* Upcasting Vs Down casting

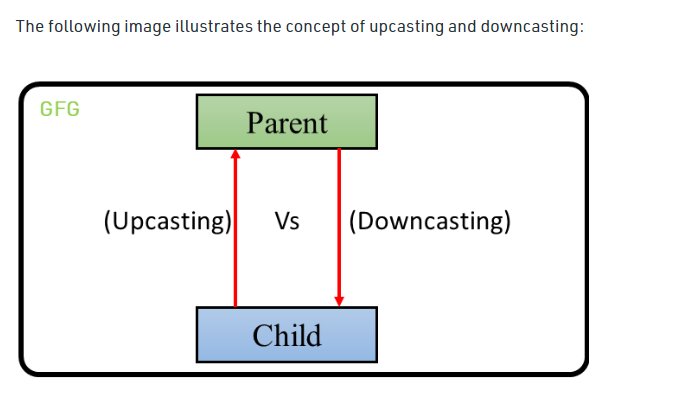
Typecasting is one of the most important concepts which basically deals with the conversion of one data type to another datatype implicitly or explicitly.

In this article, the concept of typecasting for objects is discussed.   
Just like the data types, the objects can also be typecasted.

However, in objects, there are only two types of objects, i.e., parent object and child object. Therefore, typecasting of objects basically means that one type of object (i.e.) child or parent to another. There are two types of typecasting.

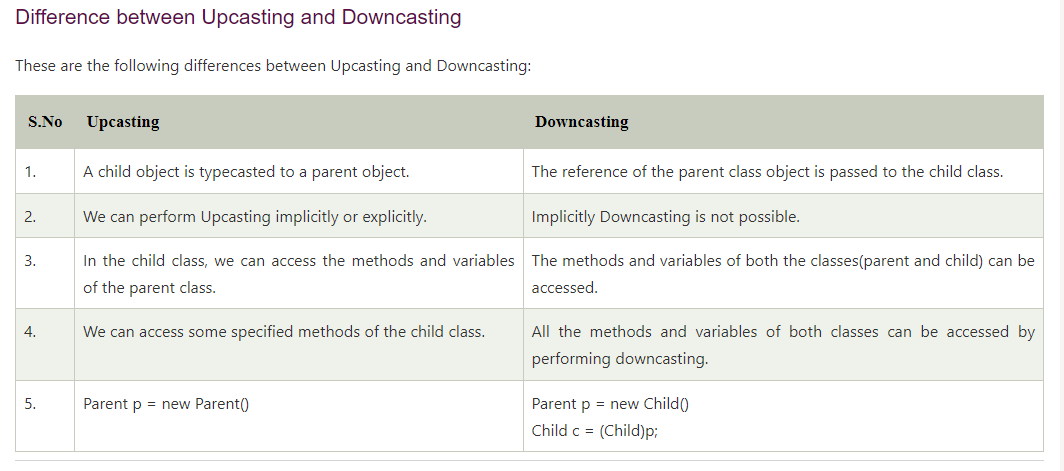
They are:

1. **Upcasting:** Upcasting is the [typecasting](https://www.geeksforgeeks.org/type-conversion-java-examples/) **of a child object to a parent object**. Upcasting can be done implicitly. Upcasting gives us the flexibility to access the parent class members, but it is not possible to access all the child class members using this feature. Instead of all the members, we can access some specified members of the child class. For instance, we can access the overridden methods.
2. **Down casting:** Similarly, down casting means the typecasting of a **parent object to a child object**. Down casting cannot be implicit.

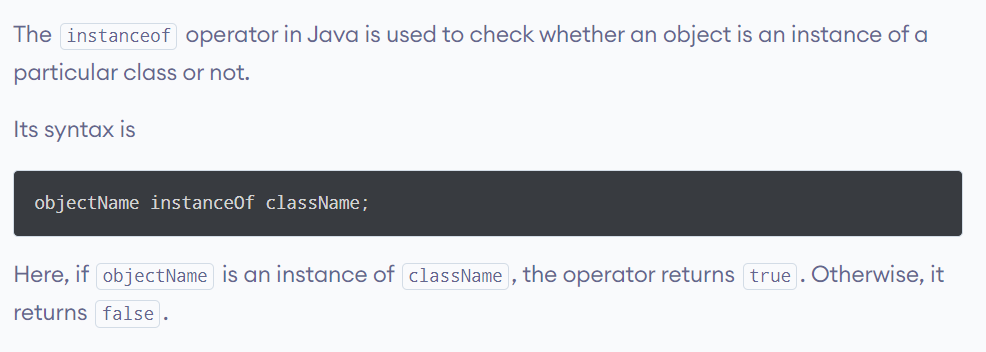


Diagram

Description automatically generated



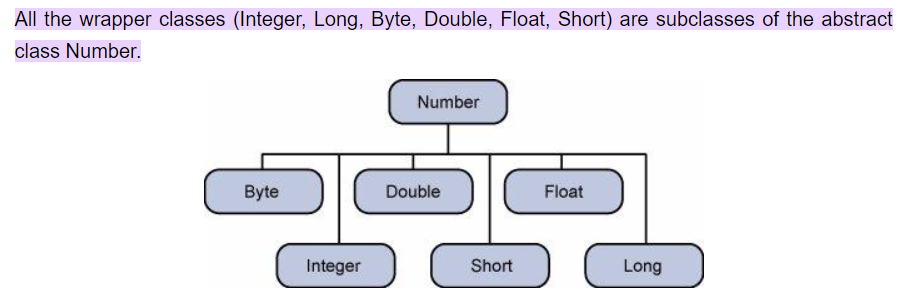
* instanceof Operator



* Wrapper classes for primitive data types

Table

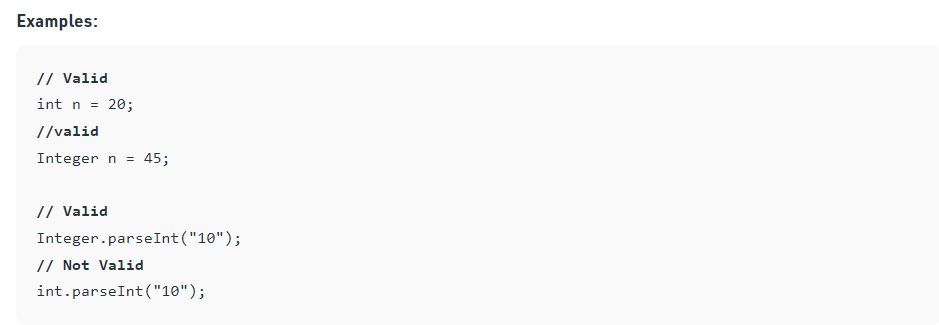
Description automatically generated with low confidence



Difference between an Integer and int in Java with Examples:

int is a primitive data type while Integer is a Wrapper class.

* int, being a primitive data type has got less flexibility. We can only store the binary value of an integer in it.
* Since Integer is a [wrapper class](https://www.geeksforgeeks.org/wrapper-classes-java/) for int data type, it gives us more flexibility in storing, converting, and manipulating an int data.
* Integer is a class and thus it can call various in-built methods defined in the class. Variables of type Integer store references to Integer objects, just as with any other reference (object) type.



Graphical user interface, text

Description automatically generated

Text, application

Description automatically generated

* The Covariant Return Type

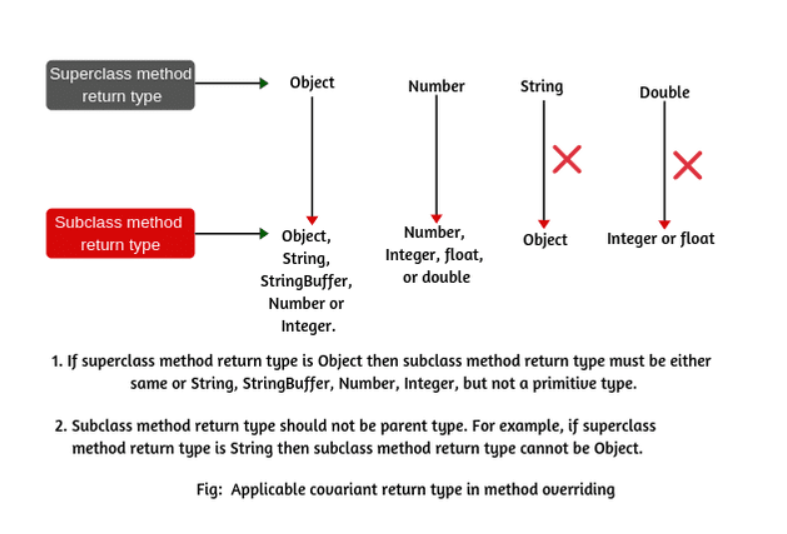
There are mainly three rules for covariant return types that should be kept in mind.

They are as follows:

1. The return type of overriding method in the subclass should be either the same as the return type of [superclass or subclass](https://www.scientecheasy.com/2020/07/java-superclass-subclass.html/).

2. The return type of overriding method in the subclass should not be a parent of the parent method return type.

3. The covariant return type is applicable only for object types not for primitive types.



* Final keyword

Diagram

Description automatically generated

* Final keyword

Once we declare a variable with the final keyword, we can’t change its value again. If we attempt to change the value of the final variable, then we will get a compilation error.

Generally, we can consider a final variable as a constant, as the final variable acts like a constant whose values cannot be changed.

* Final Method

We can declare a method as final, once you declare a method final it cannot be overridden. So, you cannot modify a final method from a sub class.

* Final Class

A class declared with the [final keyword](https://javagoal.com/final-keyword-in-java/) is known as the final class. A final class can’t be inherited by subclasses. By use of the final class, we can restrict the [inheritance](https://javagoal.com/java-inheritance/) of the class. We can create a class as a final class only if it is complete in nature, which means it must not be an [abstract class](https://javagoal.com/abstract-class-in-java). In java, all the [wrapper classes](https://javagoal.com/wrapper-class-in-java/) are final classes like String, Integer, etc.

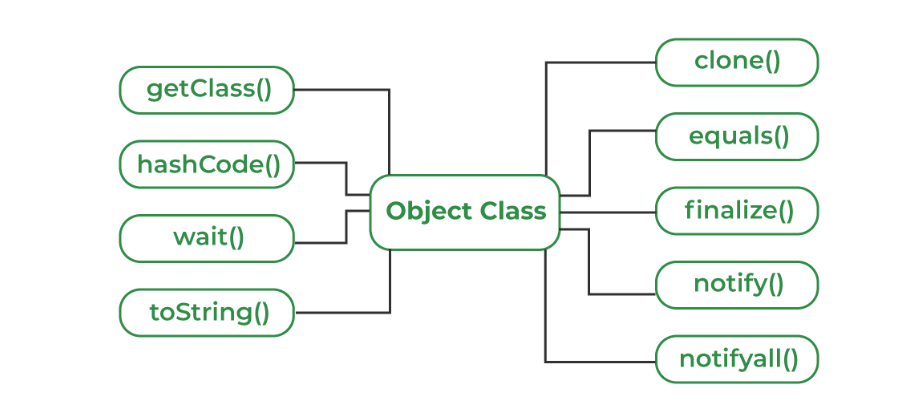
If we try to inherit a final class, then the compiler throws an error at compilation time.

* Object Class

The Object class is the parent class of all the classes in java by default. In other words, it is the topmost class of java.

Object class is present in java.lang package.

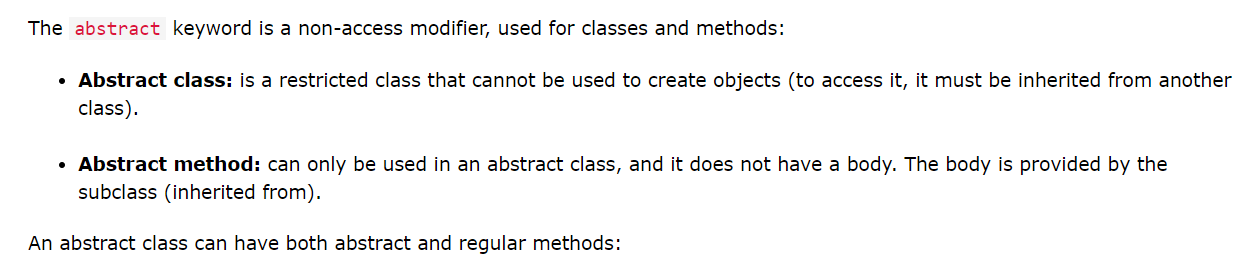
Every class in Java is directly or indirectly derived from the Object class. If a class does not extend any other class, then it is a direct child class of Object and if extends another class then it is indirectly derived. Therefore, the Object class methods are available to all Java classes. Hence Object class acts as a root of the inheritance hierarchy in any Java Program.

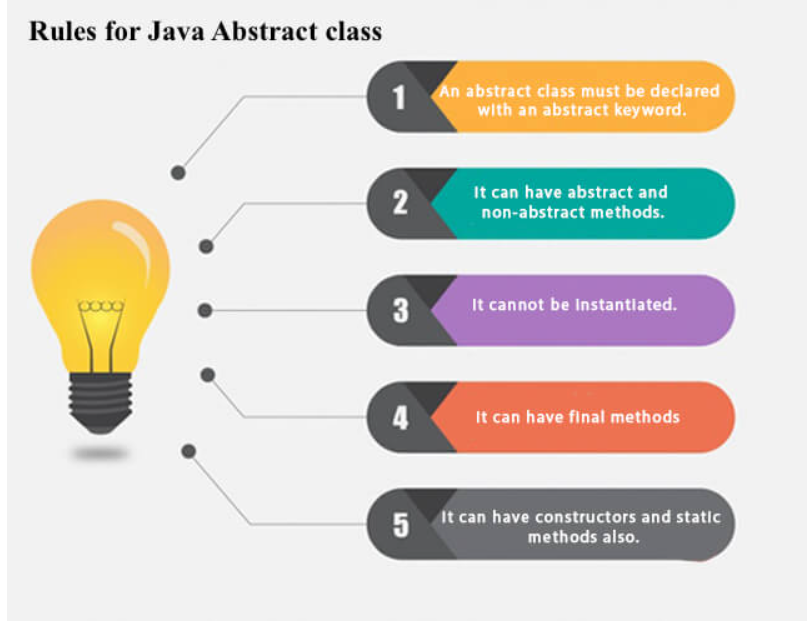


* Concreate Class

A class in java that can be created using ‘new’ keyword is called a concrete class in java. It is also known as the complete blueprint of its own self and can be instantiated. This class has the implementation of all the methods in it. Hence, this class can never contain any unimplemented methods. A concrete class can always extend an abstract class.

* Abstract Class





* Interface

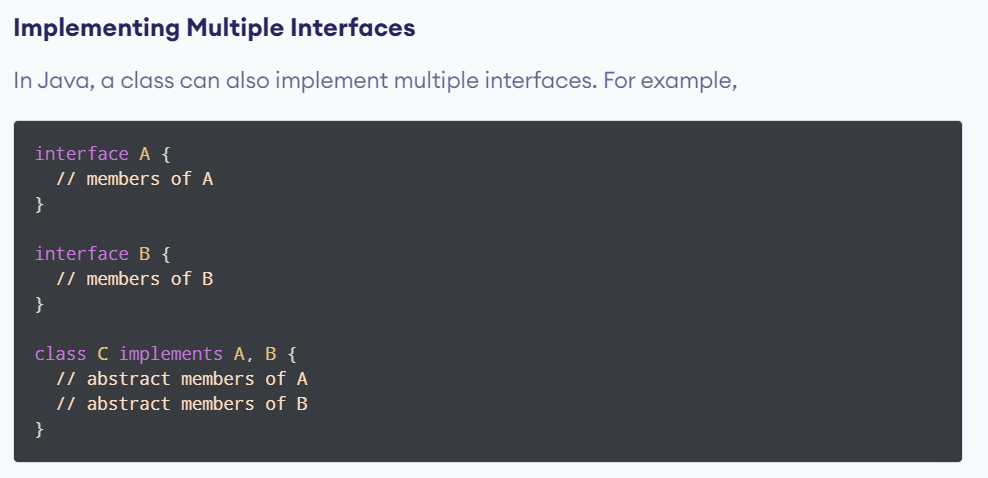


Graphical user interface, text, application

Description automatically generated

Text

Description automatically generated



Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated



Graphical user interface, text, application

Description automatically generated

Text

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

* Marker Interface

A marker interface is an [interface](https://www.baeldung.com/java-interfaces) that doesn't have any methods or constants inside it. It provides run-time type information about objects, so the compiler and JVM have additional information about the object.

A marker interface is also called a tagging interface.