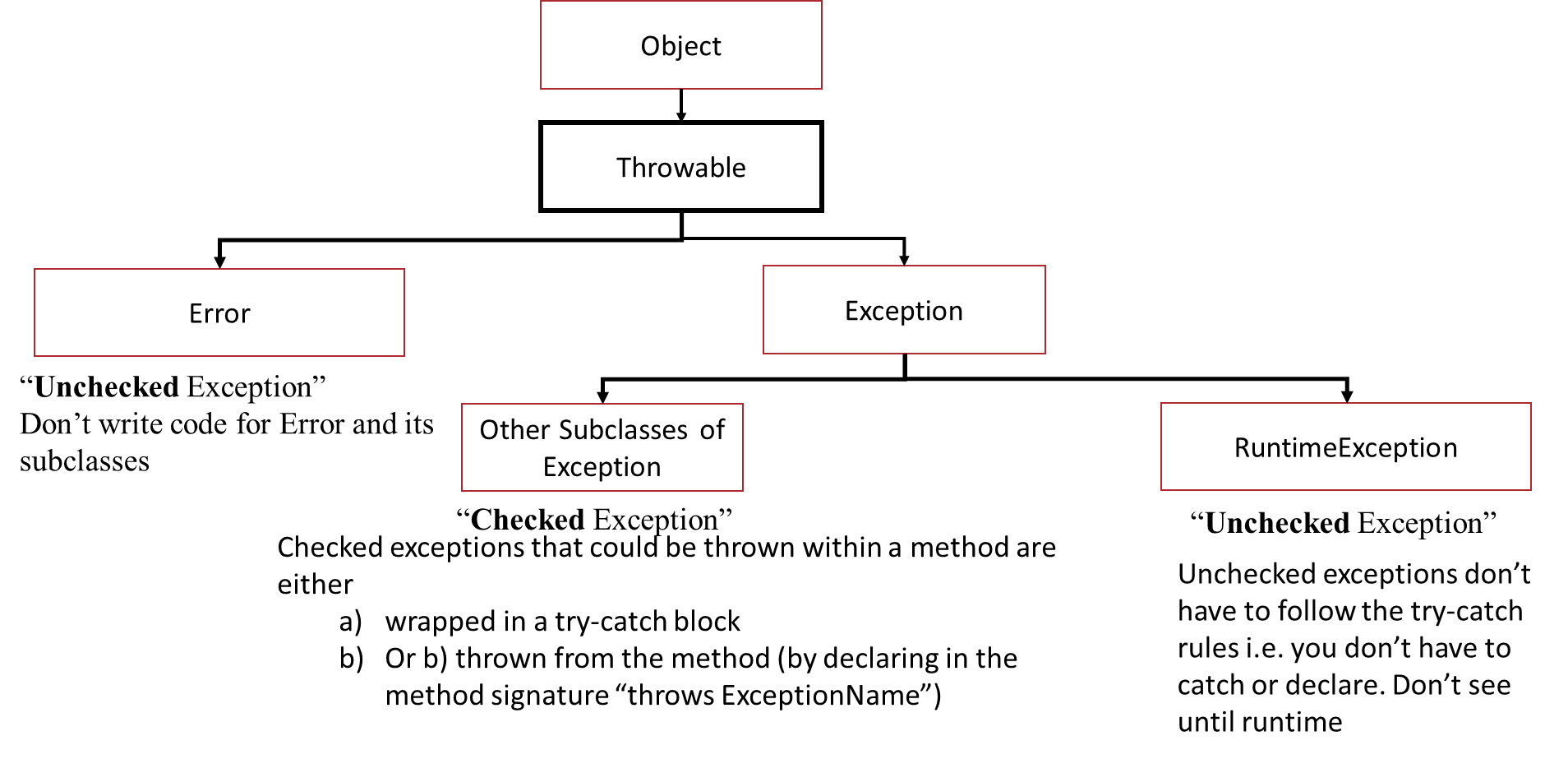
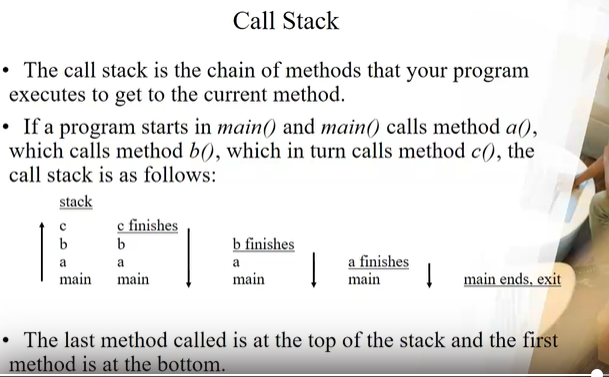
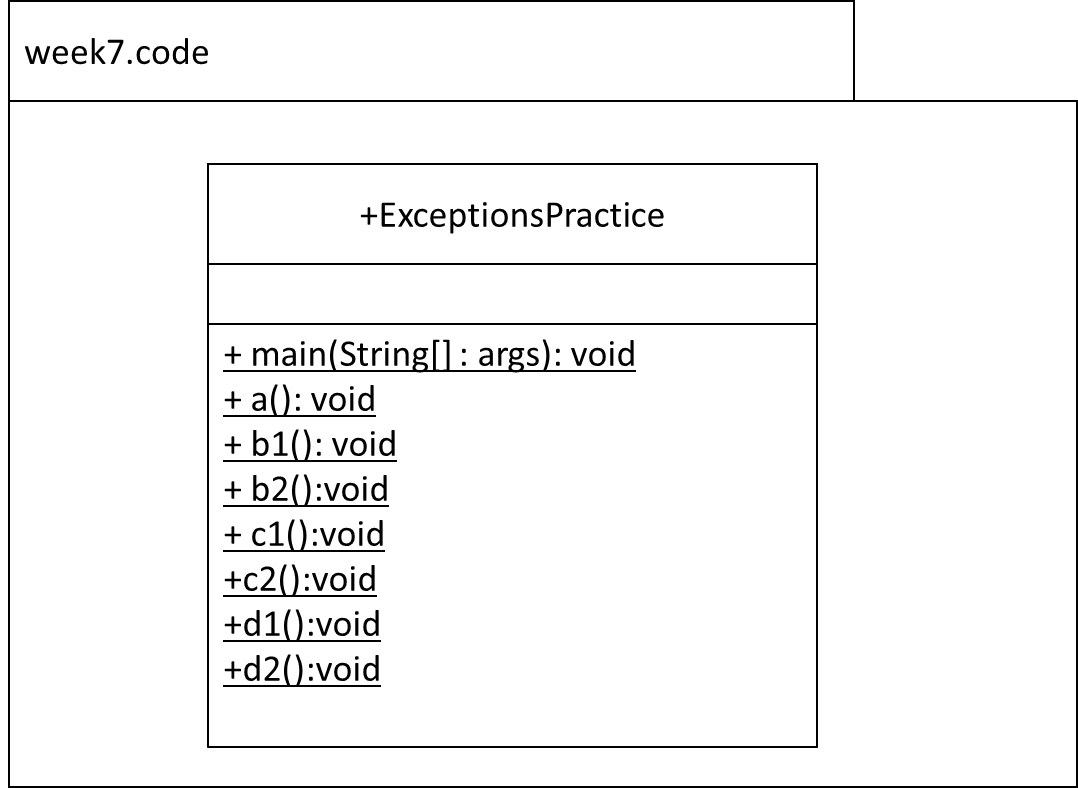
**Background**





**Exception propagation:** Throw exception from one called method to the next called method up the stack. If exception not caught by any method in the stack, the call stack is dumped to screen and program stops running.



~~Q1.~~

~~Code the following:~~

~~In main() call the method a().~~

~~In the method a(), throw an IOException.~~

~~This exception is caught and handled back in the method main().~~

~~Q2.~~

~~Code the following:~~

~~In main() call the method b1()~~

~~In the method b1(), call the method b2().~~

~~In the method b2(), throw an IOException. This exception is caught and handled in the method b1().~~

~~Q3.~~

~~Code the following:~~

~~In main() call the method c1()~~

~~In the method c1(), call the method c2().~~

~~In the method c2(), throw an IOException. This exception is caught and handled in the method main().~~

~~Q4.~~

~~Code the following:~~

~~In main() call the method d1()~~

~~In the method d1(), call the method d2().~~

~~In the method d2(), throw an~~ *~~ClassNotFoundException~~*~~. This exception is not caught in any method in the application.~~

**~~The following questions are taken from the Exceptions lab done earlier in the year. Add methods as outlined in the questions~~**

~~Q5.~~

~~To demonstrate that the (checked) exceptions you are trying to catch must be thrown in the~~ *~~try~~* ~~block you are checking. This question is about compilation and not runtime.~~

~~Code a method named~~ *~~testMethodD~~*~~; return type~~ *~~String~~*~~.~~

~~In the method, code a~~ *~~try~~* ~~block that simply returns “ok”. However, try and catch the following (checked) exceptions:~~ *~~IOException~~*~~,~~ *~~NoSuchMethodException~~* ~~and~~ *~~ClassNotFoundException~~*~~. Will they compile? why or why not? Catch~~ *~~RuntimeExeption~~* ~~– will it compile? why or why not? Catch~~ *~~Exception~~* ~~– will it compile? why or why not?~~

~~Q6.~~

~~This example demonstrates the “impact” of throwing~~ ***~~unchecked~~*** ~~exceptions~~

~~Code four methods~~ *~~Y1~~*~~,~~ *~~Y2~~*~~,~~ *~~Y3~~* ~~and~~ *~~Y4~~*~~. These methods take no parameters and return nothing. In~~ *~~main~~*~~() call~~ *~~Y1~~*~~;~~ *~~Y1~~*~~() then calls~~ *~~Y2~~*~~();~~ *~~Y2~~*~~() calls~~ *~~Y3~~*~~() and~~ *~~Y3~~* ~~calls~~ *~~Y4~~*~~(). In~~ *~~Y4~~*~~() throw a~~ *~~RuntimeException~~* ~~(for demo purposes). This exception is to be caught in~~ *~~main~~*~~() only. What is the impact?~~

~~Note that in~~ *~~main(), Y1~~*~~() with~~ **~~no~~** ~~surrounding~~ *~~try~~* ~~catch block will compile…~~

~~Q7.~~

~~Create the custom exception class~~ *~~MyException~~*~~.~~

~~Call method~~ *~~a~~*~~() from~~ *~~main()~~*~~;~~ *~~a~~*~~() calls~~ *~~b~~*~~() and~~ *~~b~~*~~() calls~~ *~~c~~*~~().~~

~~In~~ *~~c~~*~~() throw an~~ *~~IOException~~* ~~with the message “~~*~~An I/O exception occurred~~*~~”. Note that, the signature of~~ *~~c~~*~~() says that it “~~*~~throws MyException~~*~~” (and not~~ *~~IOException~~*~~).~~

~~In~~ *~~b()~~*~~, catch the~~ *~~MyException~~* ~~generated by~~ *~~c~~*~~() and in this exception handler, output the exception caught i.e.~~ *~~MyException~~* ~~and details of the original exception that was generated in~~ *~~c~~*~~() i.e.~~ *~~IOException~~*~~.~~