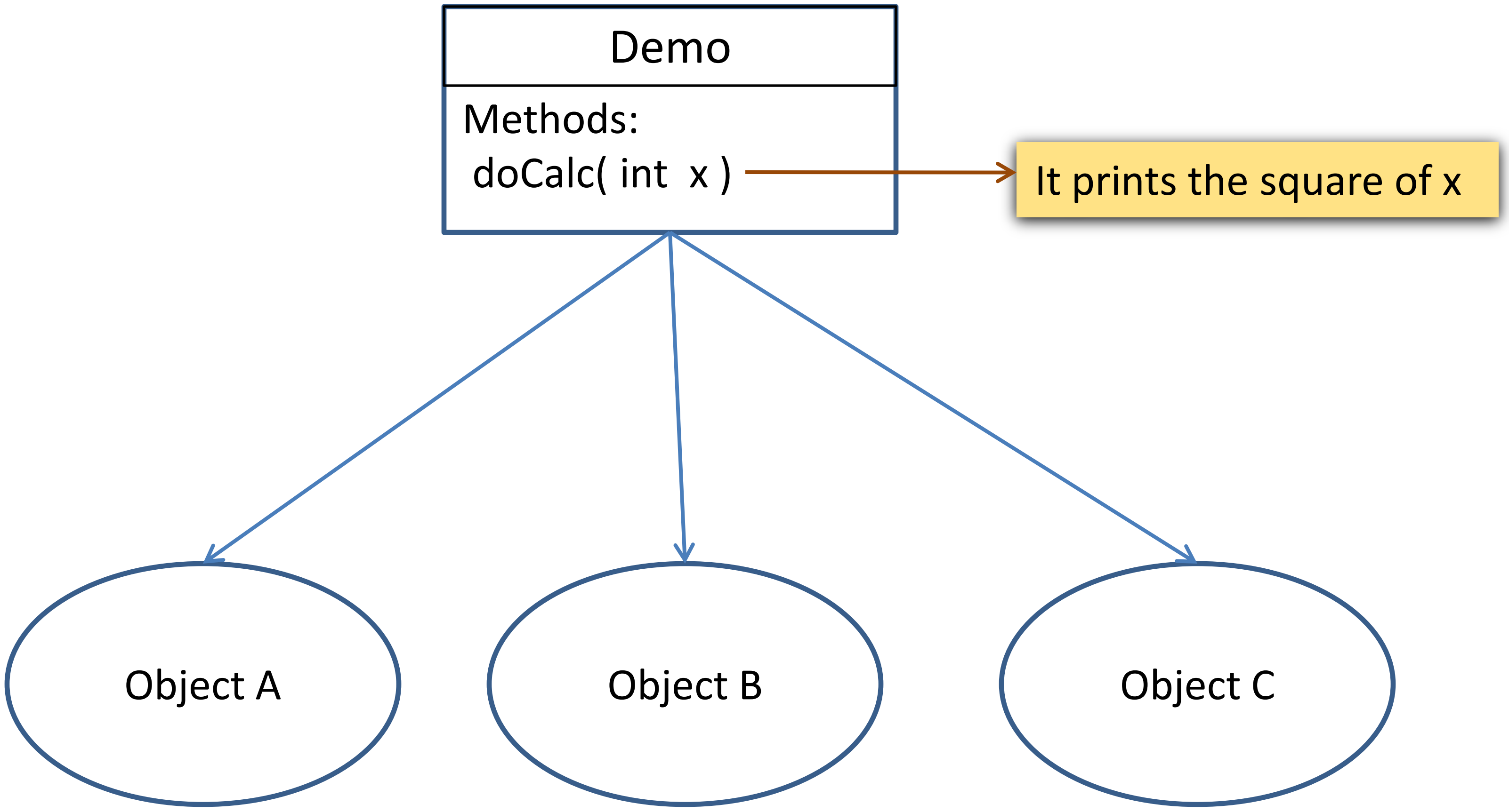
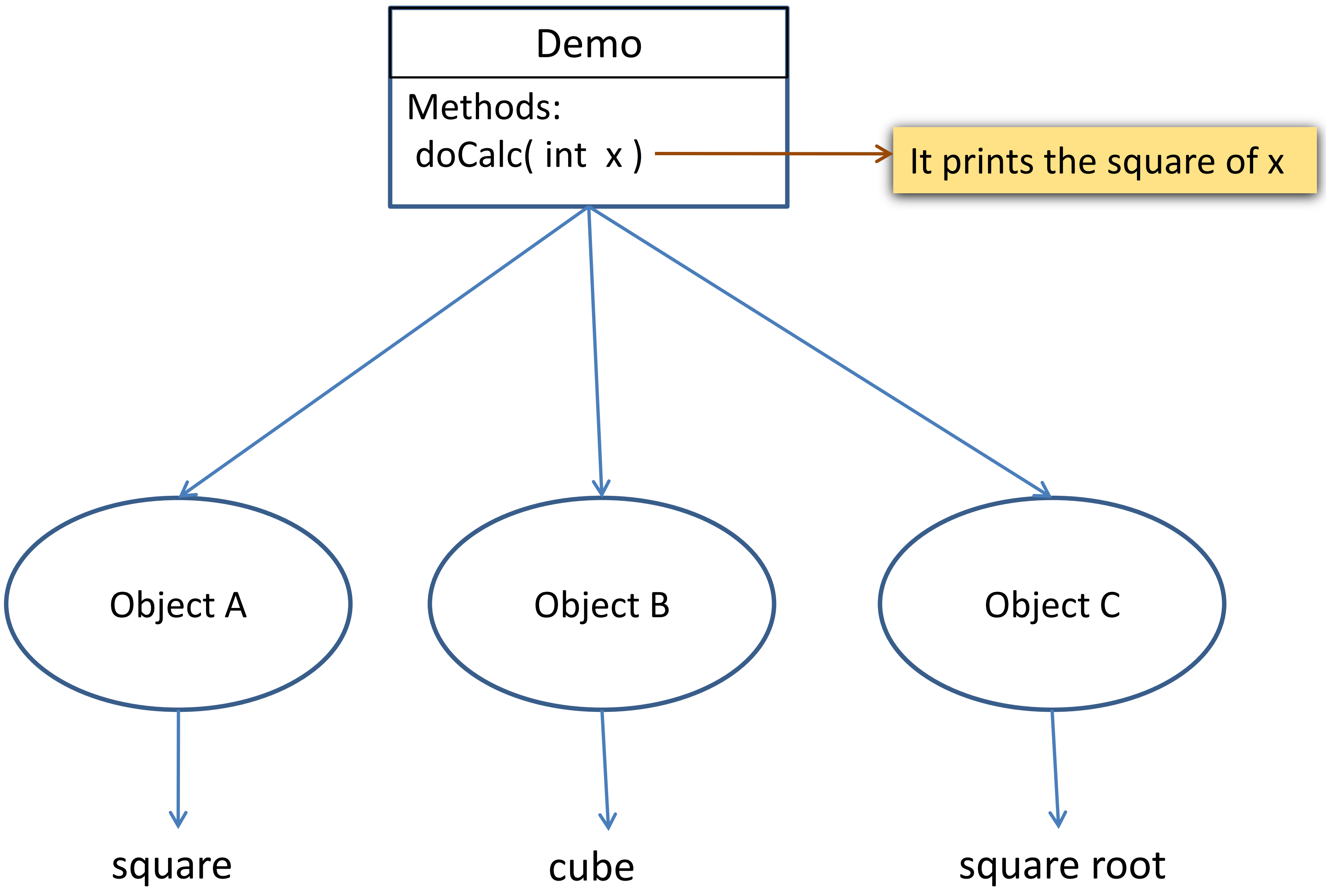
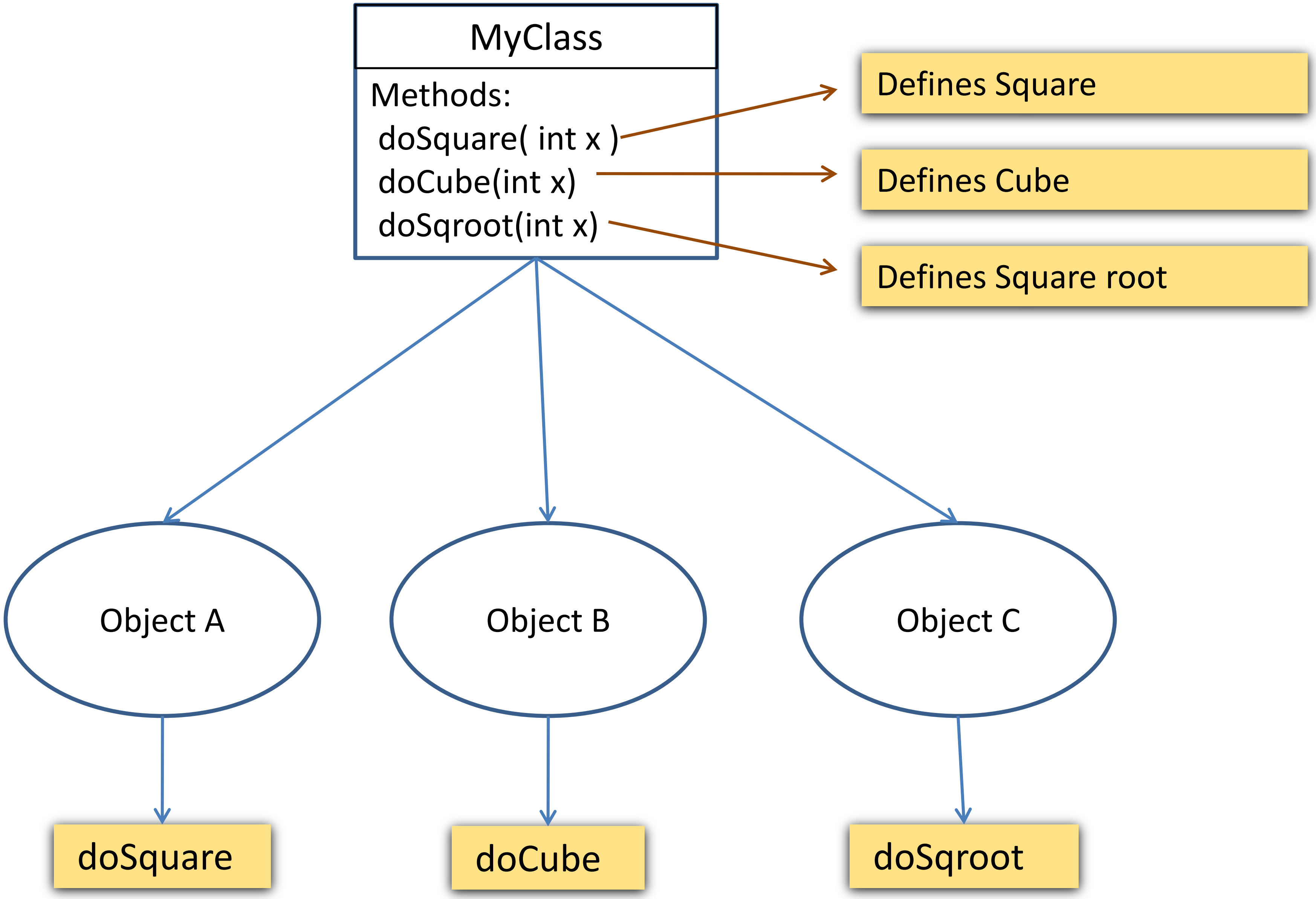
Abstract Classes



Here, A.doCalc( ) prints the square of given value, Similarly B.doCalc( ) and C.doCalc( ) also prints the square of the given value.



What ? Objects needs are different



It is the responsibility of Class to define the behavior of objects

SuperClass

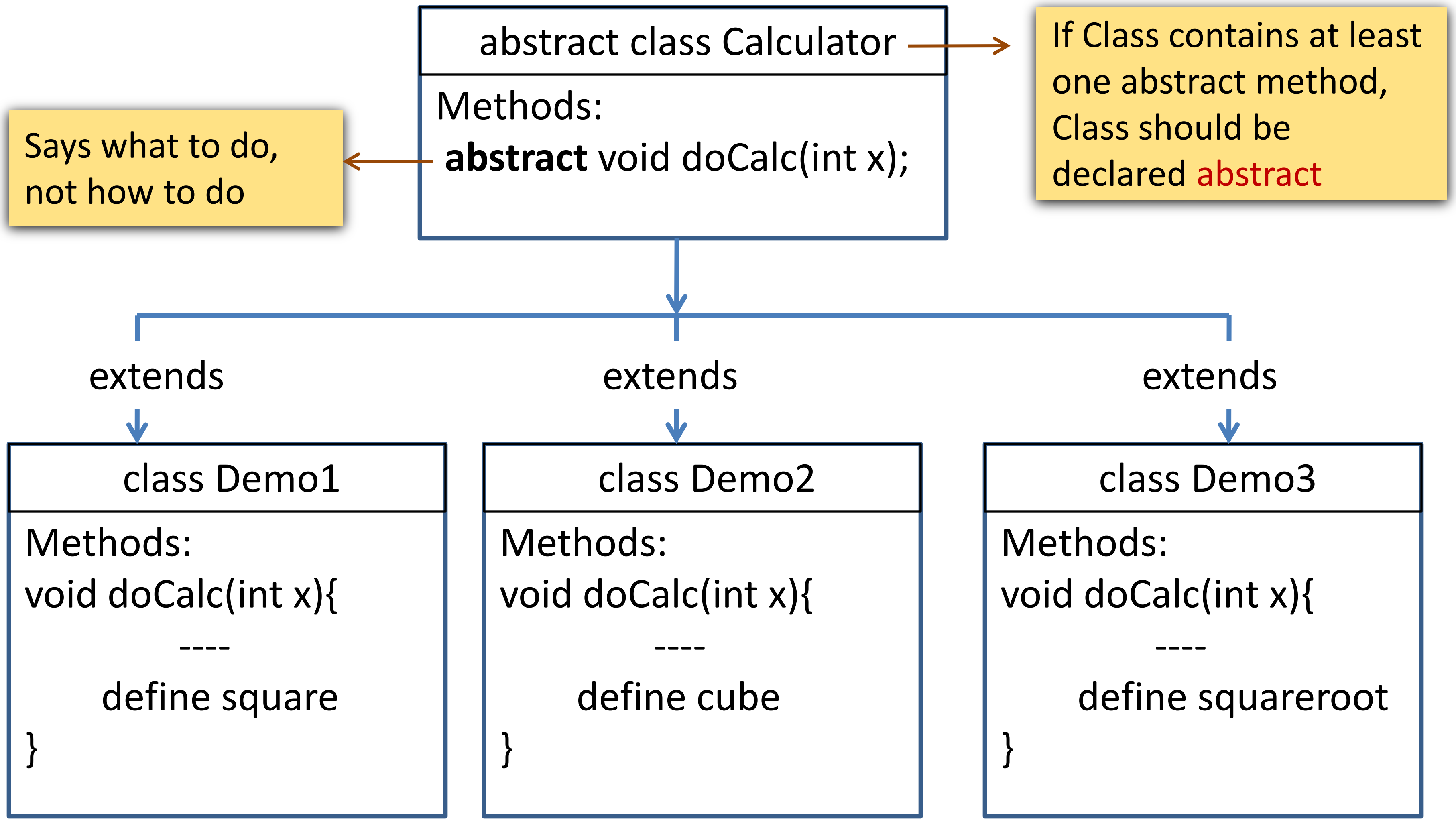
Methods:

abstract void doCalc(int x);

If Class contains at least one abstract method, Class should be declared abstract

Simply says what to do, not how to do

Now, Any class, which extends SuperClass, it is mandatory for that class to override abstract methods otherwise it generates compile-time error



```
abstract class Calculator
Methods:
 abstract void doCalc(int x);
```

If Class contains at least one abstract method, Class should be declared abstract

Says what to do, not how to do

```
class Demo1
Methods:
void doCalc(int x){
        ----
    define square
}
```

```
class Demo2
Methods:
void doCalc(int x){
        ----
    define cube
}
```

```
class Demo3
Methods:
void doCalc(int x){
        ----
    define squareroot
}
```

Now, Any class, which extends SuperClass, it is mandatory for that class to override abstract methods otherwise it generates compile-time error

Abstract Methods

- ❑ An abstract method contains only the declaration for a method without any implementation details. It simply says what to do not how to do.

```
abstract void calcData ( double x , double y ) ;
```

Abstract Classes

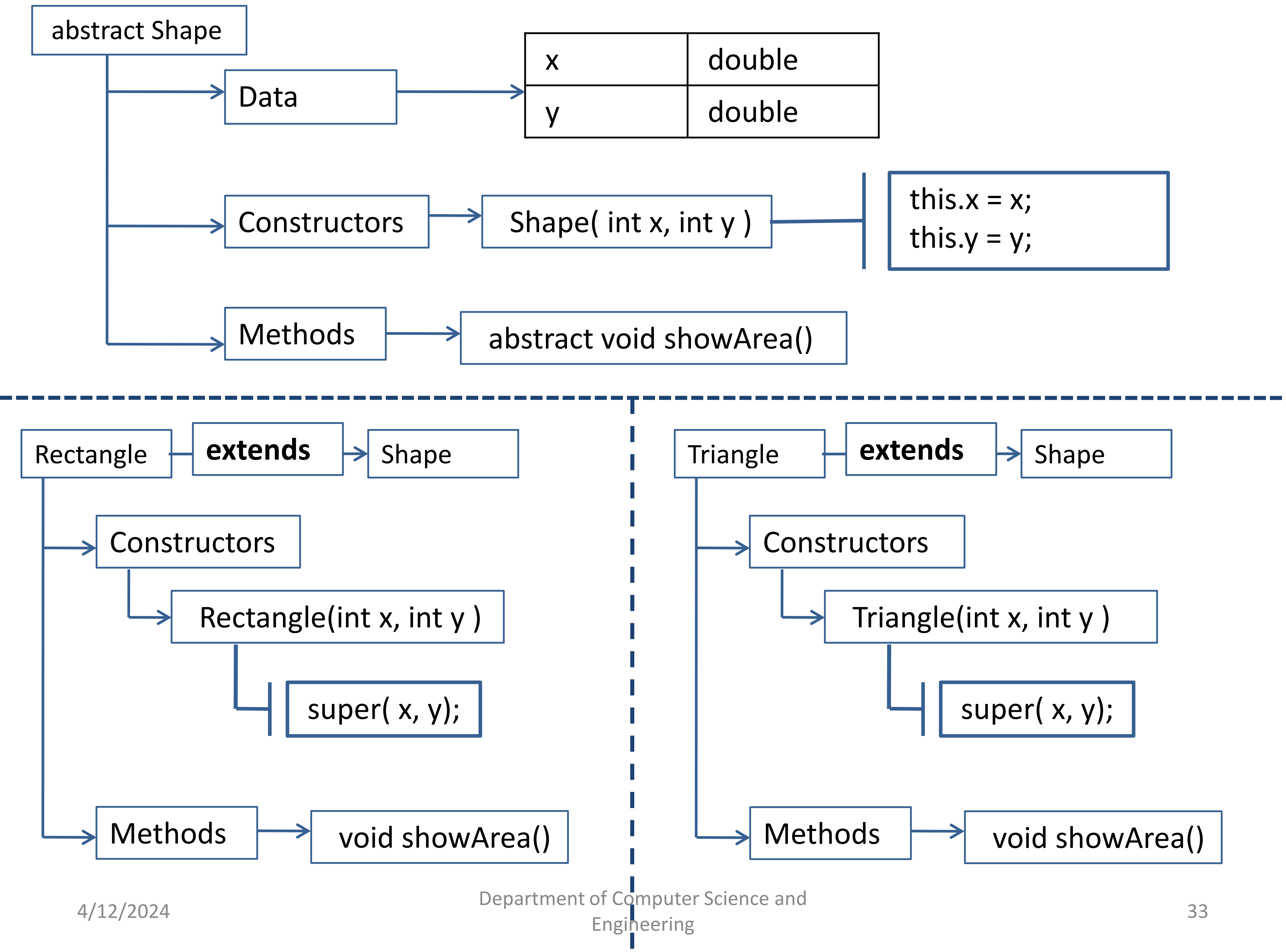
- ❑ An abstract class is a class that is declared abstract—it may or may not include abstract methods.
- ❑ If a class includes at least one abstract method, that class must be declared abstract.
- ❑ The non-abstract methods of an abstract class are **concrete** methods.

```
abstract  class MyClass {
     // declare abstract and non-abstract methods
 }
```

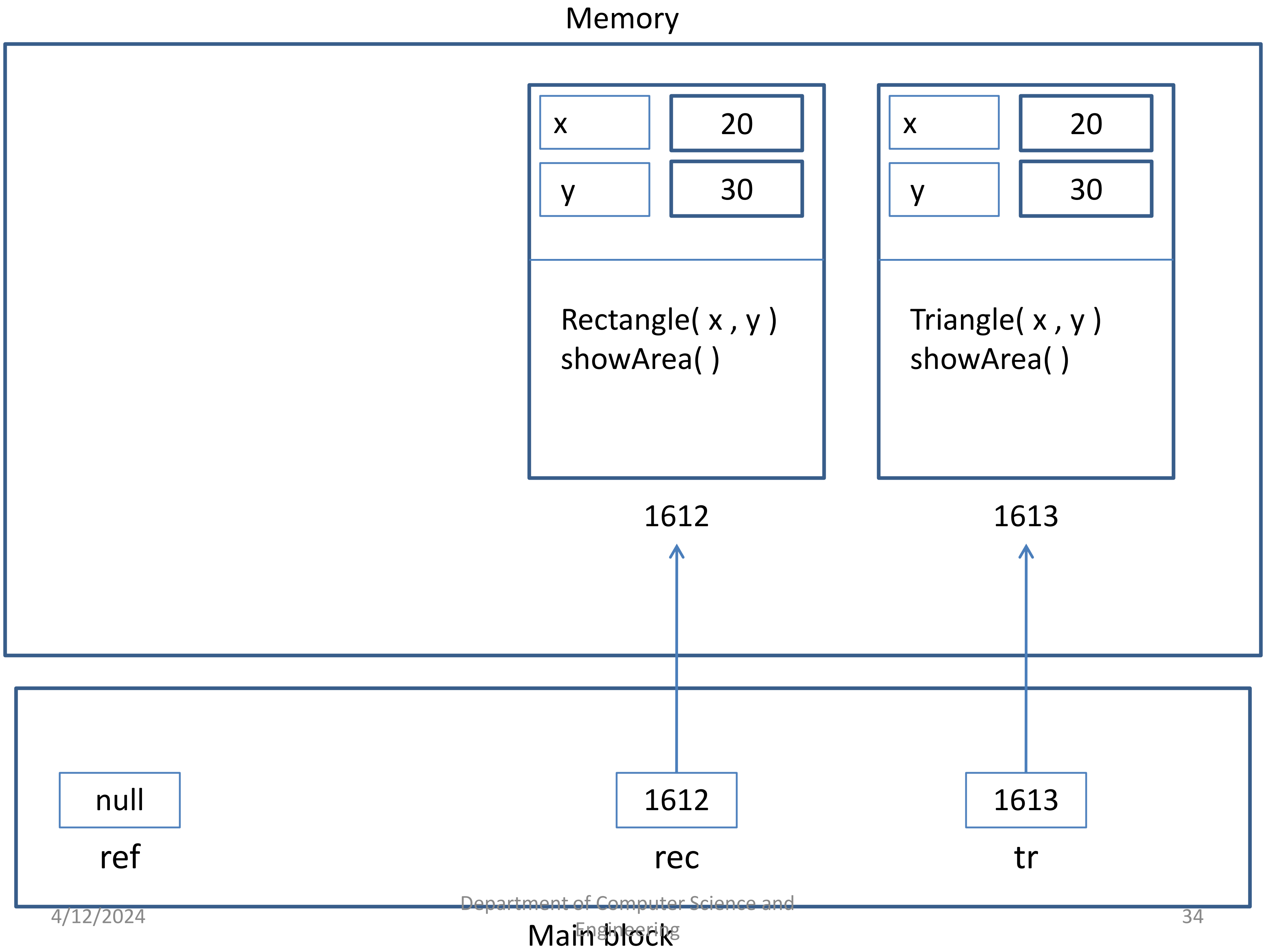
- ❑ Abstract classes cannot be instantiated

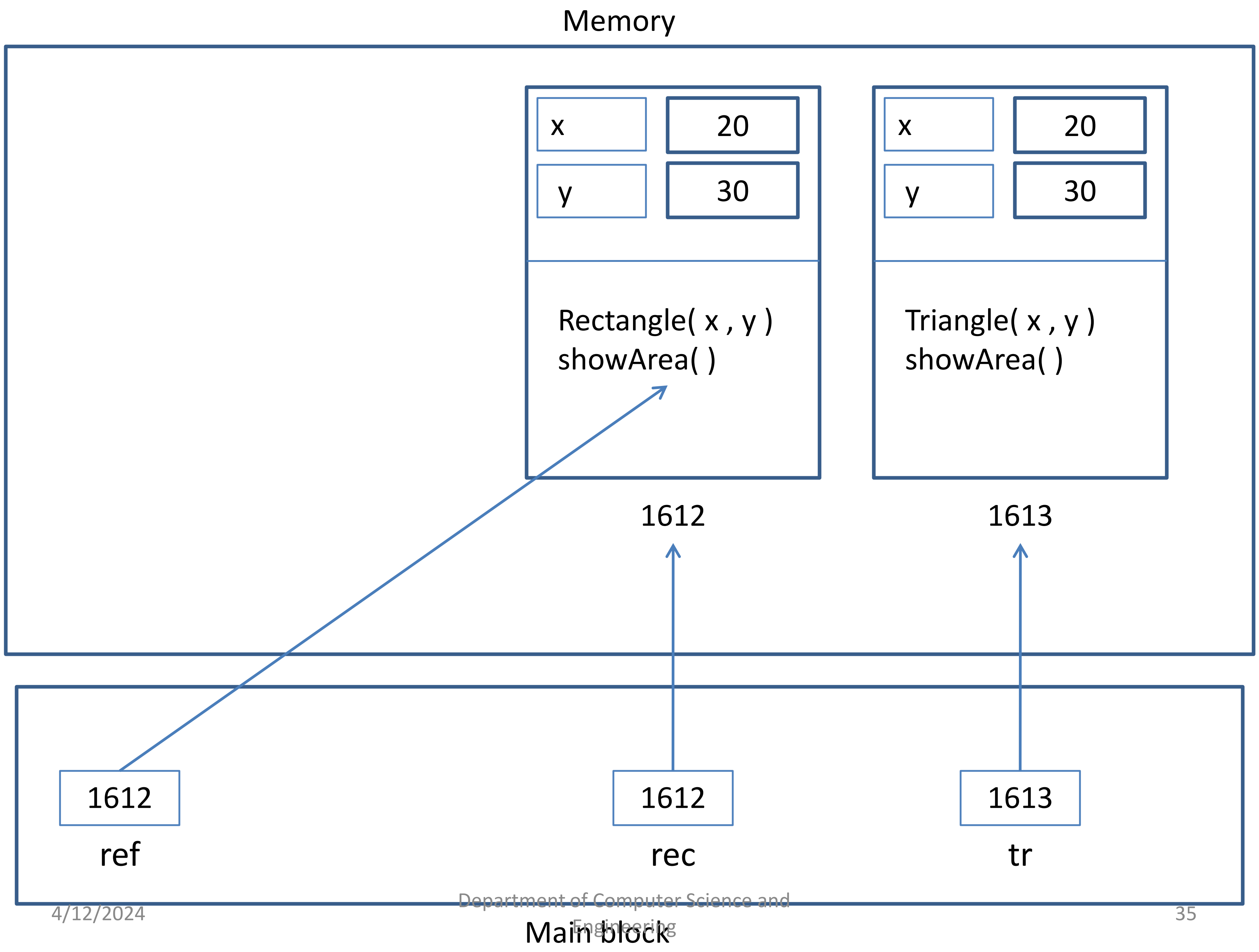
- When an abstract class is sub-classed, the subclass usually provides implementations for all of the abstract methods in its parent class. However, if it does not, the subclass must also be declared abstract. otherwise it generates compile-time error.

```
abstract  class Parent {

       . . . .

}


abstract class Child extends Parent {

       . . . .     //  if not implementing abstract methods

 }
```

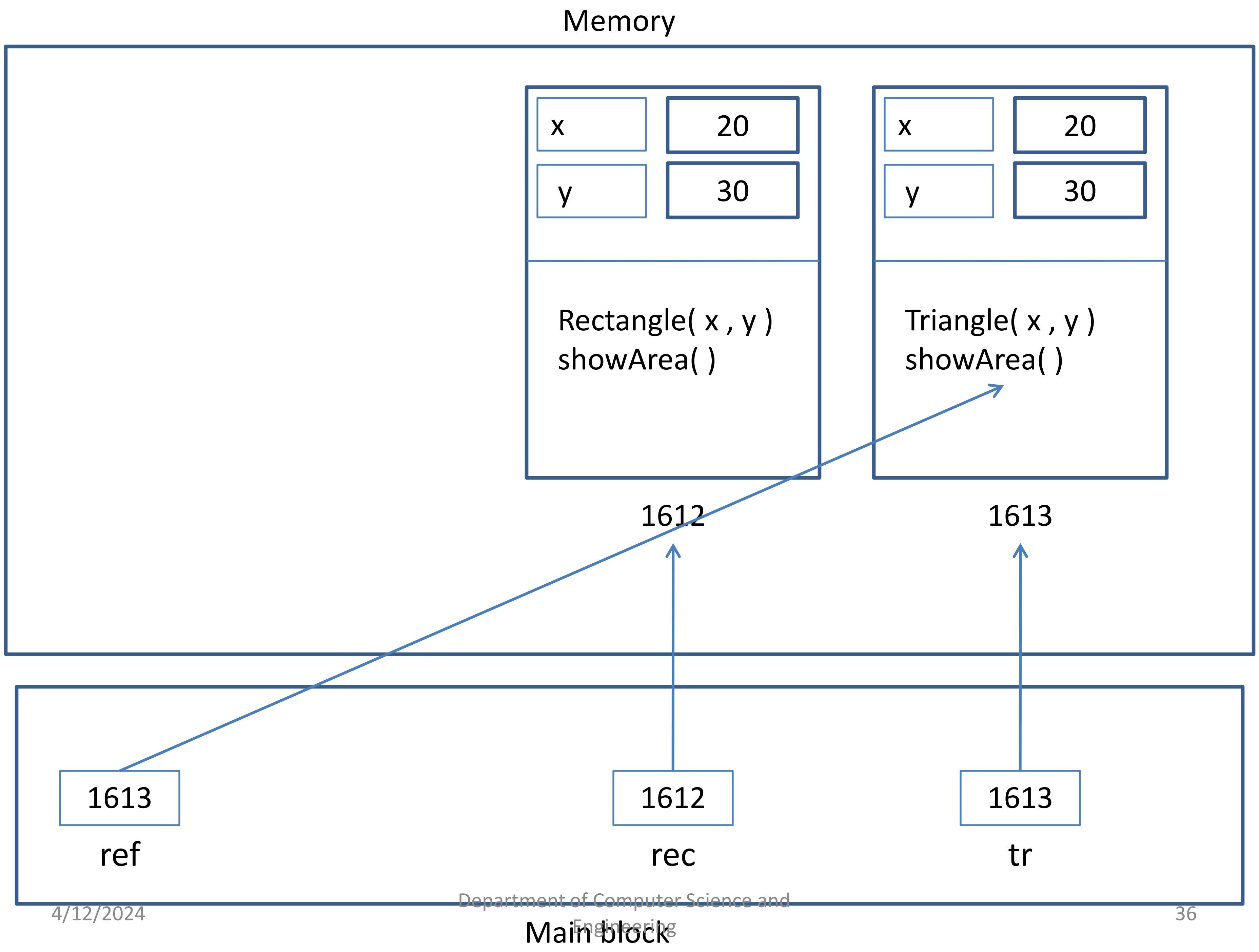


abstract Shape
Data
x
double
y
double
Constructors
Shape(int x, int y)
this.x = x;
this.y = y;
Methods
abstract void showArea()
Rectangle
extends
Shape
Constructors
Rectangle(int x, int y)
super(x, y);
Methods
void showArea()
Triangle
extends
Shape
Constructors
Triangle(int x, int y)
super(x, y);
Methods
void showArea()

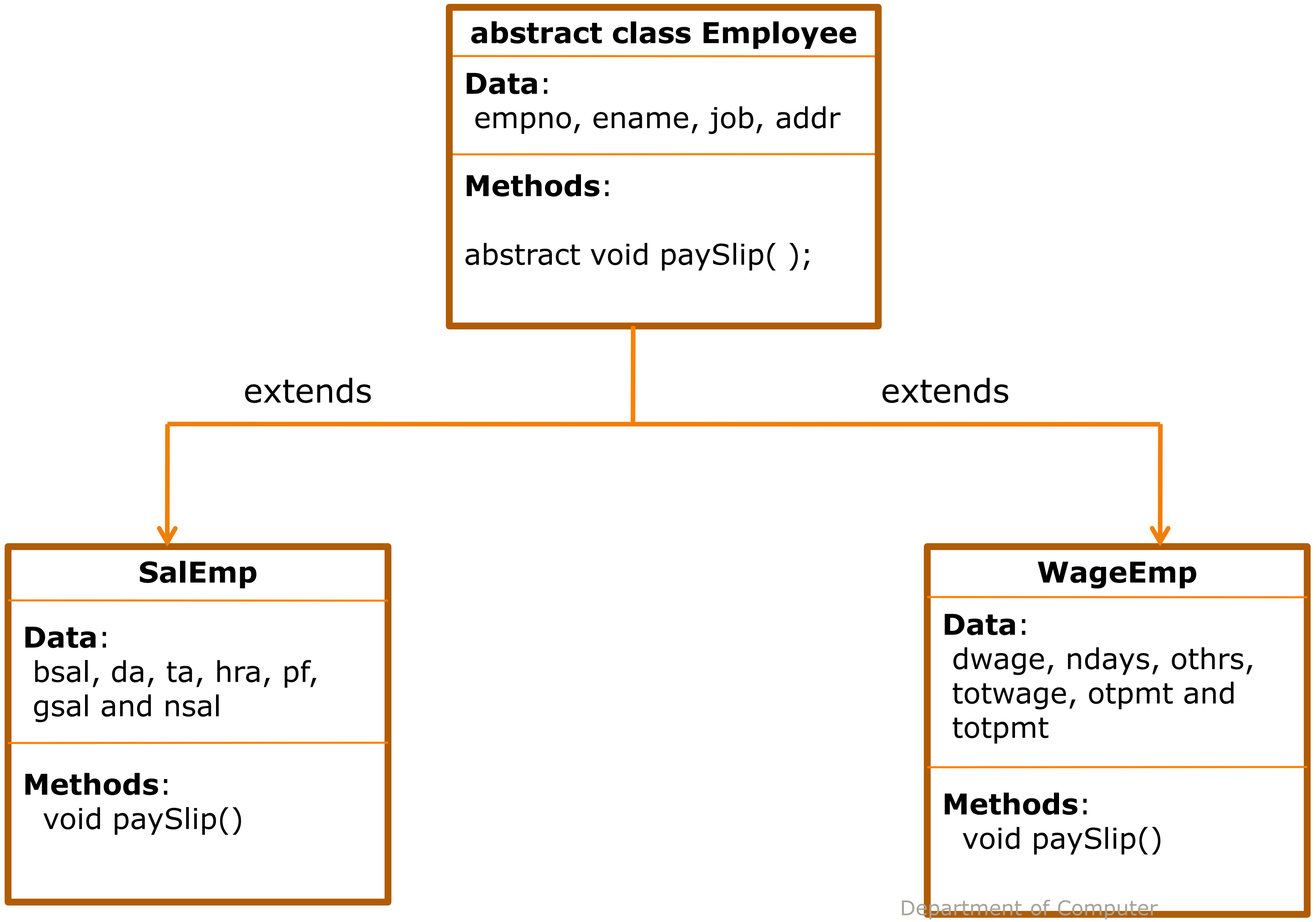

Memory
x
20
y
30
Rectangle(x , y)
showArea( )
x
20
y
30
Triangle(x , y)
showArea( )
1612
1613
null
ref
1612
rec
1613
tr
Main block

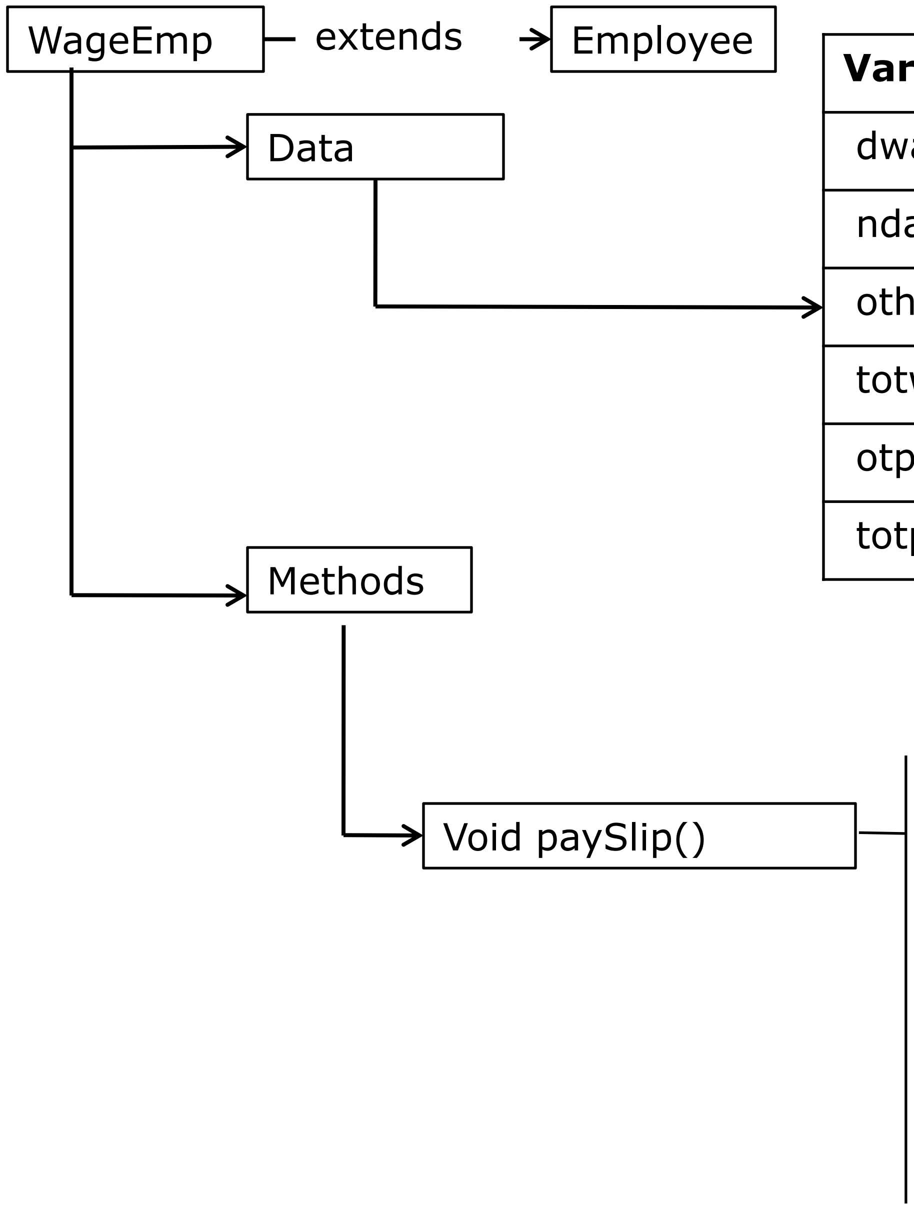


Memory
x
20
y
30
Rectangle(x , y)
showArea( )
x
20
y
30
Triangle(x , y)
showArea( )
1612
1613
1612
ref
1612
rec
1613
tr
Main block



Memory
x
20
y
30
Rectangle(x , y)
showArea( )
x
20
y
30
Triangle(x , y)
showArea( )
1612
1613
1613
ref
1612
rec
1613
tr
Main block


abstract class Employee
Data:
empno, ename, job, addr
Methods:
abstract void paySlip( );
extends
extends
SalEmp
Data:
bsal, da, ta, hra, pf, gsal and nsal
Methods:
void paySlip()
WageEmp
Data:
dwage, ndays, othrs, totwage, otpmt and totpmt
Methods:
void paySlip()



Variables	**Data Type**
dwage	double
ndays (no of days)	int
othrs (overtime hrs)	double
totwage	double
otpmt (overtime pmt)	double
totpmt (total pmt)	double

```
Read:
 dwage, ndays and  othrs

totwage = dwage x ndays
 otpmt = ((dwage / 8) x 2 ) x othrs
 totpmt = totwage + otpmt

Print:  totwage, otpmt, and totpmt
```