SNS COLLEGE OF ENGINEERING

Kurumbapalayam (Po), Coimbatore – 641 107

An Autonomous Institution

Accredited by NAAC – UGC with 'A' Grade

Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai

23CSB101

OBJECT ORIENTED PROGRAMMING

UNIT II

INHERITANCE, PACKAGES AND INTERFACES

Overloading Methods

By

M.Kanchana

Assistant Professor/CSE

Overloading Methods

Method Overloading is a feature in Java that allows a class to have **more than one methods having same name**, but with **different signatures** (Each method must have different number of parameters or parameters having different types and orders).

Advantage:

- ✓ Method Overloading increases the readability of the program.
- ✓ Provides the flexibility to use **similar** method with different parameters.

Overloading Methods

Three ways to overload a method

1.Number of parameters. (Different number of parameters in argument list)

For example: This is a valid case of overloading

add(int, int)

add(int, int, int)

2.Data type of parameters. (Difference in data type of parameters)

For example:

add(int, int)

add(int, float)

3. Sequence of Data type of parameters.

For example:

add(int, float)

add(float, int)

Overloading Methods

Method Overloading and Type Promotion

Type Promotion: When a data type of **smaller size** is promoted to the data type of **bigger size** than this is called type promotion,

for example: **byte** data type can be promoted to **short**, a short data type can be promoted to int, long, double etc.

Data Type	Size (Bytes)	Range
`byte`	1	-128 to 127
`short`	2	-32,768 to 32,767
`int`	4	-2,147,483,648 to 2,147,483,647
`long`	8	-9,223,372,036,85 5,808 to 9,223,372,036,854,775,807

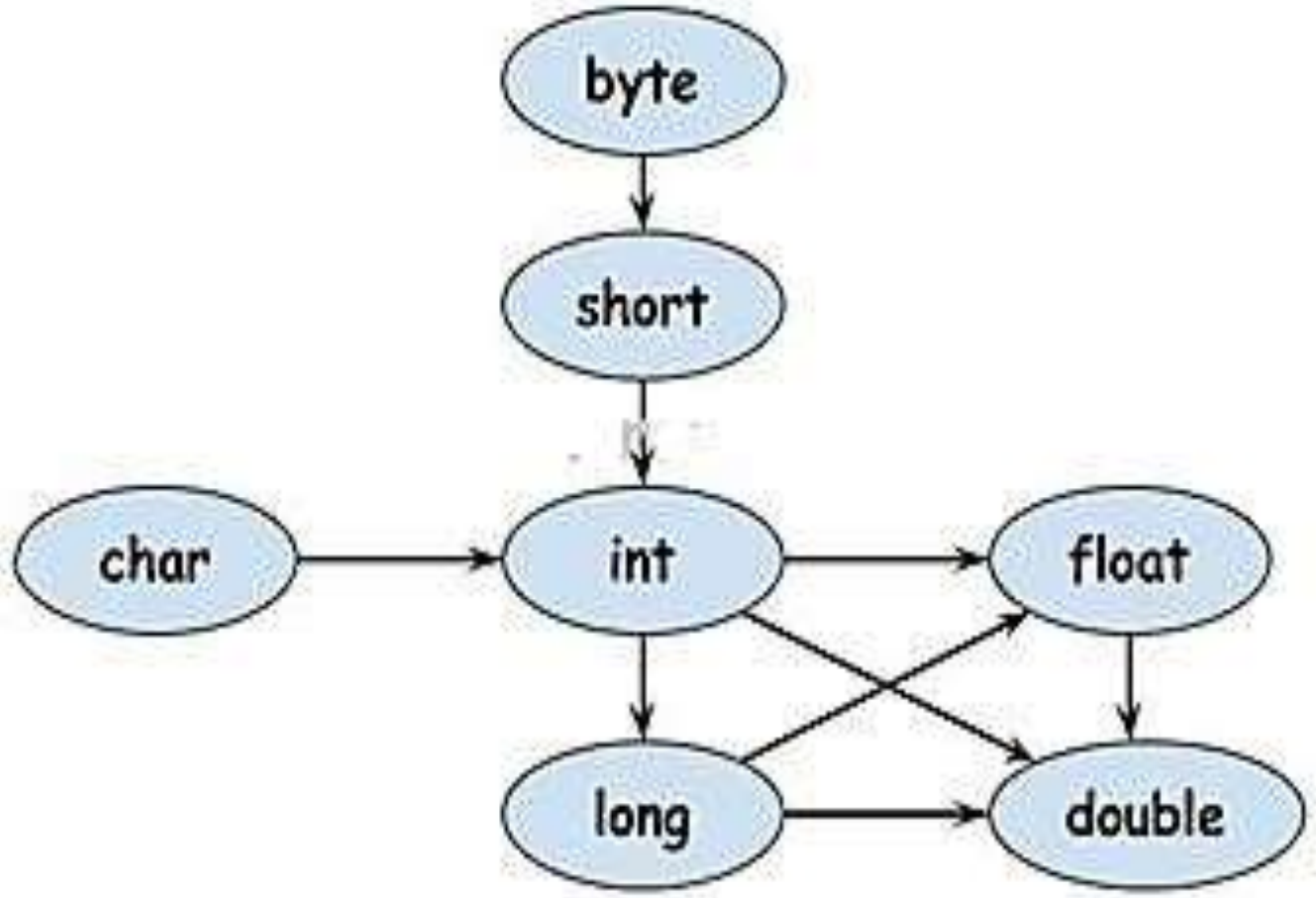
Overloading Methods

Type Promotion in Method Overloading:
One type is promoted to another implicitly if no matching datatype is found.

Type Promotion Table:
The data type on the left side can be promoted to the any of the data type present in the right side of it.

byte → short → int → long → double short
→ int → long → float → double int → long
→ float → double
float → double
long → float → double
char → int → long → float → double

Overloading Methods



Overloading Methods

```
class Overloading
{
void sum(int a, float b)
{
System.out.println(a+b);
}
void sum(int a, int b, int c)
{
System.out.println(a+b+c);
}

public static void main(String args[])
{
OverloadingCalculation1 obj=new OverloadingCalculation1();
obj.sum(20,20);
obj.sum(100,'A');
obj.sum(20,20,20);
}
}
```

Overloading Methods

```
class Overloading
{
void sum(int a, float b)
{
System.out.println(a+b);
}
void sum(int a, int b, int c)
{
System.out.println(a+b+c);
}
public static void main(String args[])
{
Overloading obj=new Overloading ();
obj.sum(20,20);
obj.sum(100,'A');
obj.sum(20,20,20);
}
}
```

Output:

40.0

165.0

60

Object as Parameters

- **Primitive Types**: Passed by value, meaning changes to the parameter do not affect the original variable.

- **Objects:** The reference to the object is passed by value, but the object itself can be modified.

Java does not support pass by reference directly. However, when you pass an object to a method, the reference to the object is passed by value.

This means that while the reference itself is a copy, it still points to the same object in memory. Therefore, changes made to the object's fields inside the method will affect the original object.

Object as Parameters

```
class Box {
    int value;
}
public class AdditionExample {
    public int addPrimitives(int x, int y) {
        return x + y;
    }

public void addToBox(Box b, int add) {
        b.value = b.value + add;
    }
```

```
public static void main(String[] args) {
        AdditionExample example = new AdditionExample();

        // Addition using primitives
        int a = 10;
        int b = 20;
        int primitiveSum = example.addPrimitives(a, b);

        System.out.println("Sum using primitives: " +
primitiveSum);

        // Addition using an object
        Box box = new Box();
        box.value = 10;

        example.addToBox(box, 20);

System.out.println("Box value after addition: " + box.value);
    }
}
```

Object as Parameters

OUTPUT:

Sum using primitives: 30
Box value after addition: 30

THANK YOU