# Group 5 Assignment

**Unexpected Behavior with Overloaded Methods**

## 1) Method Overloading :

When we define two or more methods with same name in Java, it is called method overloading. Java utilizes method signatures to differentiate among methods with identical names.

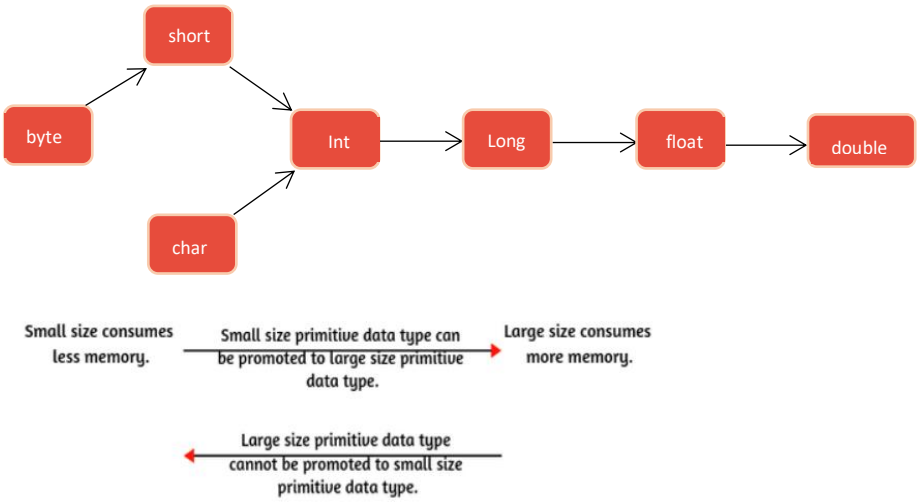
These method signatures can be differentiated by three ways:

1. Number of input parameters
2. Type of input parameters
3. Both

## 2) How Java ensures unambiguity while type conversion during method overloading :

**2.1 )** As specified in **Java Language Specification** 15.12.2.5 , " If more than one member method is both accessible and applicable to a method invocation, it is necessary to choose one to provide the descriptor for the run-time method dispatch".

**2.2)** While resolving overloaded methods if the exact match method is not available, you will not immediately get any compile-time error.
First, the compiler promotes the lower data type argument to the higher data type argument and then checks whether the match method is available or not. If the match method is available, it will be considered otherwise the procedure of data type promotion will reiterate. This process is called **Automatic Type Promotion**.




**2.3)** The following rules define the direct super-type relation among the primitive types in respective cases:

- double > float
- float > long
- long > int
- int > char
- int > short
- short > byte

## Example 1: Auto type promotion of int to float

### Source code

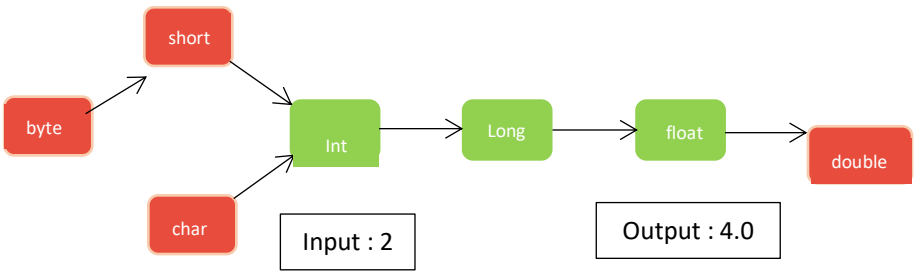
```java
/*Author : Group 5*/
 date: 11/09/21
public class AutoTypePromotion {

    // Argument data type req = double
    public void m(double a) {
        System.out.println(a * a);
    }
    // Argument data type req = float
    public void m(float a) {
        System.out.println(a * a);
    }

    public static void main(String[] args) {
        AutoTypePromotion atp = new AutoTypePromotion();
        atp.m( a: 2);
    //Argument is of type Int
    }
}
```

double required

float required

Int passed

### Auto type promotion flow




### Output

```
4.0
```

## Example 2: Auto type promotion of int to double

### Source code

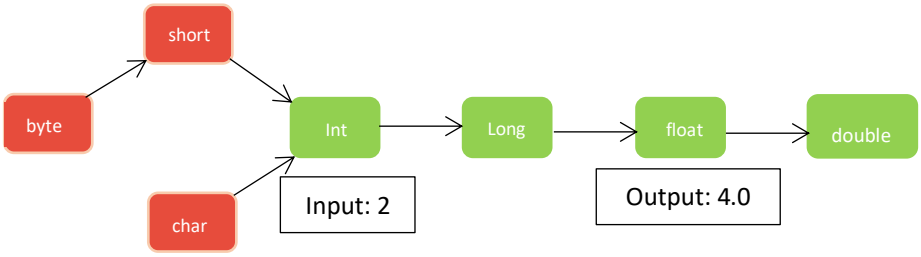
```java
/*Author : Group 5*/
 date: 11/09/21
public class AutoTypePromotion {

    // Argument data type req = double
    public void m(double a) {
        System.out.println(a + a);
    }
    // Argument data type req = short
    public void m(short a) {
        System.out.println(a + a);
    }

    public static void main(String[] args) {
        AutoTypePromotion atp = new AutoTypePromotion();
        atp.m( a: 2);
    //Argument is of type Int
    }
}
```

double required

short required

Int passed

### Auto type promotion flow




### Output

4.0

## Example 3: Auto type promotion of char to int

**Source code**

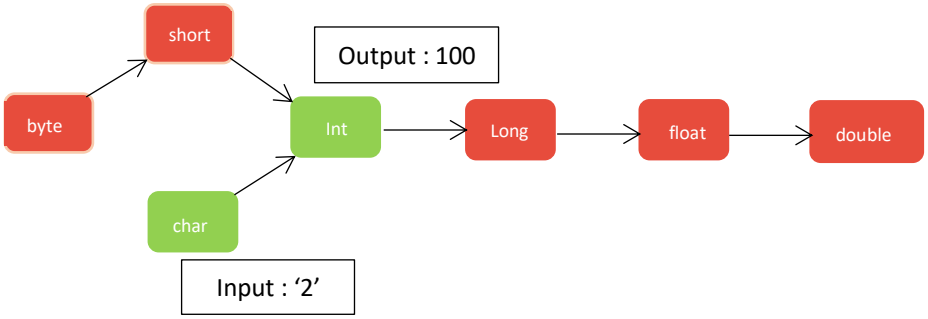
```
/*Author : Group 5*/
 date: 11/09/21
public class AutoTypePromotion {

    // Argument data type req = int
    public void m(int a) {
        System.out.println(a + a);
    }
    // Argument data type req = float
    public void m(float a) {
        System.out.println(a + a);
    }

    public static void main(String[] args) {
        AutoTypePromotion atp = new AutoTypePromotion();
        atp.m( a: '2');
    //Argument is of type char
    }
}
```

Int required

float required

char passed

**Auto type promotion flow**




**Output**

```
100
```

## Example 4: Auto type promotion of long to float

### Source code

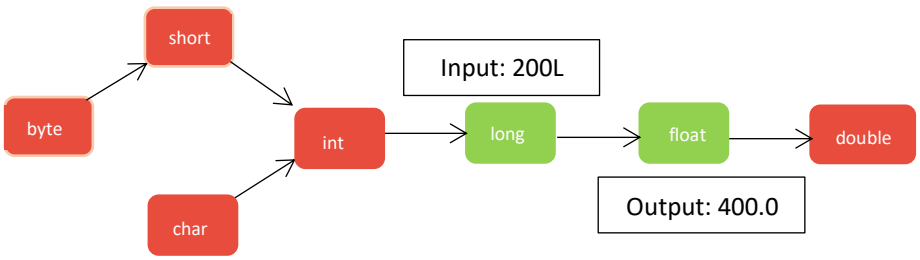
```java
/*Author : Group 5*/
  date: 11/09/21
public class AutoTypePromotion {

    // Argument data type req = double
    public void m(double a) {
        System.out.println(a + a);
    }
    // Argument data type req = float
    public void m(float a) {
        System.out.println(a + a);
    }

    public static void main(String[] args) {
        AutoTypePromotion atp = new AutoTypePromotion();
        atp.m( a: 200L);
    //Argument is of type char
    }
}
```

double required

float required

long passed

### Auto type promotion flow




### Output

```
400.0
```