Java Type Casting

<https://www.quora.com/What-is-upcasting-in-Java>

## **Java Type Casting**

Type casting is when you assign a value of one primitive data type to another type.

In Java, there are two types of casting:

* **Widening Casting** (automatically) (upcasting )- converting a smaller type to a larger type size  
  byte -> short -> char -> int -> long -> float -> double
* **Narrowing Casting** (manually)(downcasting) - converting a larger type to a smaller size type  
  double -> float -> long -> int -> char -> short -> byte

## **Widening Casting**

Widening casting is done automatically when passing a smaller size type to a larger size type:

### **Example**

public class MyClass {

public static void main(String[] args) {

int myInt = 9;

double myDouble = myInt; // Automatic casting: int to double

System.out.println(myInt); // Outputs 9

System.out.println(myDouble); // Outputs 9.0

}

}

## **Narrowing Casting(downcasting)**

Narrowing casting must be done manually by placing the type in parentheses in front of the value:

### **Example**

public class MyClass {

public static void main(String[] args) {

double myDouble = 9.78;

int myInt = (int) myDouble; // Manual casting: double to int

System.out.println(myDouble); // Outputs 9.78

System.out.println(myInt); // Outputs 9

}

**Upcasting:**

If the reference variable of Parent class refers to the object of Child class, it is known as upcasting(Widening).

Upcasting is casting a subtype to a supertype.

Here we can call the methods defined/declared in class ‘Parent’ but during runtime, it will call overridden methods of the child class.

If the method is not overridden in child’s class then only parent’s method which will be inherited to the child will be called.

But the same is not applicable to variables because variables decision happens at a compile time, so always Parent class variables will be accessed (not child’s inherited variables).

**Example:**

1. class Parent{
2. int x=5;
3. void show(){
4. System.out.println("parent");
5. }
6. void ShowP(){
7. System.out.println("Parent-Show");
8. }
9. }
10. class Child extends Parent{
11. int x=10;
12. void show(){
13. System.out.println("child");
14. }
15. void ShowC(){
16. System.out.println("Child-Show");
17. }
18. }
19. public class ParentChild {
20. public static void main(String[] args) {
21. Parent p = new Child();
22. p.show();
23. p.ShowP();
24. System.out.println(p.x);
25. }
26. }

**Output**

Child-Show

Parent-Show

5

show() is a method and overridden in a child so child’s method is called.

ShowP() is a method not overridden but inherited to the child and called as it is.

But look at the variable output, x is 5 in parent and 10 in the child, even though the child has overridden ‘x’ but the variable of a parent is chosen.

Mammal is the super interface:

|  |  |
| --- | --- |
|  | public interface Mammal {      public void eat();        public void move();        public void sleep();  } |

Animal is the abstract class:

|  |  |
| --- | --- |
|  | public abstract class Animal implements Mammal {      public void eat() {          System.out.println("Eating...");      }        public void move() {          System.out.println("Moving...");      }        public void sleep() {          System.out.println("Sleeping...");      }    } |

Dog and Cat are the two concrete sub classes:

|  |  |
| --- | --- |
|  | public class Dog extends Animal {      public void bark() {          System.out.println("Gow gow!");      }      public void eat() {          System.out.println("Dog is eating...");      }  }    public class Cat extends Animal {      public void meow() {          System.out.println("Meow Meow!");      }  } |

## What is Upcasting in Java?

***Upcasting***is casting a subtype to a supertype, upward to the inheritance tree. Let’s see an example:

|  |  |
| --- | --- |
|  | Dog dog = new Dog();  Animal anim = (Animal) dog;  anim.eat(); |

Here, we cast the Dog type to the Animal type. Because Animal is the supertype of Dog, this casting is called upcasting.

Note that the actual object type does not change because of casting. The Dog object is still a Dog object. Only the reference type gets changed. Hence the above code produces the following output:

|  |  |
| --- | --- |
|  | Dog is eating… |

Upcasting is always safe, as we treat a type to a more general one. In the above example, an Animal has all behaviors of a Dog.

This is also another example of upcasting:

|  |  |
| --- | --- |
|  | Mammal mam = new Cat();  Animal anim = new Dog(); |

## What is Downcasting in Java?

***Downcasting***is casting to a subtype, downward to the inheritance tree. Let’s see an example:

|  |  |
| --- | --- |
| 1  2 | Animal anim = new Cat();  Cat cat = (Cat) anim; |

Here, we cast the Animal type to the Cat type. As Cat is subclass of Animal, this casting is called downcasting.

Unlike upcasting, downcasting can fail if the actual object type is not the target object type. For example:

|  |  |
| --- | --- |
| 1  2 | Animal anim = new Cat();  Dog dog = (Dog) anim; |

This will throw a ClassCastException because the actual object type is Cat. And a Cat is not a Dog so we cannot cast it to a Dog.