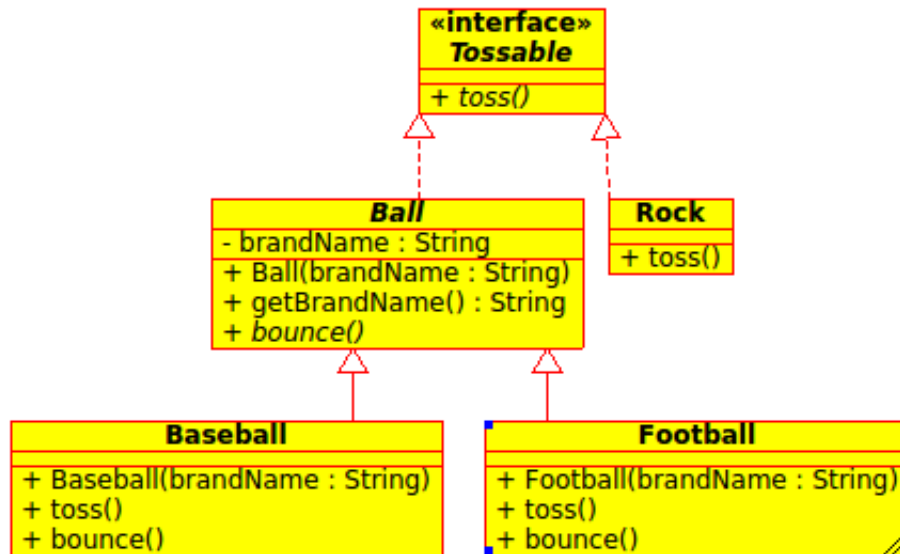


Interfaces and Abstract Classes



1. Fill in each cell of the table with one of three values:

- Y An object of this type could be assigned to a variable of this type.
- N An object of this type could *not* be assigned to a variable of this type.
- – It is not possible to instantiate an object of this type.

		Object Type				
		Tossable	Ball	Rock	Baseball	Football
Variable Type	Tossable					
	Ball					
	Rock					
	Baseball					
	Football					

2. Write the source code for the UML diagram.

- In *Rock.java*, the `toss` method should print "Tossing a Rock!".
- In *Baseball.java*, the `toss` method should print "Tossing a Baseball!", and the `bounce` method should print "Bouncing a Baseball!".
- In *Football.java*, the `toss` method should print "Tossing a Football!", and the `bounce` method should print "Bouncing a Football!".

3. Indicate whether each code snippet will:

- **N** – not compile;
- **X** – compile but generate an exception at run-time; or
- **R** – compile and run without generating an exception.

	Code Snippet	Result
a)	<code>Ball ball = new Football("Spalding");</code>	
b)	<code>Ball ball = new Football("Spalding"); Baseball baseball = (Baseball) ball;</code>	
c)	<code>Object obj = new Baseball("Spalding");</code>	
d)	<code>Object obj = new Baseball("Spalding"); Tossable tossable = obj;</code>	
e)	<code>Tossable tossable = new Baseball("Spalding"); Object obj = tossable;</code>	
f)	<code>Tossable tossable = new Baseball("Spalding"); tossable.getBrandName();</code>	