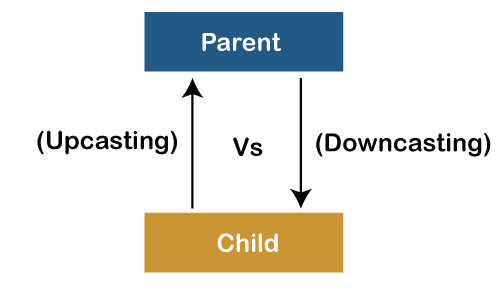
Upcasting and Downcasting in Java

A process of converting one data type to another is known as **Typecasting** and **Upcasting** and **Downcasting** is the type of object typecasting. In Java, the object can also be typecasted like the datatypes. **Parent** and **Child** objects are two types of objects. So, there are two types of typecasting possible for an object, i.e., **Parent to Child** and **Child to Parent** or can say **Upcasting** and **Downcasting**.

In [Java](https://www.javatpoint.com/java-tutorial), the object can also be typecasted like the datatypes. **Parent** and **Child objects** are two types of objects. So, there are two types of typecasting possible for an object, i.e., **Parent to Child** and **Child to Parent** or can say **Upcasting** and **Downcasting**.

**Typecasting** is used to ensure whether variables are correctly processed by a function or not. In **Upcasting** and **Downcasting**, we typecast **a child object to a parent object** and **a parent object to a child object** simultaneously. We can perform Upcasting implicitly or explicitly, but downcasting cannot be implicitly possible.



Let's dive into deep of both these type of object casting:

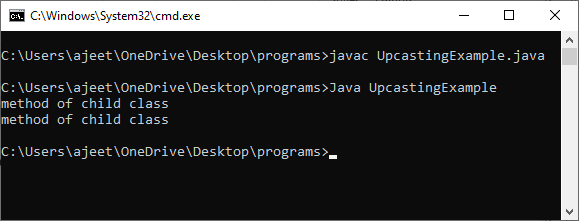
1) Upcasting

**Upcasting** is a type of object typecasting in which a **child object** is typecasted to a **parent class object**. By using the Upcasting, we can easily access the variables and methods of the parent class to the child class. Here, we don't access all the variables and the method. We access only some specified variables and methods of the child class. **Upcasting** is also known as **Generalization** and **Widening**.

**UpcastingExample.java**

1. **class**  Parent{
2. **void** PrintData() {
3. System.out.println("method of parent class");
4. }
5. }
7. **class** Child **extends** Parent {
8. **void** PrintData() {
9. System.out.println("method of child class");
10. }
11. }
12. **class** UpcastingExample{
13. **public** **static** **void** main(String args[]) {
15. Parent obj1 = (Parent) **new** Child();
16. Parent obj2 = (Parent) **new** Child();
17. obj1.PrintData();
18. obj2.PrintData();
19. }
20. }

**Output:**



2) Downcasting

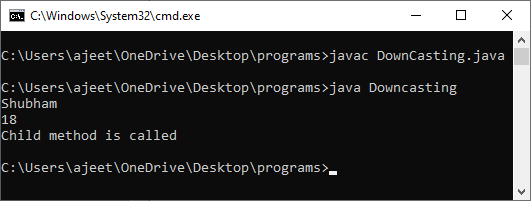
**Upcasting** is another type of object typecasting. In Upcasting, we assign a parent class reference object to the child class. In Java, we cannot assign a parent class reference object to the child class, but if we perform downcasting, we will not get any compile-time error. However, when we run it, it throws the **"ClassCastException"**. Now the point is if downcasting is not possible in Java, then why is it allowed by the compiler? In Java, some scenarios allow us to perform downcasting. Here, the subclass object is referred by the parent class.

Below is an example of downcasting in which both the valid and the invalid scenarios are explained:

**DowncastingExample.java**

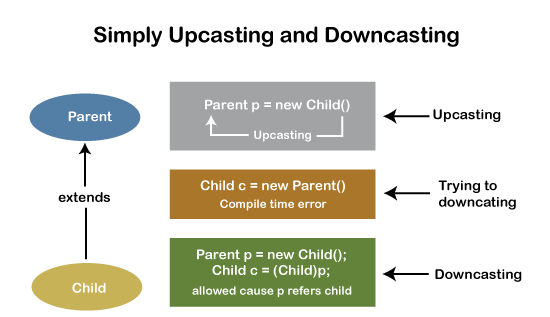
1. //Parent class
2. **class** Parent {
3. String name;
5. // A method which prints the data of the parent class
6. **void** showMessage()
7. {
8. System.out.println("Parent method is called");
9. }
10. }
12. // Child class
13. **class** Child **extends** Parent {
14. **int** age;
16. // Performing overriding
17. @Override
18. **void** showMessage()
19. {
20. System.out.println("Child method is called");
21. }
22. }
24. **public** **class** Downcasting{
26. **public** **static** **void** main(String[] args)
27. {
28. Parent p = **new** Child();
29. p.name = "Shubham";
31. // Performing Downcasting Implicitly
32. //Child c = new Parent(); // it gives compile-time error
34. // Performing Downcasting Explicitly
35. Child c = (Child)p;
37. c.age = 18;
38. System.out.println(c.name);
39. System.out.println(c.age);
40. c.showMessage();
41. }
42. }

**Output:**



Why we need Upcasting and Downcasting?

In Java, we rarely use **Upcasting**. We use it when we need to develop a code that deals with only the parent class. **Downcasting** is used when we need to develop a code that accesses behaviors of the child class.



Difference between Upcasting and Downcasting

These are the following differences between Upcasting and Downcasting:

|  |  |  |
| --- | --- | --- |
| **S.No** | **Upcasting** | **Downcasting** |
| 1. | A child object is typecasted to a parent object. | The reference of the parent class object is passed to the child class. |
| 2. | We can perform Upcasting implicitly or explicitly. | Implicitly Downcasting is not possible. |
| 3. | In the child class, we can access the methods and variables of the parent class. | The methods and variables of both the classes(parent and child) can be accessed. |
| 4. | We can access some specified methods of the child class. | All the methods and variables of both classes can be accessed by performing downcasting. |
| 5. | Parent p = new Parent() | Parent p = new Child() Child c = (Child)p; |