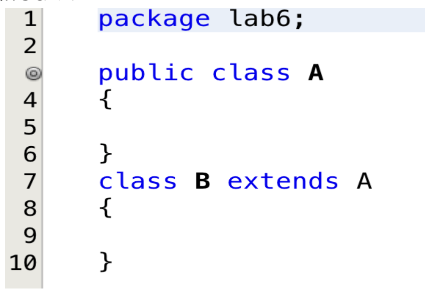
ASDV 2420, Advanced Programming I
Lab 4 – Inheritance
February 7, 2017

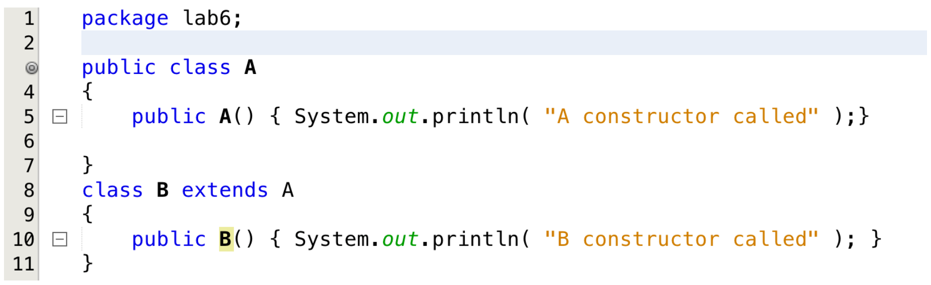
1. Create a new project lab8

1. Create a new class and call it **A**.
2. Derive a subclass from it and call it **B**. You can put 2 or more classes in the same java file as shown below. ONLY ONE class is public, in this case class A. The name of the file A.java requires the the class is public and called **A**



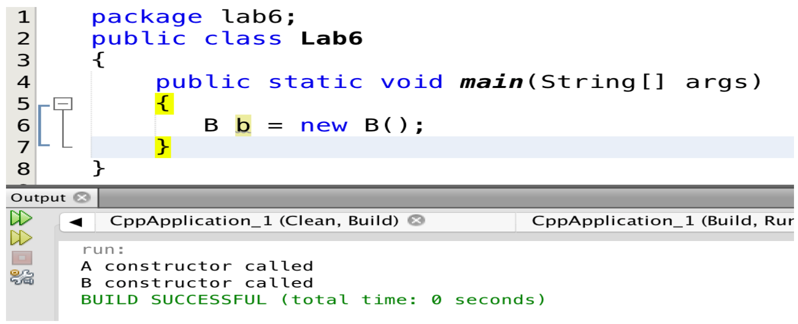
```java
package lab6;

public class A
{

}
class B extends A
{

}
```

3. Add 2 parametereless constructors in both class A and B as shown below:



```java
package lab6;

public class A
{
    public A() { System.out.println( "A constructor called" );}

}
class B extends A
{
    public B() { System.out.println( "B constructor called" ); }
}
```

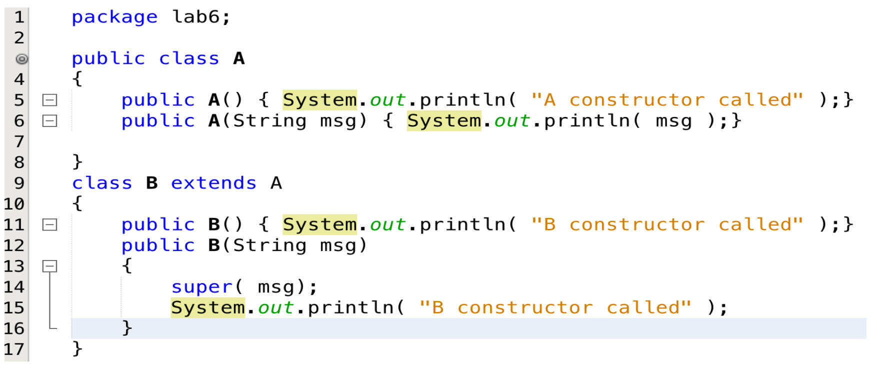
4. Inside the main() of class Lab7 create an object and run lab7.java. Observe that the constructor of the *superclass* **A** was EXECUTED BEFORE the constructor of the *subclass* **B**.



```java
package lab6;
public class Lab6
{
    public static void main(String[] args)
    {
        B b = new B();
    }
}
```

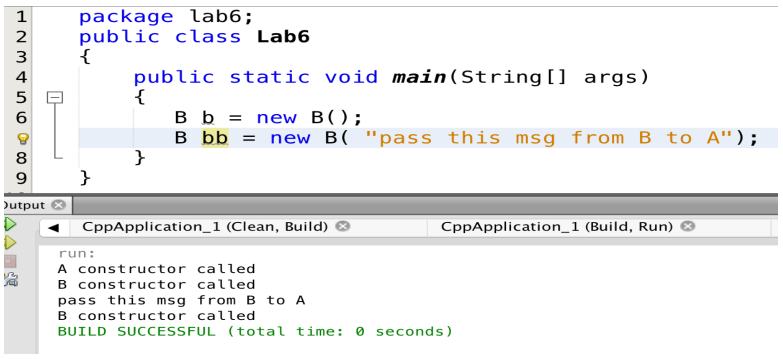
```
run:
A constructor called
B constructor called
BUILD SUCCESSFUL (total time: 0 seconds)
```

5. Add the 2 2-parameters constructors to classes A and B as shown in lines 6 and 12 below. Observe the use of keyword **super in B, calling the appropriate constructor of its superclass A.**



```
 1    package lab6;
 2
      public class A
 4    {
 5        public A() { System.out.println( "A constructor called" );}
 6        public A(String msg) { System.out.println( msg );}
 7
 8    }
 9    class B extends A
10    {
11        public B() { System.out.println( "B constructor called" );}
12        public B(String msg)
13        {
14            super( msg);
15            System.out.println( "B constructor called" );
16        }
17    }
```

6. Use the test-code below and test it. MAKE SURE YOU UNERSTAND what's going on. DON"T just type. If you don't understand ask your partner first, and if still problems of understanding exist, then ask your instructor.

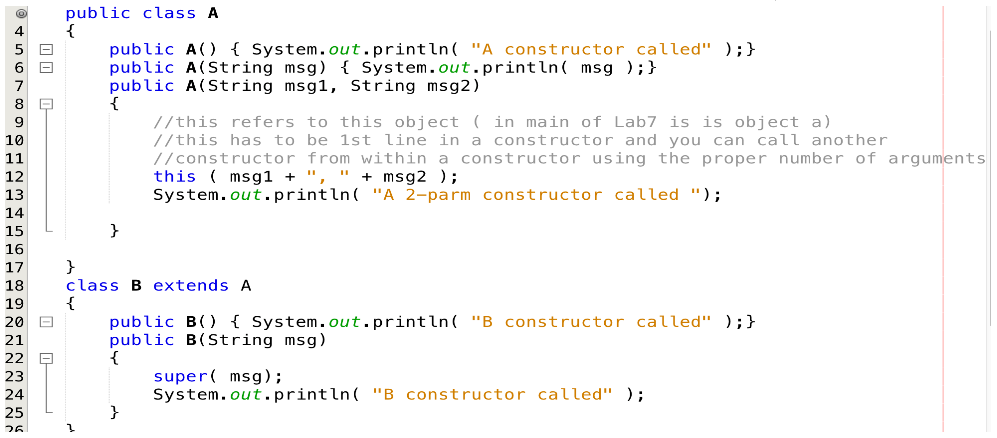


```
1    package lab6;
2    public class Lab6
3    {
4        public static void main(String[] args)
5        {
6            B b = new B();
             B bb = new B( "pass this msg from B to A");
8        }
9    }
```

Output
CppApplication_1 (Clean, Build)   CppApplication_1 (Build, Run)

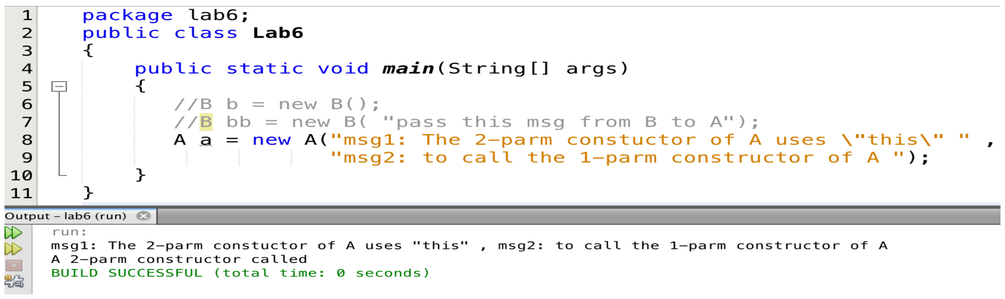
```
run:
A constructor called
B constructor called
pass this msg from B to A
B constructor called
BUILD SUCCESSFUL (total time: 0 seconds)
```

7. Add a 3-parm constructor in class A (lines 7 to 15) . Please type the comments inside the constructor and understand how "this" is used to call a constructor from within a constructor.



```
      public class A
 4    {
 5        public A() { System.out.println( "A constructor called" );}
 6        public A(String msg) { System.out.println( msg );}
 7        public A(String msg1, String msg2)
 8        {
 9            //this refers to this object ( in main of Lab7 is is object a)
10            //this has to be 1st line in a constructor and you can call another
11            //constructor from within a constructor using the proper number of arguments
12            this ( msg1 + ", " + msg2 );
13            System.out.println( "A 2-parm constructor called ");
14
15        }
16
17    }
18    class B extends A
19    {
20        public B() { System.out.println( "B constructor called" );}
21        public B(String msg)
22        {
23            super( msg);
24            System.out.println( "B constructor called" );
25        }
```

8. Call the 3-parm constructor of class A as shown below. Comment out lines shown also so you can have a clearer output.

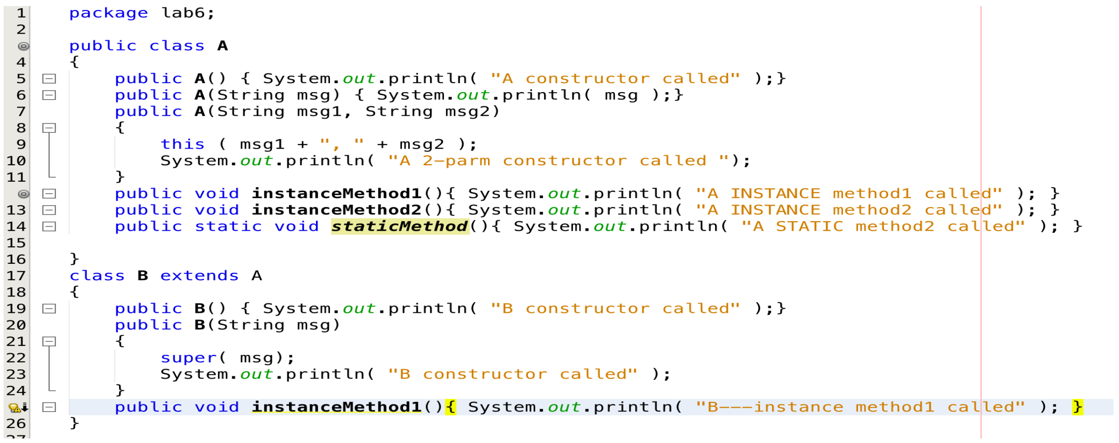


```
package lab6;
public class Lab6
{
    public static void main(String[] args)
    {
      //B b = new B();
      //B bb = new B( "pass this msg from B to A");
      A a = new A("msg1: The 2-parm constuctor of A uses \"this\" " ,
                  "msg2: to call the 1-parm constructor of A ");
    }
}
```

Output - lab6 (run)

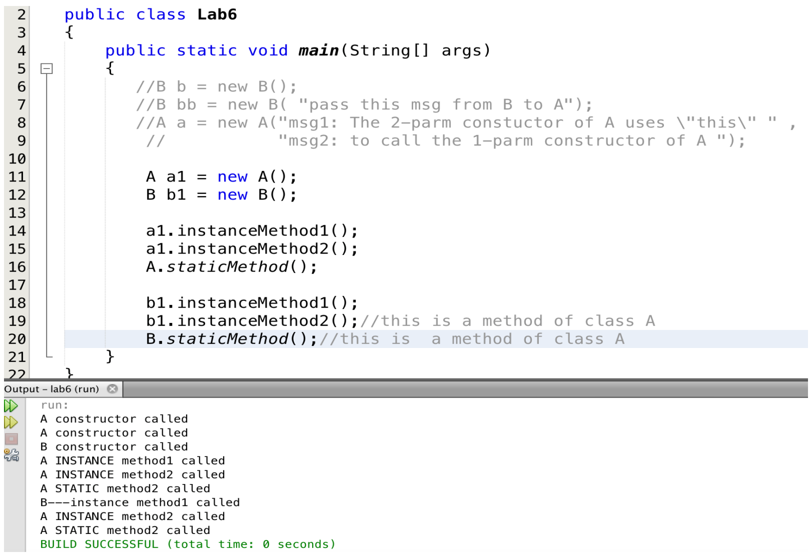
```
run:
msg1: The 2-parm constuctor of A uses "this" , msg2: to call the 1-parm constructor of A
A 2-parm constructor called
BUILD SUCCESSFUL (total time: 0 seconds)
```

9. Add the methods ***instanceMethod1***, ***instanceMethod2*** , ***staticMethod2*** in class *A* and ***instanceMethod1*** in class *B*. As you see instanceMethod1 of class *A* **is *overridden (has the same signatures)*** in class *B*.



```
package lab6;

public class A
{
    public A() { System.out.println( "A constructor called" );}
    public A(String msg) { System.out.println( msg );}
    public A(String msg1, String msg2)
    {
        this ( msg1 + ", " + msg2 );
        System.out.println( "A 2-parm constructor called ");
    }
    public void instanceMethod1(){ System.out.println( "A INSTANCE method1 called" ); }
    public void instanceMethod2(){ System.out.println( "A INSTANCE method2 called" ); }
    public static void staticMethod(){ System.out.println( "A STATIC method2 called" ); }

}
class B extends A
{
    public B() { System.out.println( "B constructor called" );}
    public B(String msg)
    {
        super( msg);
        System.out.println( "B constructor called" );
    }
    public void instanceMethod1(){ System.out.println( "B---instance method1 called" ); }
}
```

10. In the main shown blow, create 2 objects **a1** and **b1** and call the methods as shown. Understand how the call to static and instance methods is done and how the overridden instanceMethod1 of class B HIDES the instanceMethod1 of class A. The *staticMethdod1* of class A is shared by both class A and class B REGARDLESS of how many objects of type A or type B we create using the new operator.

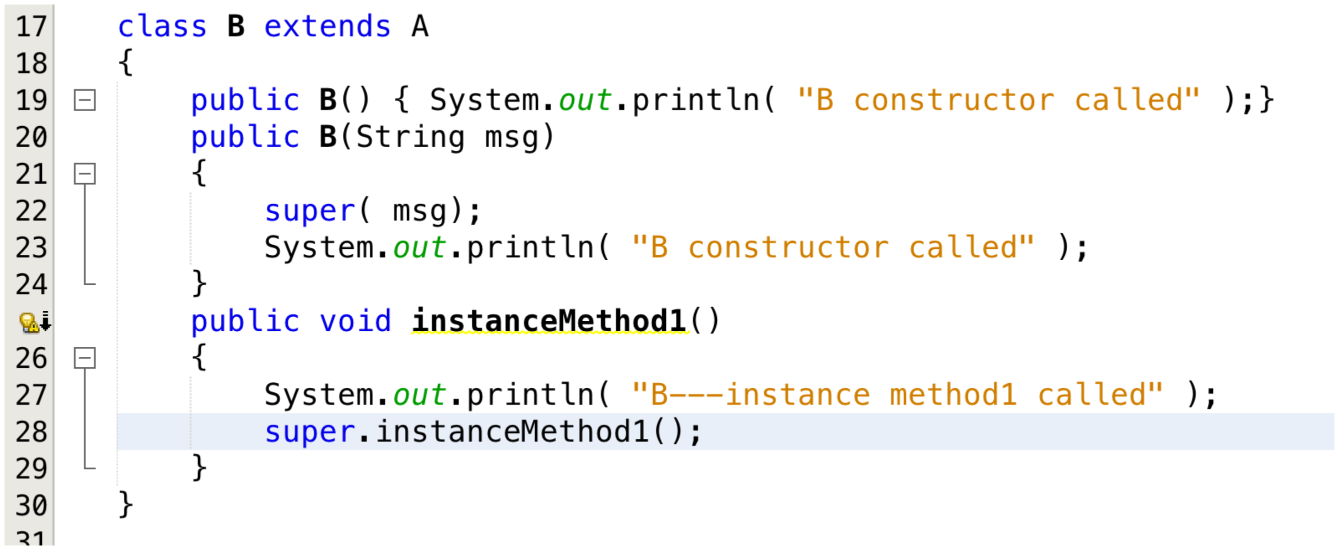


```java
public class Lab6
{
    public static void main(String[] args)
    {
       //B b = new B();
       //B bb = new B( "pass this msg from B to A");
       //A a = new A("msg1: The 2-parm constuctor of A uses \"this\" " ,
        //           "msg2: to call the 1-parm constructor of A ");

        A a1 = new A();
        B b1 = new B();

        a1.instanceMethod1();
        a1.instanceMethod2();
        A.staticMethod();

        b1.instanceMethod1();
        b1.instanceMethod2();//this is a method of class A
        B.staticMethod();//this is  a method of class A
    }
}
```

Output – lab6 (run)

```
run:
A constructor called
A constructor called
B constructor called
A INSTANCE method1 called
A INSTANCE method2 called
A STATIC method2 called
B---instance method1 called
A INSTANCE method2 called
A STATIC method2 called
BUILD SUCCESSFUL (total time: 0 seconds)
```

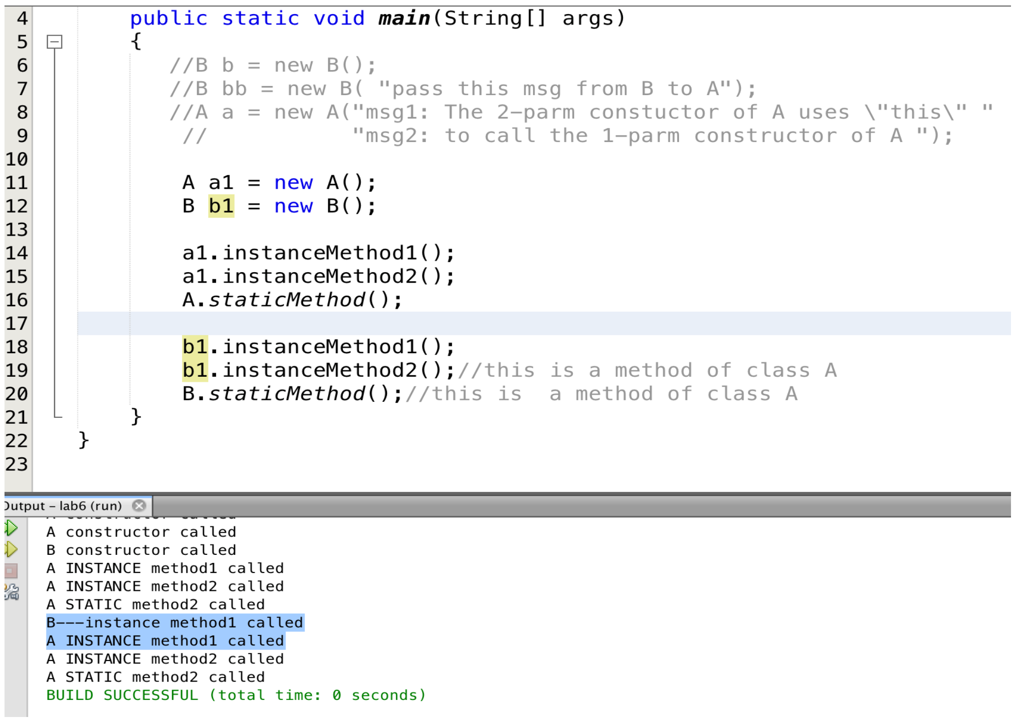
11. Add line 28 to instanceMethod1 of class B. When we user **super** inside an instance method we refer to the super class of the class of the instance method. So all that IS NOT PRIVATE in a super class CAN BE ACCESSED using super in a subclass. The super keyword does NOT have to be the first line as in constructors.



```java
17    class B extends A
18    {
19        public B() { System.out.println( "B constructor called" );}
20        public B(String msg)
21        {
22            super( msg);
23            System.out.println( "B constructor called" );
24        }
          public void instanceMethod1()
26        {
27            System.out.println( "B---instance method1 called" );
28            super.instanceMethod1();
29        }
30    }
```

12. Run the project without modifying the main of Lab7 and observe the additional lines before the last 2 lines of output:

    B---instance method1 called
    A INSTANCE method1 called



```
    public static void main(String[] args)
    {
       //B b = new B();
       //B bb = new B( "pass this msg from B to A");
       //A a = new A("msg1: The 2-parm constuctor of A uses \"this\" "
        //            "msg2: to call the 1-parm constructor of A ");

        A a1 = new A();
        B b1 = new B();

        a1.instanceMethod1();
        a1.instanceMethod2();
        A.staticMethod();

        b1.instanceMethod1();
        b1.instanceMethod2();//this is a method of class A
        B.staticMethod();//this is  a method of class A
    }
}
```

```
A constructor called
B constructor called
A INSTANCE method1 called
A INSTANCE method2 called
A STATIC method2 called
B---instance method1 called
A INSTANCE method1 called
A INSTANCE method2 called
A STATIC method2 called
BUILD SUCCESSFUL (total time: 0 seconds)
```

13. (The Person, Student, Employee, Faculty, and Staff classes) Design a class named Person and its two subclasses named Student and Employee. Make Faculty and Staff subclasses of Employee. A person has a name, address, phone number, and email address. A student has a class status (freshman, sophomore, junior, or senior). Define the status as a constant. An employee has an office, salary, and date hired. Use the Date class to create an object for date hired. A faculty member has office hours and a rank. A staff member has a title. Override the toString method in each class to display the class name and the person's name.

14. (Use ArrayList) Write a class ArraylistGeneric that creates an ArrayList and adds a Person object, a Date object, a string, and a Employee object to the list, and use a loop to display all the elements in the list by invoking the object's toString() method.

15. (Remove duplicates) ADD a STATIC method TO ArraylistGeneric that removes the duplicate elements from an array list of integers using the following header: *public static void removeDuplicate(ArrayList<Integer> list)*
Write a test code in main() method that prompts the user to enter 10 integers to a list and displays the distinct integers separated by exactly one space.

Here is a sample run:

```
Enter ten integers: 34 5 3 5 6 4 33 2 2 4
The distinct integers are 34 5 3 6 4 33 2
```