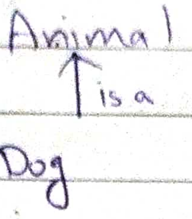
* "Instanceof" method from java.lang package lets you find out if an object belongs to the that class or not. here class can be Superclass or interface.
while using instance of if we check with Superclass then still it will return true
eg ~~Animal~~ Animal

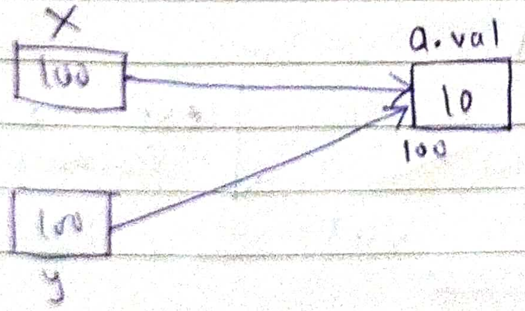


Dog is a animal
∴ if (dog instanceOf animal) = true
if (dog instance of animal) = true
because dog is Subclass of animal and every dog is animal.

Object Identity

two objects are identical if they refer to the same instance

eg



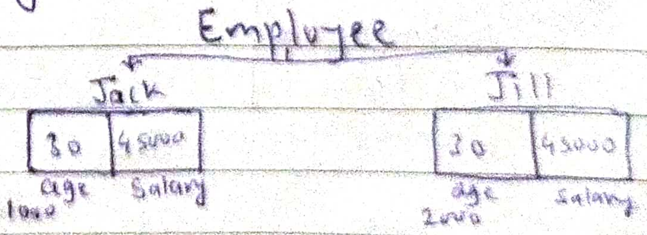
x ~~Jack~~ & y Jill are equal as well as identical

identical has to be equal

object equality

two objects are equal if they Contain Same value and if they are of same class

eg.



Jack & Jill are not identical ~~but~~ since 1000 != 2000 but they are equal

Equal need not to be identical

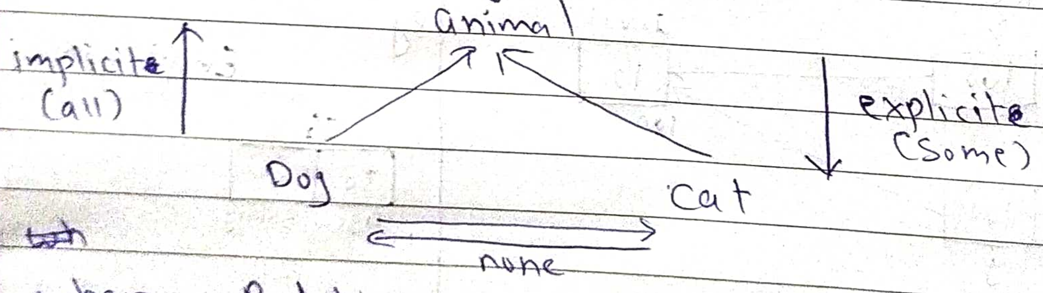
Java.lang is autoimported package in java.

object equality can be checked by inheriting hashcode() and equals()
"X.hashCode() == Y.hashCode() && X.equals(Y)"
equality can only be checked by X.equals(Y) but if we have 20 fields inside a class then only checking X.equals(Y) will be time consuming thats why we use hashcode() if hashcode are not equal it will not go for equals().

In hashcode we are performing Some hashing function and the comparing both

In equals we are checking if both objects belong from same class using instanceOf() method and then we are checking each field by "==" operator.

* Rules for TypeCasting



~~both~~
1) when Relation is implicite no need of type casting
2) When Relation is explicite we have to mention type casting
3) When there is no inheritance (for eg. dog & cat) type casting is not possible

this keyword refers to the current object in a method or constructor.

* Exception Handling.

$$\begin{tikzcd}
 & \text{Exception} \arrow[dl] \arrow[dr] & \\
\text{Checked} & & \text{Unchecked}
\end{tikzcd}$$

Checked

1) checked exception are checked by compiler whether programmar has handled them or not
2) If its not ~~true~~ handled. compiler will give "unreported exception"

Unchecked.

1) compiler will not check if these exceptions are handled by programmar or not.
2) Compiler will not force you to handle these but it is advisable to handle

Two methods to handle exception
1) try... catch
2) Throws keyword :- not advisable.

Throw :- used to throw userdefined (custom) Exception

Throws :- used to show that the method is going to throw exception and if caller of that method doesn't handle that exception the compiler will force you to handle it.

while declaring any exception it is prefered that ~~extends~~ inherit exception from 'Runtime Exception' instead of 'Exception'

making checked exception was mistake by java so only focus on unchecked exception

* Try Catch
1) one try block can have multiple catch block and one finally block

2) Try to maintain heirarchy while using catch block

3) Finally block can contain clean up code. So if catch block also have exception then still finally will get execute and terminates program.