

# TYPE CASTING

## Definition:-

For change the original data type into another data type.

It is divided into type:-

### 1. Implicit

- ✓ Conversion performance internally through technology
- ✓ Lower to higher
- ✓ Known as Widening

#### Example:-

```
byte b=10;  
int i=b;// Implicitly type casting
```

### 2. Explicit

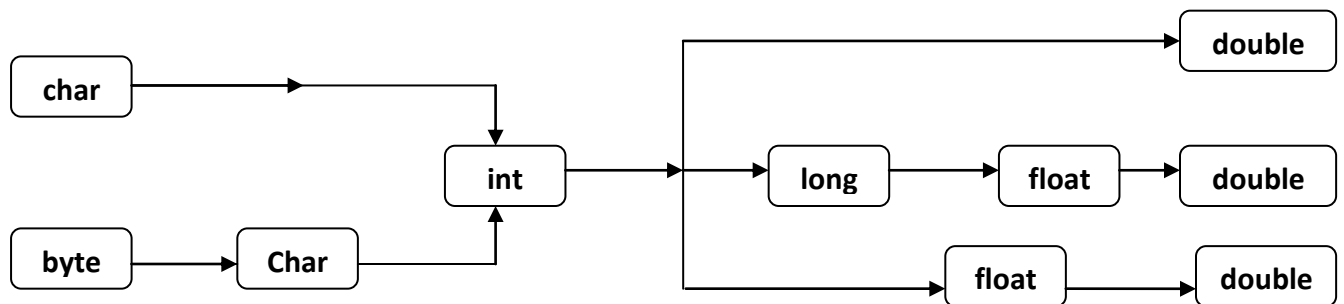
- ✓ Something done by program for conversion
- ✓ Higher to lower
- ✓ Known as narrowing

#### Example:-

```
int i=20;  
byte b=(byte)i;
```

## Type Promotion

- ✓ It is always use in implicit type casting.
- ✓ It is always promote into higher data-type.



- ✓ If matching situation is not found then error will display.
- ✓ In following fig. long promote into float. That means chance to loss the data.

# OPERATORS

## **Definition:-**

Operator is symbols which perform some task for desired result.

There are three types of operators:-

1. Unary
2. Binary
3. Ternary

## **Precedence:-**

- ✓ Brackets
- ✓ Postfix (++,-,!,)
- ✓ Prefix (++,-)
- ✓ New
- ✓ Multiplication(\*,%,/)
- ✓ Addition(+,-)
- ✓ Shift(>>,<<)
- ✓ Relational (>,<,>=,<=)
- ✓ Equality (==)
- ✓ Bitwise And (&)
- ✓ Bitwise X-or(^)
- ✓ Bitwise Or (|)
- ✓ Logical And (&&)
- ✓ Logical Or (||)
- ✓ Conditional (?:)
- ✓ Assignment(+)

## **Association:-**

1. Left to Right
2. Right to Left

## **Rules of Association:-**

Subject : Programming in Java

Paper Code : CC-3

Course : BCA, Sem-II

©Chanchal Acharya, e-mail : c.acharya.me@gmail.com

- ✓ **Unary operator:-** Except postfix increment, prefix decrement all the unary operator associate value right to left.
- ✓ **Binary operator:-** Except assignment and relational operator all binary operator associate value left to right.
- ✓ **Ternary operator:-** It always associate value right to left.

# OOPS ( *Object Oriented Programming Structure.* )

## Class

- ✓ Class is a representative of similar type of objects. (29-03-2011)
- ✓ It is a collection of data member and member function.

Data member and member function or methods have two types:-

1. **Instance:-** Instance data member or member function allocate memory separately for all reference variable or object.
2. **Static:-** Static data member and member function allocate memory common for all reference variable or object.

## Example 1:

```
class Emp
```

```
{
```

```
String name;
```

```
int salary;
```

```
void get(int s, String n)
```

```
{
```

```
    name=n;
```

```
    salary=s;
```

```
}
```

```
void show()
```

```
{
```

```
    System.out.println("Name =" +name);
```

```
    System.out.println("Salary =" +salary+" thousand");
```

```
}
```

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

```
{
```

```
    Emp obj = new Emp();
```

```
    obj.get(25, "Vishal kumar");
```

```
    obj.show();
```

```
}
```

```
}
```

Instance data member

Instance member function

Return reference Id

Reference variable

### Output

Name=Vishal Kumar

Salary=25 thousand

Subject : Programming in Java

Paper Code : CC-3

Course : BCA, Sem-II

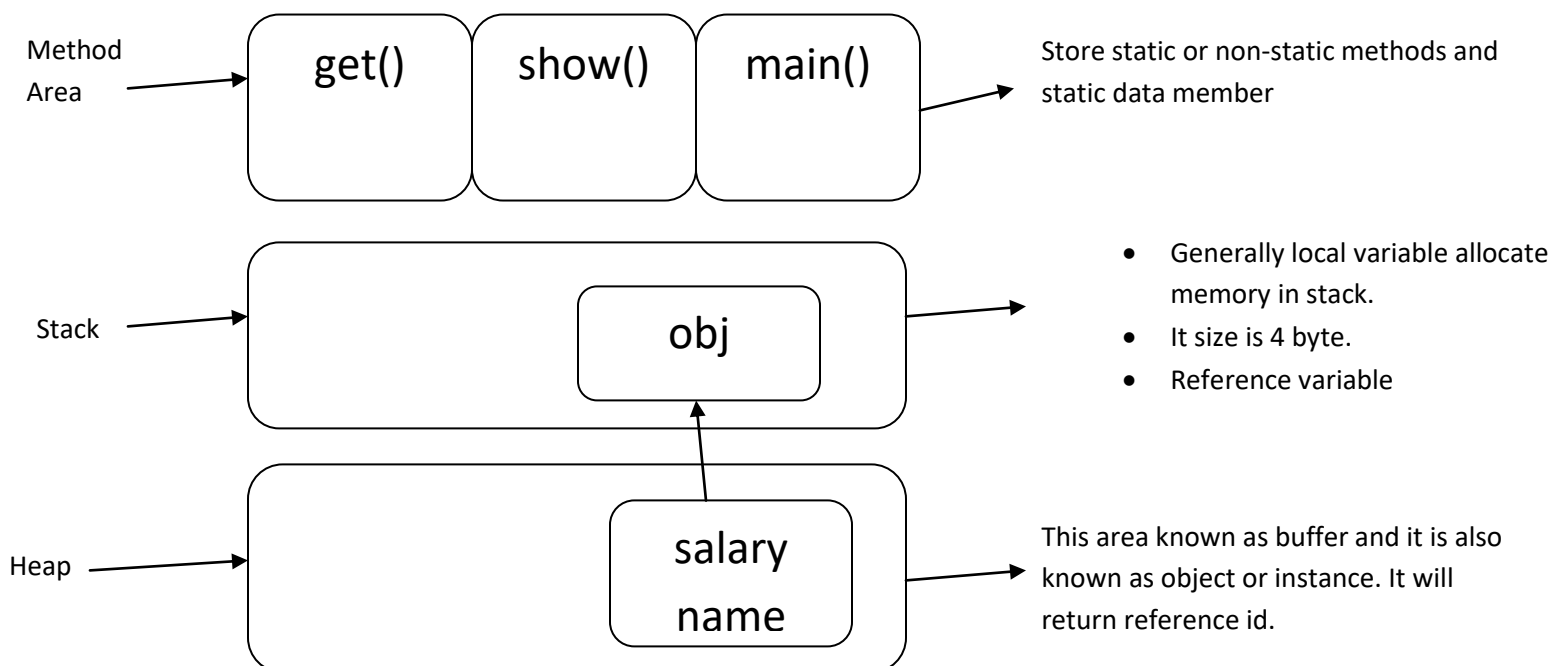
©Chanchal Acharya,

e-mail : c.acharya.me@gmail.com

### Notes:-

- ✓ new operator use to allocate memory for object at runtime in heap area.
- ✓ Reference variable carry four byte space for store reference id.
- ✓ Javac keyword is use to compile the source code and covert into byte-code.
- ✓ Java keyword is use to load byte-code into JVM(Java Virtual Machine).
- ✓ Java allocate memory for member and method in two step:-
  - **Static memory allocation:-** At the time of byte-code load in JVM, which member or method can occupy space in memory is known as static memory allocation. Reason is before execution we all ready know how many space will occupied.
  - **Dynamic memory allocation:-** Memory allocate at the time of execution/runtime is known as dynamic memory allocation.
- ✓ To find the size of reference variable java provides method which is known as “public static void premain()”. This concept available after jdk1.6 version, before it no concept will provided.
- ✓ In java no any concept of pointer explicitly but internally java use pointer to pointer concept to provide reference id in encrypted form into reference variable in stack area.
- ✓ We never display the reference id reason is, it is encrypted form. But if we print it then its output something like this:-

**Class\_name@3e25a5**



Subject : Programming in Java

Paper Code : CC-3

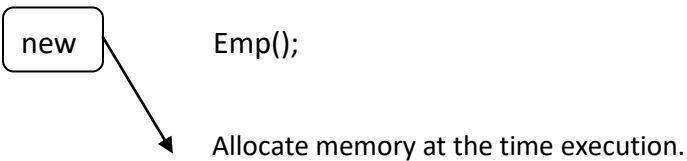
Course : BCA, Sem-II

©Chanchal Acharya,

e-mail : c.acharya.me@gmail.com

This object has an id which can known as reference Id. And which variable holds the reference id is known as reference variable.

```
Emp obj = new Emp();
```



Allocate memory at the time execution.

- ✓ .(dot) is known as associative operator. **(30-03-2011)**
- ✓ new Emp(); is known as Anonymous object and it will return reference id.
- ✓ This reference id store in a container known as reference variable.
- ✓ Reference variable store in stack and take 4 byte space in memory.

### Important points

- ✓ All object share the same method location.
- ✓ Always keep those properties of an object as instance data member whose values are changing for each object.
- ✓ Static member allocate memory at class loading time. It will take space inside class initialize. It is the part of class area.
- ✓ All object sharing the same memory location of static member.
- ✓ All static things are part of class & non-static things are a part of object.
- ✓ Static member function can use static and non-static data member or member function. Static data member or member function call directly by its name or help of class name or by the help object but non-static data-member or member function always access by the help of reference variable..
- ✓ Non-static member function never call directly inside static method.
- ✓ The reference id of an object can be put into the any no. of reference variable.

### Example:-

```
class Demo
{
    //int x=10;//Static initialize of non-static data member.
```

Subject : Programming in Java

Paper Code : CC-3

Course : BCA, Sem-II

©Chanchal Acharya,

e-mail : c.acharya.me@gmail.com

```
int x;
static int y=100;
Demo(int n)
{
    x=n;
}
Demo()
{
    x=10;
}
void show()
{
    System.out.println("X="+x);
    System.out.println("Y="+y);
}
static void disp()
{
    System.out.println("Y="+y);
}
public static void main(String[]args)
{
    new Demo().show();
    new Demo().disp();
    disp();
    Demo.disp();
    System.out.println(new Demo().x);
    System.out.println(new Demo(60).x);
}
}
```

# POLYMORPHISM (01-04-2011)

- ✓ It will reduce the complexity. It always depends on object behavior.
- ✓ When an object which have different behavior and then this object is known as polymorphism.

**Normally it has two types:-**

1. **Compile Time:-** Compile time occurs when ever an object is bound its functionality at compile.

**Example:-** Function Overloading, Operator Overloading(But java does not support operator overloading explicitly)

2. **Run Time:-** Run time polymorphism occurs when ever object bound its functionality at run time.

**Example:-** function Overriding

Polymorphism is achieved by behaviors (functionality).

It has two type:-

1. Early binding: - Before the event occur.
2. Late binding (runtime binding):- At the run time.

## Important Points:-

- ✓ Java does not support explicitly operator overloading.
- ✓ Java support implicitly operator overloading. **Example:-** "+"
- ✓ By default every method or member function in java are virtual. We know that virtual function always bind at run time.
- ✓ So, java supports only run time polymorphism.
- ✓ It never support compile time polymorphism.
- ✓ When some specific components able to adjust into different object then it is known as **Polymorphism** but when completely any object merge into different object than it is known as "**Generalization**".

**Some book says that private, final and static methods are bind at compile time. But actually java doesn't support compile time polymorphism.**

## Example of polymorphism

- ✓ Student
- ✓ Doctor
- ✓ Patient
- ✓ Leader. Etc.....



Subject : Programming in Java

Paper Code : CC-3

Course : BCA, Sem-II

©Chanchal Acharya,

e-mail : c.acharya.me@gmail.com

## Function Overloading

It says that within a class more than one function/method which name is similar but different in prototype.

### **Prototype**

1. Access modifier
2. Access specifier
3. Return type :- Doesn't matter in function overloading.
4. Function name :- Same (Constant).
5. Arguments :- Different either in no. or in sequence( Data type)

#### ✓ Change in arguments

- Change the no. of arguments in each function.
- Keep no. of arguments same but different in data-type.

#### ✓ Change in return type

- Return type does not participate in function overloading

#### ✓ Change in arguments and return type

They can achieve the property of function overloading.

### **Example:-**

```
class Demo
```

```
{
```

```
int x=0;
```

```
int y=0;
```

```
void add()
```

```
{
```

```
    System.out.println(x+y);
```

```
}
```

```
void show(int x)
```

```
{
```

```
    this.x=x;
```

```
    System.out.println(this.x+y);
```

```
}
```

```
void show(int x, int y)
```

```
{
```

```
    this.x=x;
```

```
    this.y=y;
```

Subject : Programming in Java

Paper Code : CC-3

Course : BCA, Sem-II

©Chanchal Acharya,

e-mail : c.acharya.me@gmail.com

```
        System.out.println(this.x+this.y);
    }

    public static void main(String[]args)
    {
        Demo d=new Demo();
        d.add();
        d.add(70);
        d.add(10,50);
    }
}
```

### Output

0  
70  
60

### this

It holds the reference id of current class or object. (05-04-2011)

**Data Shadowing:-** When a class level variable and local variable are has a same name then this is known as **data shadowing**.

### Example:-

class DataShadowing

```
{
    int x=10;
    void show(int x)
    {
        System.out.println(x);
        System.out.println(this.x);
    }
    void show(int x,DataShadowing d)
    {
        System.out.println(x);
        System.out.println(d.x);
    }
    public static void main(String[]args)
    {
        DataShadowing d=new DataShadowing();
        d.show(50);
        d.show(70,d);
    }
}
```

### Output

50  
10  
70

Subject : Programming in Java

Paper Code : CC-3

Course : BCA, Sem-II

©Chanchal Acharya,

e-mail : c.acharya.me@gmail.com

}

### Note:-

- ✓ Every non-static function contain reference id as a last argument implicitly.
- ✓ this can't use inside static method.

### Example:-

```
class Temp
```

```
{
```

```
    int x=8;
```

```
    void show(int x,Temp this)
```

```
    {
```

```
        System.out.println(x);
```

```
        System.out.println(this.x);
```

```
    }
```

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

```
    {
```

```
        Temp t=new Temp();
```

```
        t.show(40,t);
```

```
    }
```

```
}
```

Implicitly

Implicitly

### Output

40

8