**Garbage collection in java can not be enforced. But still sometimes, we call the System.gc( ) method explicitly. System.gc() method provides just a "hint" to the JVM that garbage collection should run. It is not guaranteed! that it will work or not.**

**system.gc() VS runtime.gc()**

**Both are same. There is no difference between System.gc() and Runtime.getRuntime().gc(); System.gc()internally calls Runtime.gc(). The only difference is System.gc() is a class method whereas Runtime.gc() is an instance method. The call System.gc() is effectively equivalent to the call : Runtime.getRuntime().gc().**

**What are the methods in object?**

**clone, getClass,**equals, wait, notify, notifyAll, finalize, hashCode, toString.

**What happens once an object is made in Java**

1.Memory allocates from heap to hold all instance [variables](https://data-flair.training/blogs/variables-in-java/).

2.The instance variables of the objects initializes to their default values.

3.The first thing a constructor will is call

Blank final variable

1. A blank instance level final variable cannot be left uninitialized.
2. The blank Instance level final variable must be initialized **in each constructor**.
3. The blank Instance level final variable cannot be initialized in class methods.
4. A blank static final variable cannot be left uninitialized.
5. The **static final variable** must be initialized in a **static block**.
6. A static final variable cannot be initialized **in constructor or class methods.**

#### Example

public class Tester {

**public final int a;**

**public static final int b;**

   static {

      b = 2;

   }

   Tester() {

      this(1);

   }

   Tester( int a) {

      this.a = a;

   }

   public static void main(String[] args) {

      Tester tester = new Tester();

      System.out.println("a = " + tester.a + ", b = " + b);

   }

}

#### Output

a = 1, b = 2

### **Can you declare the main method as final?**

Yes, We can declare the main method as public static final void main(String[] args){}.



***Why Generic****:This avoids ClassCastException at Runtime because you will get the error at compilation.*

***Overriding*** *only happens with method.child can access data member of parent class.*

*z = pr%q + w/x* – *y*

z= p \* r % q + w / x - y;

The \*, % and / operations are evaluated first in *left-to-right* order (i.e., they associate

from left to right), because they have higher precedence than + and -. The + and -

operations are evaluated next.

1.pr(\*)

2.pr%q(%)

3.w/x(/)

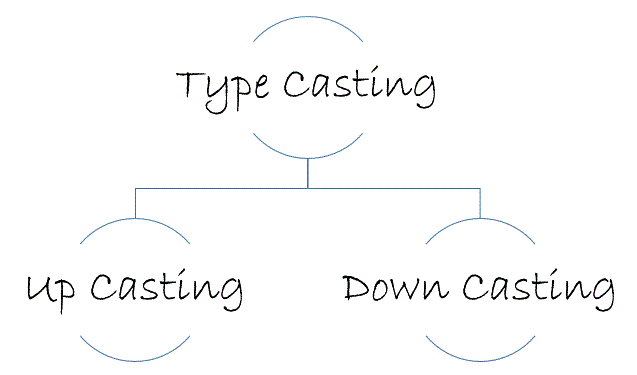
4.pr%q+w/x(+)

5. pr%q+w/x-y(-)

6.=

A great strength of Java is its rich set of ***predefined classes*** that you can *reuse* rather than “reinventing the wheel.” These classes are **grouped** into **packages.**

There are two types of type casting.:

  
Casting an object of child class to a parent class is called **upcasting**.

Casting an object of a parent class to its child class is called **downcasting**.

Example: *Vehicle* is Parent Class and *Two-wheeler* is Child Class.

Vehicle v1 = **new** Vehicle();

Vehicle v2 = **new** TwoWheeler();//Upcasting

TwoWheeler v3 = (TwoWheeler) **new** Vehicle();//DownCasting

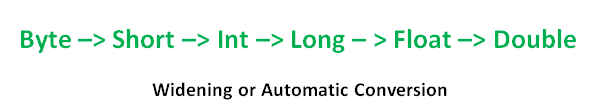
TwoWheeler v4 = **new** TwoWheeler();

# **Type conversion in Java with Examples**

**Widening or Automatic Type Conversion**

Widening conversion takes place when two data types are automatically converted. This happens when:

* When we assign value of a smaller data type to a bigger data type.



Example:

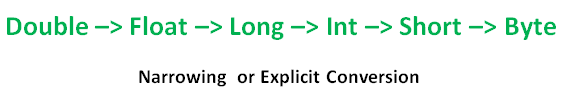
int i = 100;

   // automatic type conversion

      long l = i;

**Narrowing or Explicit Conversion**

If we want to assign a value of larger data type to a smaller data type we perform explicit type casting or narrowing.



Example:

**char** c='a';

**char** c2='1';

**int** a=c;

**int** b=c2;

System.out.println(a); //97

System.out.println(b);  //49

We can convert **int to char in java** using typecasting. Bz char have low than int.

**int** a=65;

**char** c=(**char**)a;  //we need to type cast otherwiae CT error.

System.out.println(a);

1. **Can a main() method be declared final?**

Yes. Any inheriting class will not be able to have it's own default main() method.

1. **Does the order of public and static declaration matter in main() method?**

No. It doesn't matter but void should always come before main().

1. **Can a class be defined inside an Interface?**

Yes it's possible.

1. **Can an Interface be defined inside a class?**

Yes it's possible.

1. **Why does Java not support operator overloading?**

Operator overloading makes the code very difficult to read and maintain. To maintain code simplicity, Java doesn't support operator overloading.

The non-Unicode letter characters $ and \_ may appear as the first character of an identifier

1. **If a class is declared without any access modifiers, where may the class be accessed?**

A class that is declared without any access modifiers is said to have package access. This means that the class can only be accessed by other classes and interfaces that are defined within the same package.

1. A public class may be accessed outside of its package. A non-public class may not be accessed outside of its package.

 JVM’s are platform specific

**JIT =** quick compiler.

**JVM=interpreter**

java.lang.Object **base class Of All Classes.**

In Java,[**Arrays**](https://data-flair.training/blogs/java-array/) are objects.

**Function of assert keyword used in Jdk1.4.x?**

In order to validate certain expressions, we use [assert](https://data-flair.training/blogs/java-assert/)keyword for the important arguments. Meaning, without that the method does nothing.

**constant Variable in Java=**  should declare as static and final.

**How to Create Object without using the keyword “new” in java?**

Without new, the Factory methods are used to create objects for a class. For example  
**Calender c=Calender.getInstance();**  
Here Calender is a class, and the method getInstance() is a Factory method which can create an object for Calendar class.

**Servlet is pure java object or not?**

Yes, pure java object.

Difference between execute().executeQuery() and executeUpdate().

**String use ‘concate ‘method,String Buffer and String Builder used ‘append’ method.**

**1. We can pass *object as parameter*:-**

**No need for copy constructor in Java :Bz all objects are passed by reference. Moreover, Java does not even support automatic pass-by-value**

It is good practice to initialize instance variables of an object before **passing object** **as parameter** to method otherwise it will take default initial values.

**Example:-**

public class Rectangle{

int length;

int width;

Rectangle(int l, int b)

{

length = l;

width = b;

}

void area(Rectangle r1)

{

int a = r1.length \* r1.width;

System.out.println("Area of Rectangle : " + a);

}

public static void main(String args[]){

Rectangle r1 = new Rectangle(10, 20); // ***best way to initialize instant variable before pass***

***‘object’ as parameter***

r1.area(r1); // ***we pass r1.lentgh and r1.width indirectly***

}}

Output:- Area of Rectangle : 200

**2.Method can return object as class type**

In Java Programming a method can return **any type of data**, including **class types** that you create.  
For example, in the following program, the **getRectangleObject( )** method returns an object.

class Rectangle

{ int a;

Rectangle(int l) // ***initialize instant variable before pass object***

{

a = l;

}

Rectangle getRectangleObject()

{

Rectangle rect = new Rectangle(10);

return rect;

}

}

public class RetOb {

public static void main(String args[]) {

Rectangle ob1 = new Rectangle(5);

Rectangle ob2 = ob1.getRectangleObject();// ***method that return object and store in obj2***

System.out.println("ob1.a : " + ob1.a);

System.out.println("ob2.a: " + ob2.a); }}

**3.We pass Object as parameter and return Object:**

**public class Practice**

**{**

**int a;**

**Practice( int b)**

**{**

**a=b;**

**}**

**void passobj(Practice p1)**

**{**

**System.out.println(p1.a);**

**}**

**Practice change()**

**{**

**Practice p3 = new Practice(30);**

**return(p3);**

**}**

**public static void main(String[] args)**

**{**

**Practice p1 = new Practice(20);**

**p1.passobj(p1);**

**Practice p2 = p1.change();**

**System.out.println(p1.a);**

**System.out.println(p2.a);**

**}**

**}**

**Output: 20 20 30**

**Note: when Wrapper class object pass, it will work as pass by value**