :



```
class Stack{
        :
        friend  void viewContent ( Stack x ) ;
        Stack ( Stack & z);
        };
```

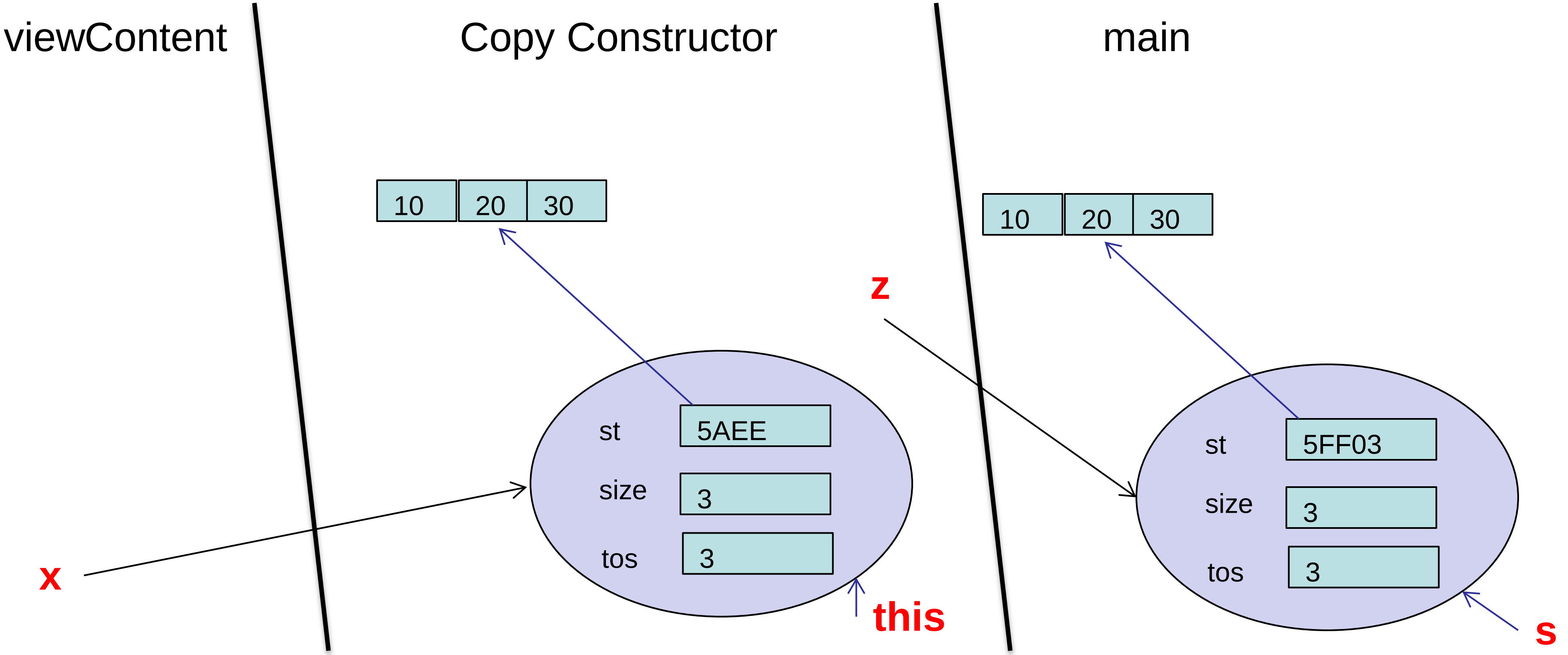
```
void viewContent ( Stack x ) {
:
}
```

```
int main() {

        Stack s(3) ; ............................

        viewContent( s) ;
```

The copy constructor will run for object x (this) to have a copy from object s(z)

Passing an object to a function

1. Solutions for dynamic area problem☺
    2. using Copy Constructor :



Passing an object to a function

1. **Solutions for dynamic area problem☺**
    2. **When the Copy Constructor called ?**

» When we pass an object as a parameter by value to function. √

» When we construct new object from old one.

```
Stack s1;
Stack s2 ( s1);  // this new : s2 and z old : s1
```

» Return from function by value

```
Stack getStack() {
            Stack a; ..................
           return a; // this new : will receive and z old : a
}
```

Passing an object to a function

1. Solutions for dynamic area problem☺
   2. When the Copy Constructor called ?

```
Stack s1(10);

Stack s2 (s1);

viewContent(s1);

Stack s3( getStack() );


// end of the program
```

Lab Exercise

Lab Exercise

- 1st Assignment :

1. **Complete Stack Class:**
   1. viewContent function once call by reference.
   2. viewContent function once call by value and without copy constructor.
   3. viewContent function once call by value and with copy constructor.

   – **count objects and constructors and destructors calls**