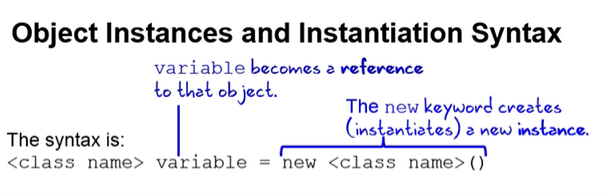
Introduction:

****

**\*\*\* finalize()** method :

This method is called just before an object is garbage collected. It is called by the [Garbage Collector](http://www.geeksforgeeks.org/garbage-collection-java/) on an object when garbage collector determines that there are no more references to the object. We should override finalize() method to dispose system resources, perform clean-up activities and minimize memory leaks. For example before destroying Servlet objects web container, always called finalize method to perform clean-up activities of the session.  
**Note :**finalize method is called just **once** on an object even though that object is eligible for garbage collection multiple times.

|  |
| --- |
| // Java program to demonstrate working of finalize()  public class Test  {      public static void main(String[] args)      {          Test t = new Test();          System.out.println(t.hashCode());            t = null;            // calling garbage collector          System.gc();            System.out.println("end");      }        @Override      protected void finalize()      {          System.out.println("finalize method called");      }  } |

Run on IDE

Output:

366712642

end

finalize method called

**Type casting:**

• Casting from a subclass to a superclass

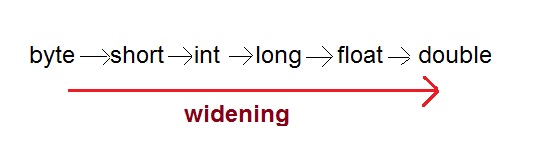
• Casting from a superclass to a subclass

In the case of casting from a subclass to a superclass, we can cast either implicitly or explicitly. **Implicit casting** simply means **auto casting**; **explicit casting** means that we have to provide the class type in parentheses. Casting from subclass to superclass is completely reliable because subclasses contain information about their super classes.

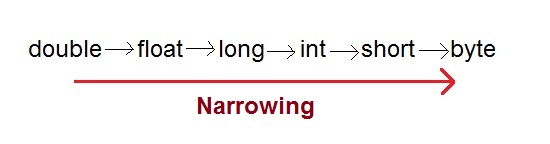
Casting from a superclass to a subclass has to be done explicitly and is also not reliable because the compiler has no idea whether the class being cast to is a subclass of the superclass or not.

In Java, type casting is classified into two types,

* Widening Casting(Implicit)



* Narrowing Casting(Explicitly done)



**OBJECT CASTING:**

Upcasting and downcasting are important part of Java, which allow us to build complicated programs using simple syntax, and gives us great advantages, like Polymorphism or grouping different objects. ***Java permits an object of a subclass type to be treated as an object of any superclass type. This is called upcasting. Upcasting is done automatically, while downcasting must be manually done by the programmer***, and i'm going to give my best to explain why is that so.

Upcasting and downcasting are NOT like casting primitives from one to other, and i believe that's what causes a lot of confusion, when programmer starts to learn casting objects.

***Polymorphism: All methods in java are virtual by default. That means that any method can be overridden when used in inheritance, unless that method is declared as final or static***.

You can see the example below how getType(); works according to the object(Dog,Pet,Police Dog) type.

Assume you have three dogs

1. Dog - This is the super Class.
2. Pet Dog - Pet Dog extends Dog.
3. Police Dog - Police Dog extends Pet Dog.
4. public class Dog{
5. public String getType () {
6. System.out.println("NormalDog");
7. return "NormalDog";
8. }
9. }
10. /\*\*
11. \* Pet Dog has an extra method dogName()
12. \*/
13. public class PetDog extends Dog{
14. public String getType () {
15. System.out.println("PetDog");
16. return "PetDog";
17. }
18. public String dogName () {
19. System.out.println("I don't have Name !!");
20. return "NO Name";
21. }
22. }
23. /\*\*
24. \* Police Dog has an extra method secretId()
25. \*/
26. public class PoliceDog extends PetDog{
27. public String secretId() {
28. System.out.println("ID");
29. return "ID";
30. }
31. public String getType () {
32. System.out.println("I am a Police Dog");
33. return "Police Dog";
34. }

}

Polymorphism : All methods in java are virtual by default. That means that any method can be overridden when used in inheritance, unless that method is declared as final or static.(Explanation Belongs to Virtual Tables Concept)

Virtual Table / Dispatch Table : An object's dispatch table will contain the addresses of the object's dynamically bound methods. Method calls are performed by fetching the method's address from the object's dispatch table. The dispatch table is the same for all objects belonging to the same class, and is therefore typically shared between them.

public static void main (String[] args) {

/\*\*

\* Creating the different objects with super class Reference

\*/

Dog obj1 = new Dog();

` /\*\*

\* Object of Pet Dog is created with Dog Reference since

\* Upcasting is done automatically for us we don't have to worry about it

\*

\*/

Dog obj2 = new PetDog();

` /\*\*

\* Object of Police Dog is created with Dog Reference since

\* Upcasting is done automatically for us we don't have to worry

\* about it here even though we are extending PoliceDog with PetDog

\* since PetDog is extending Dog Java automatically upcast for us

\*/

Dog obj3 = new PoliceDog();

}

obj1.getType();

Prints Normal Dog

obj2.getType();

Prints Pet Dog

obj3.getType();

Prints Police Dog

**Downcasting need to be done by the programmer manually**

When you try to invoke the secretID(); method on obj3 which is PoliceDog object but referenced to Dog which is a super class in the hierarchy it throws error since obj3 don't have access to secretId() method.***In order to invoke that method you need to Downcast that obj3 manually to*** PoliceDog

( (PoliceDog)obj3).secretID();

which prints ID

In the similar way to invoke the dogName();method in PetDog class you need to downcast obj2to PetDog since obj2 is referenced to Dog and don't have access to dogName(); method

( (PetDog)obj2).dogName();

Why is that so, that upcasting is automatical, but downcasting must be manual? Well, you see, upcasting can never fail. But if you have a group of different Dogs and want to downcast them all to a to their types, then there's a chance, that some of these Dogs are actually of different types i.e., PetDog, PoliceDog, and process fails, by throwing ClassCastException.

This is the reason you need to ***downcast your objects manually*** if you have referenced your objects to the super class type.

Note: Here by referencing means you are not changing the memory address of your ojects when you downcast it it still remains same you are just grouping them to particular type in this case Dog

**GETTING STARTED WITH WEB PROJECT:**

* **After server is started, then it will open browser but will show 403 error**

**So give file name in at the end of url**

[**http://localhost:8080/japplication/”NewFile.html**](http://localhost:8080/japplication/)**”**

**args[0].charAt(0);-**

* **it will take first character that is entered in argument line of command line**

public class Test {

public static void main(String args[]){

char grade = args[0].charAt(0);

switch(grade)

{

case 'A' :

System.out.println("Excellent!");

break;

case 'B' :

case 'C' :

System.out.println("Well done");

break;

case 'D' :

System.out.println("You passed");

case 'F' :

System.out.println("Better try again");

break;

default :

System.out.println("Invalid grade");

}

System.out.println("Your grade is " + grade);

}

}

Compile and run above program using various command line arguments. This would produce following result:

$ java Test a

Invalid grade

Your grade is a a

$ java Test A

Excellent!

Your grade is a A

$ java Test C

Well done

Your grade is a C

$

5251966

Satya012

[sai.kottisa.osv@fedex.com](redir.aspx?C=rDyBHfxYk0-Q6QbQVW4FbpYR3-ZDndQItZZhW7m0W2HrX8ARtdkOJpiRHY-3vHtSqZts7EOmexY.&URL=mailto%3asai.kottisa.osv%40fedex.com)

**Tomcat setup:**

* while starting of tomcat server through cmd prompt if authentication is not being done properly when accessing manager-app then add (roles="admin-gui,standard,manager-gui") in <user> tag.

**Sonar Setup:**

when starting of sonar in cmd prompt using ‘StartSonar’ cmd if error occurs like “WrapperSimpleApp: Encountered an error running main:java.lang.RuntimeException: Failed to reset file system

java.lang.RuntimeException: Failed to reset file system

then restarting PC will resolve the issue.