What is casting?

Click to continue

Casting in real world is a process by which a liquid is poured into a mold and is allowed to solidify. The liquid attains the shape of the mold.

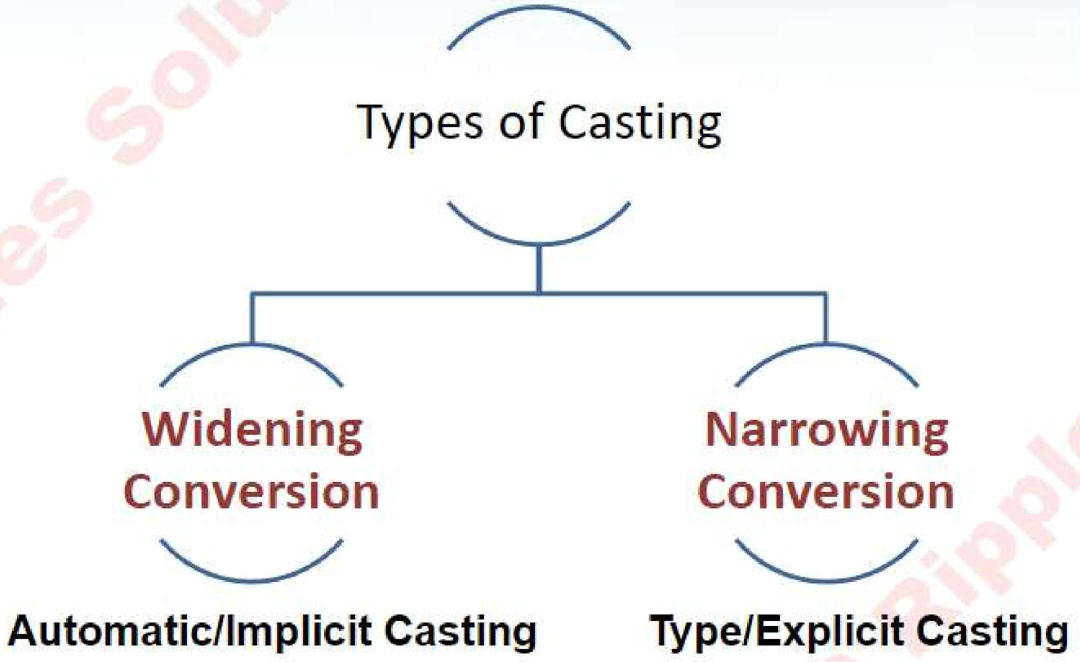
In java world **primitive type casting** is a technique where one primitive value is assigned to a variable of another primitive type.

Example: A variable **int** is casted into **double.** Here double is the mold and int variable is casted as a double.

NOTE:
- Casting can be done with any primitive types.
- Boolean type variable cannot be casted

Primitive casting

Click to continue



Automatic (or) Implicit Casting

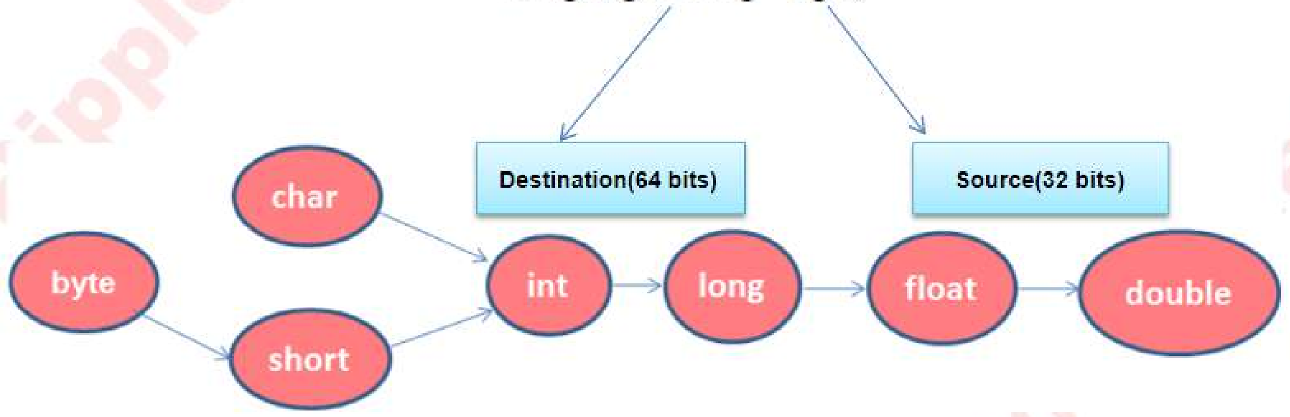
Click to continue

Implicit casting is done when,

1. Two data types to be converted are compatible.
2. The destination data type is larger than the source data type.

Illustration: An int value (source) to a long variable (destination).

```
int age=22;
long ageLong=age;
```



The above diagram represents the direction in which automatic casting is applicable.

Type (or) Explicit Casting

Click to continue

Explicit (or) Casting will be done when the destination data type is smaller than the source data type.

Illustration:

```
double tax=7.25;
float taxFloat=tax;           // compile-time error.
float taxFloat=(float) tax; // Explicit Casting.
```

When a floating-point value is assigned to an integer type the value would be truncated.

Example: if the value 2.35 is assigned to an integer, the resulting value will be 2.The 0.35 will be truncated.

Java Operators

Click to continue

Operators are special symbols that perform specific operation on one, or more operands and return a result.

Illustration:

$$x + y = z$$

Here,

- x and y are variables
- the data in x and y are operands
- '+' is the operator
- z contains the result

Java Operators

Click to continue

- Operators operate on the data stored by a ***variable***.
- The data that is being operated on is called an **operand**.
- Operators can produce a new value without changing the values of the operands.
- Operators can compare the values of two operands.

Operators Types in Java

Click to continue

- **Unary operators**: Operates on ***one*** operand.
  Illustration:
  ++ increments the value of its operand by 1.
  - - decrement the value of its operand by 1.

- **Binary operators**: Operates on ***two*** operands.
  Illustration:
  + adds the values of its two operands.

- **Ternary operators**: Operates on three operands (?).

The ternary operator evaluates a boolean expression and assigns the first value if condition is true else assigns the second value.

Illustration:
variable Y = (boolean expression) ? value if true : value if false

Ternary Operator Illustration

Click to continue

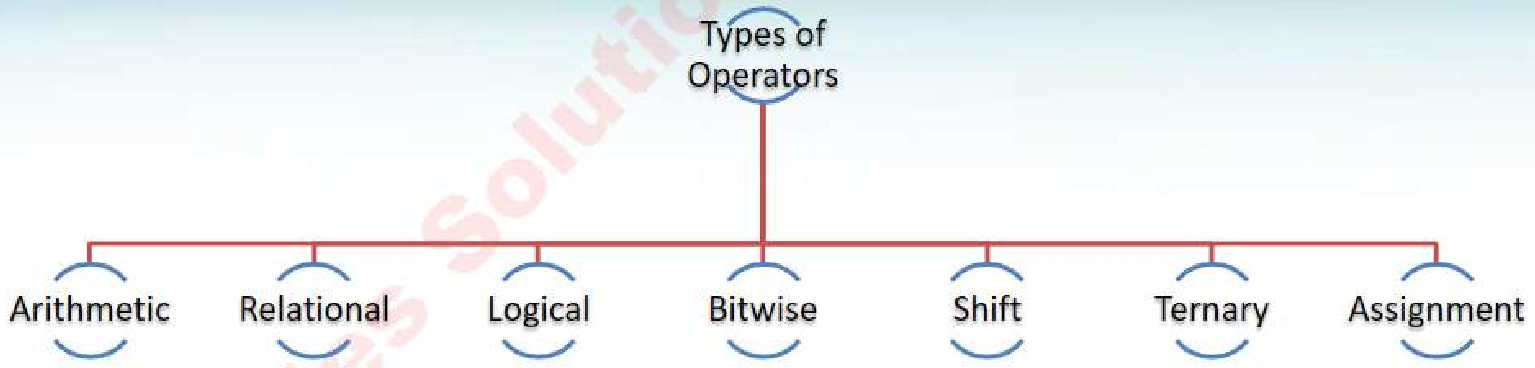
```
public static void main(String args[]) {
    int operand1 = 10;
    int operand2;
    operand2 = (operand1 > 5) ? 20 : 30;
    System.out.println("Value of operand2 is : " +
    operand2);
}
```

In the above example since the **operand1** is greater than **5** the value **20** will be assigned to **operand2**.

If **operand1** is **< 5** the value **30** will be assigned to **operan2.**

Types of operators

Click to continue



Arithmetic Operators

Operator	Value of x	Value of y	Usage	Value of z
+	3	3	Z=x+y	6
-	4	3	Z=x-y	1
*	5	5	Z=x*y	25
/	15	5	Z=x/y	3
%	4	2	Z=x%y	0

Operators

Click to continue

Unary Operators

Operator	Use	Description
-	int x=-y;	Negates the operand by and stores the value in x
++	int x=y ++	Increments the value of y by 1
--	Int x=y --	Decrements the value of x by 1

Unary Operators Example

Initial Value of a	Code Statement	Final Value of b	Final value of a
6	b=++a	7	7
6	b=a++;	6	7
6	b=--a;	5	5
6	b=a--;	6	5

Relational Operators

Operator	Use	Description
>	x>y	True if x is greater than y, otherwise false
>=	x>=y	True if x is greater than or equal to y, else false
<	x < y	True if x is less than y, otherwise false
<=	x <=y	True if x is less than or equal to y, otherwise false
==	x == y	True if x and y are equal, otherwise false
!=	x= y	True if x and y are not equal, otherwise false

Shift Operators

Click to continue

These operators operates on one or more bit patterns or binary numerals.

Operator	Use	Description
<<	x<<y	The left shift operator, <<, shifts all of the bits in a value to the left a specified number of times.
>>	x>>y	The right shift operator, >>, shifts all of the bits in a value to the right a specified number of times.
>>>	x >>> y	Shift right zero fill operator. The left operands value is moved right by the number of bits specified by the right operand and shifted values are filled up with zeros.

Assignment Operators

Operator	Use	Description
+=	x+=y	Equivalent to x=x+y
-=	x-=y	Equivalent to x=x-y
*=	x*=y	Equivalent to x=x * y
/=	x /=y	Equivalent to x=x/y
%=	x%=y	Equivalent to x=x%y

Operator Precedence

Click to continue

If in a expression there are more than one operators the order in which they are evaluated would be based on the operator precedence.

Precedence	Operator
1	( ),[ ]
2	++,--
3	*,/,%,+,-
4	<.<=,>,>=
5	==,!=
6	&&,||

Example :1

y = (*ab) + c / d;**

Here is the order on how the above expression is processed

1. **a * b**
2. **c / d**
3. **Final Result = result of step 1 + result of Step 2**