Part I: Assignments and Casting (1 point)

------------------------------------------

Let Y be a subclass of X, and let y and x be variables of classes Y and X

respectively. From lecture, you know that the assignment "x = y" is valid, but

the assignment "y = (Y) x" requires a cast to compile, and will cause a

run-time error if x references an object that isn't a Y.

What about arrays of objects? Suppose xa is an array of X's, and ya is an

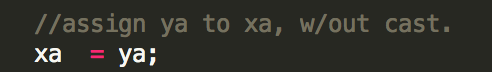
array of Y's.

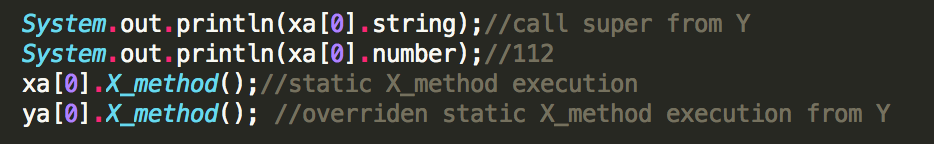
(a) At compile-time, can we assign xa to ya, and vice versa? When is a cast

required?

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

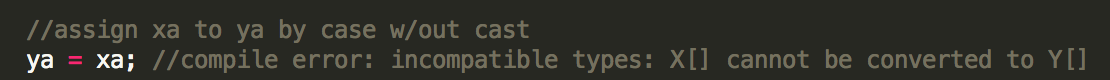
xa=ya; is OK (all y is x)



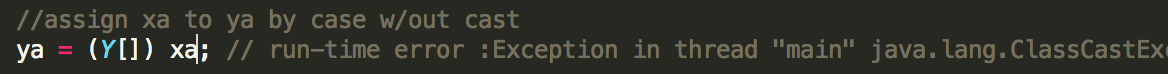


Notice is we call  it still call the method that defined in X Class.

ya=xa; is not OK, Will have compile error. needs to cast (ya=(Y[])xa);

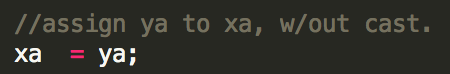


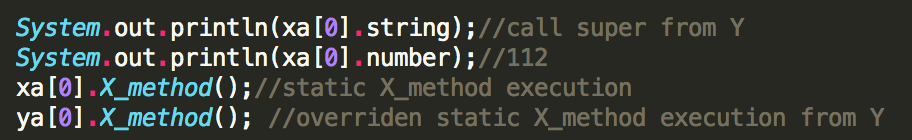
if we cast , no compile-error but will have run-time error



(b) At run-time, if ya references an array of Y's, can we assign it to xa?

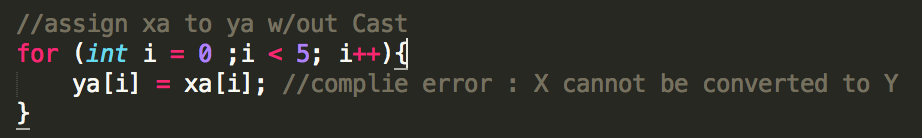
Yes, at run-time we can do xa=ya;



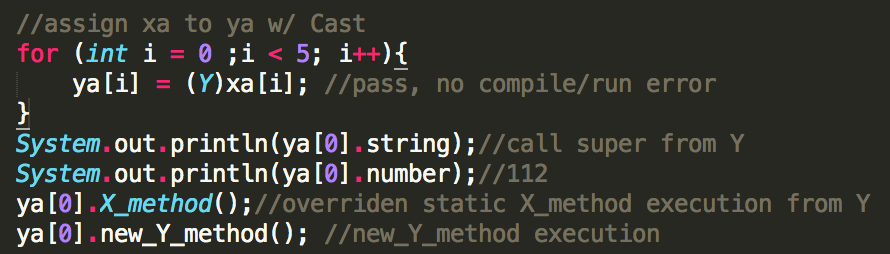


Can we then assign it back from xa to ya?

Yes, Only if we cast;



We need to cast:



notice we can also directly do cast as (the effect is same):



(c) If xa references an array of X's (that are not Y's), can we assign it to

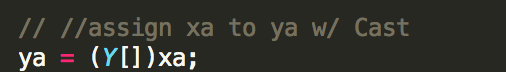
ya? Can we then assign it back from ya to xa? Does it make a difference

if the array of type X[] references objects that are all of class Y? Why

do you think this is the case?

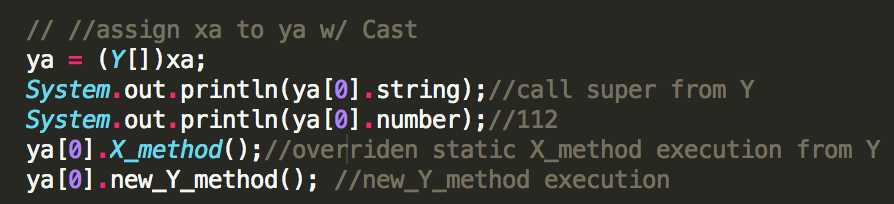
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

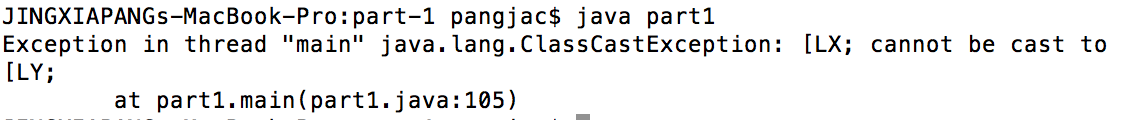
ya=(Y[])xa; will works without compile time error



but it will have ClassCastException if we want to access ya[\*];

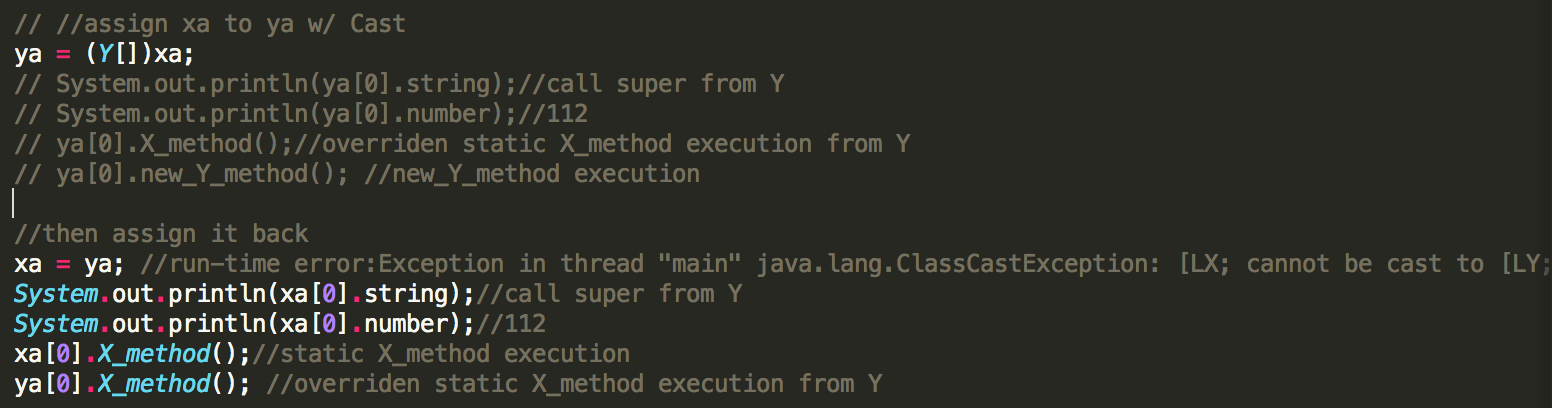
when executing:

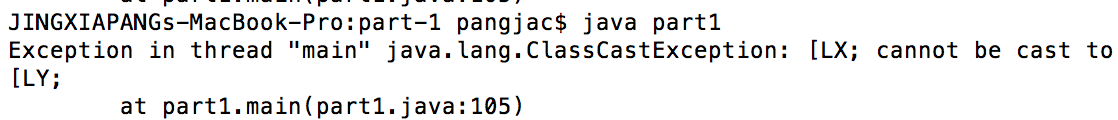




there's no compile error to assign it back;

but there's is run-time error;





there's no difference, because all y are x;

Part II: Conflicting Declarations (1 point)

--------------------------------------------

Suppose a subclass inherits a method implementation from a superclass, and

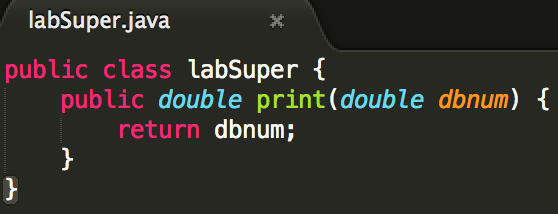
implements a Java interface (that's the "interface" keyword) that contains

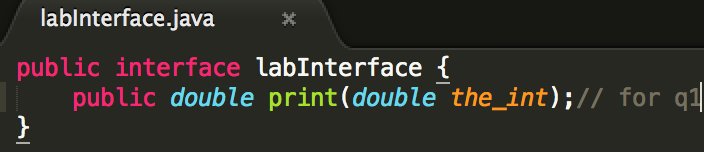
a method with the same name and prototype.

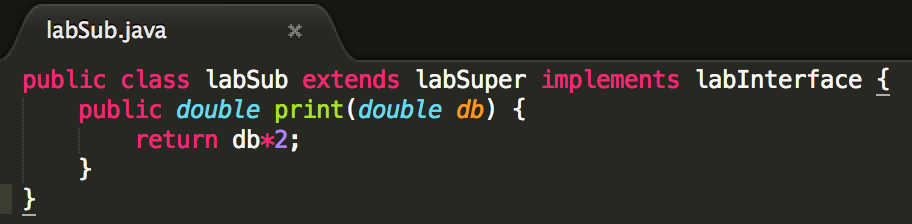
(a) Will Java compile the result?

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Java will compile this result



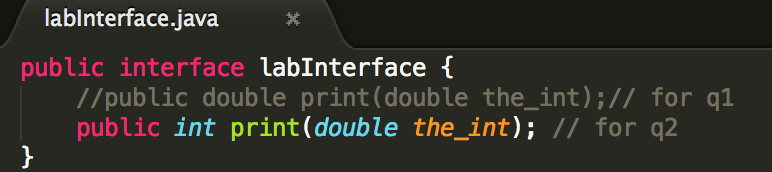




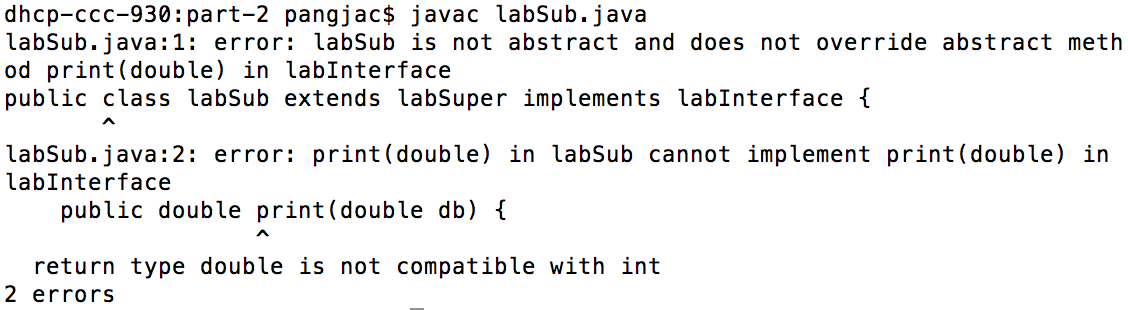
(b) What if the method declaration in the interface has a different return

type?

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

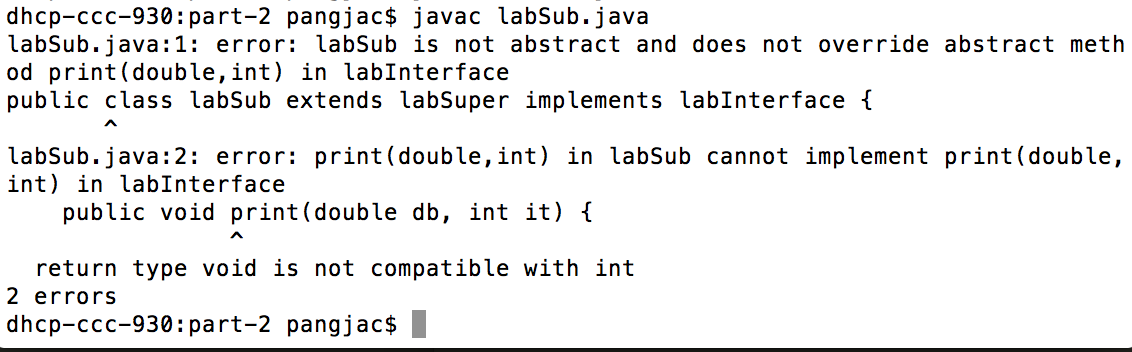


Java won't compile this result, for incompatible return type;



**Because in a class, you can not have 2 function with the same name and**

**parameters but with different return type**!



(c) What if the method declaration in the interface has the same return type,

but a signature with a different parameter type?

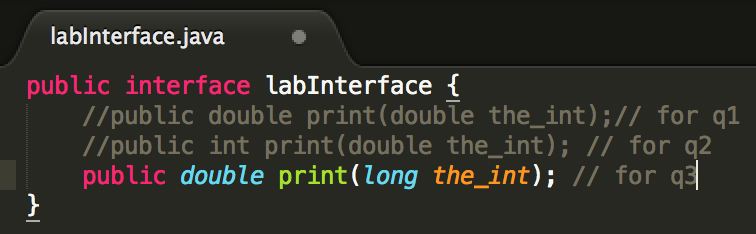
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

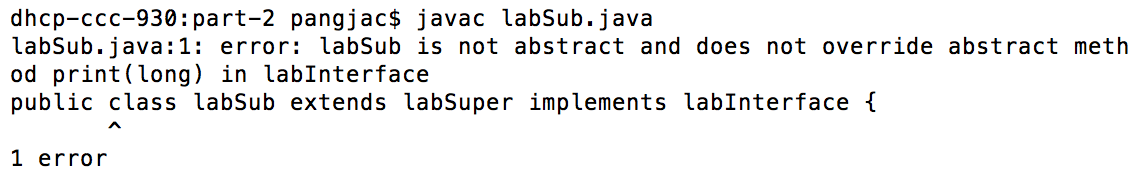
It depends, as long as you implement the method defined in java interface,

there's no problem whether or not you override method in superclass; but if

you leave method in java interface incomplete, it will issue compile time

error;





(d) What if the method declaration in the interface has the same return type,

and the same number of parameters and parameter types, but those

parameters have different names?

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

This is OK, will get compiled

Part III: More Conflicting Declarations (1 point)

--------------------------------------------------

Suppose a subclass inherits a "public static final" constant from a superclass,

and implements a Java interface that contains a "public static final" constant

with the same name.

(a) Will Java compile the result? Does it make any difference whether the

constant in the superclass and the constant in the interface have the

same value?

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

It will compile regardless of whether the values are same or not;

(b) Write a main() method in the subclass that accesses the constant using the

same name used in the superclass and the Java interface. Will Java

compile the result? Does it make any difference whether the constant in

the superclass and the constant in the interface have the same value?

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Java won't compile whether or not they are the same value

(c) Figure out how to modify your main() method so that it accesses and prints

one of the two conflicting values. (Look to Lecture 9 for clues.) Make

sure your code compiles and runs without errors.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Need to specify the full name, including interface or superclass's name

Part IV: Method Overriding (1 point)

-------------------------------------

Consider a subclass that has a method that overrides a method with the same

prototype in its superclass.

(a) Define a variable whose static type is the subclass and which references

an object of the subclass. If we cast the variable to the superclass type

before calling the overridden method

((Superclass) subclassvariable).method();

does Java call the superclass method or the subclass method?

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

It will call the subclass method, this is equivalant to:

Superclass sup = (Superclass)subclassvariable;

sup.method();

(b) Define a variable whose static type is the superclass and which references

an object of the superclass (but not the subclass). If we cast the

variable to the subclass type before calling the method, does Java call

the superclass method or the subclass method?

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Java will compile this but will result in run time error of class cast

exception

(c) Suppose you have an object whose class is the subclass. Can you figure

out a way to call the superclass method on that object without having to

go through the subclass method of the same name?

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Used google!

This is not possible; ortherwise, there's no encapsulation

Reference : https://github.com/ryanzhao/java\_cs61b/tree/master/lab/lab5