Object Oriented Programming

POLYMORPHISM

Polymorphism

- Polymorphism in Java is a concept by which we can perform a single action in different ways.
- Polymorphism means "many forms", and it occurs when we have many classes that are related to each other by inheritance

Polymorphism

- There are two types of polymorphism in Java:

- Compile-time polymorphism
- Runtime polymorphism

We can perform polymorphism in java by method overloading and method overriding.

Polymorphism

- Compile-time polymorphism
  - Method Overloading

- Runtime polymorphism
  - Method Overriding

Method Overloading

- If a class has multiple methods having same name but different in parameters, it is known as Method Overloading.

- Advantage of method overloading:

1. Method overloading increases the readability of the program.

2. This provides flexibility to programmers so that they can call the same method for different types of data.

Method Overloading

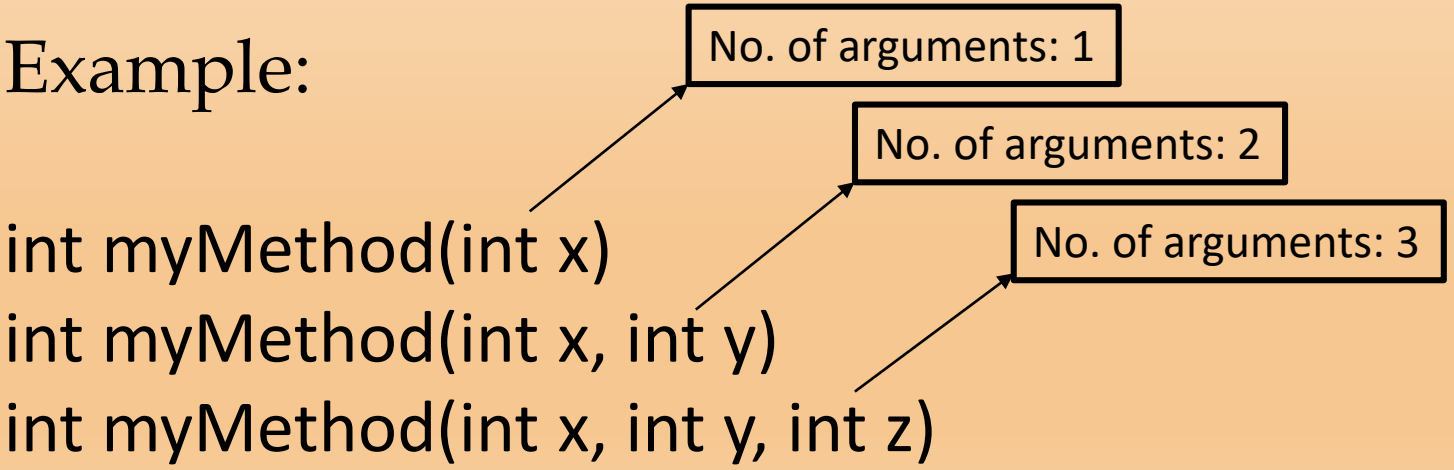
There are two ways to overload the method in java

- By changing number of arguments
- By changing the data type

Method Overloading: changing no. of arguments/ parameters

- Multiple methods having same name
- Different number of arguments in methods

Example:



Method Overloading: changing no. of arguments

Two methods have been created:

- The first add() method performs addition of two numbers
- Second add() method performs addition of three numbers

```
class Adder{
static int add(int a,int b){
      return a+b;
}
static int add(int a,int b,int c){
      return a+b+c;
}}
class TestOverloading1{
public static void main(String[] args){
      System.out.println(Adder.add(11,11));
      System.out.println(Adder.add(11,11,11));
}}
```

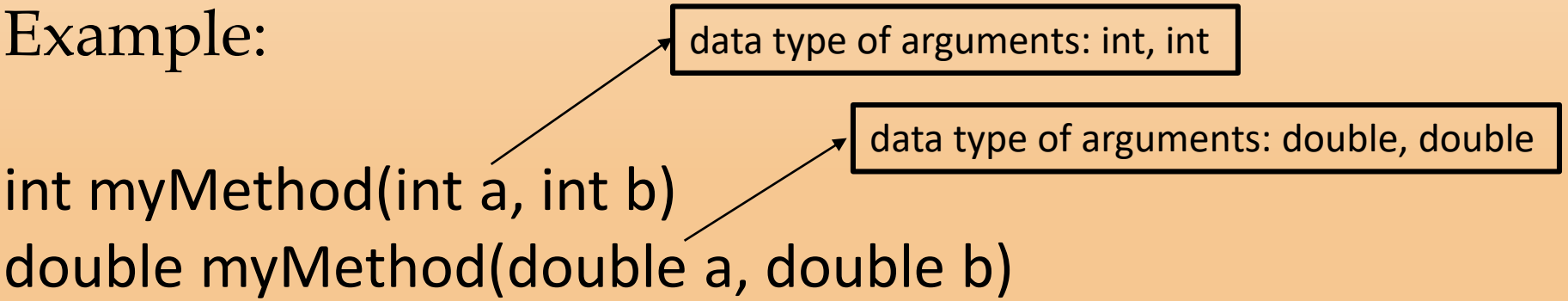
Output:

```
22
33
```

Method Overloading:
changing data type of arguments / parameters

- Multiple methods having same name
- Different data type of arguments in methods

Example:



Method Overloading: changing data type of arguments

Two methods have been created that differs in data type:

- The first add() method receives two integer arguments
- Second add() method receives two double arguments

```
class Adder{
static int add(int a, int b){
      return a+b;
}
static double add(double a, double b){
      return a+b;
}}
class TestOverloading2{
public static void main(String[] args){
      System.out.println(Adder.add(11,11));
      System.out.println(Adder.add(12.3,12.6));
}}
```

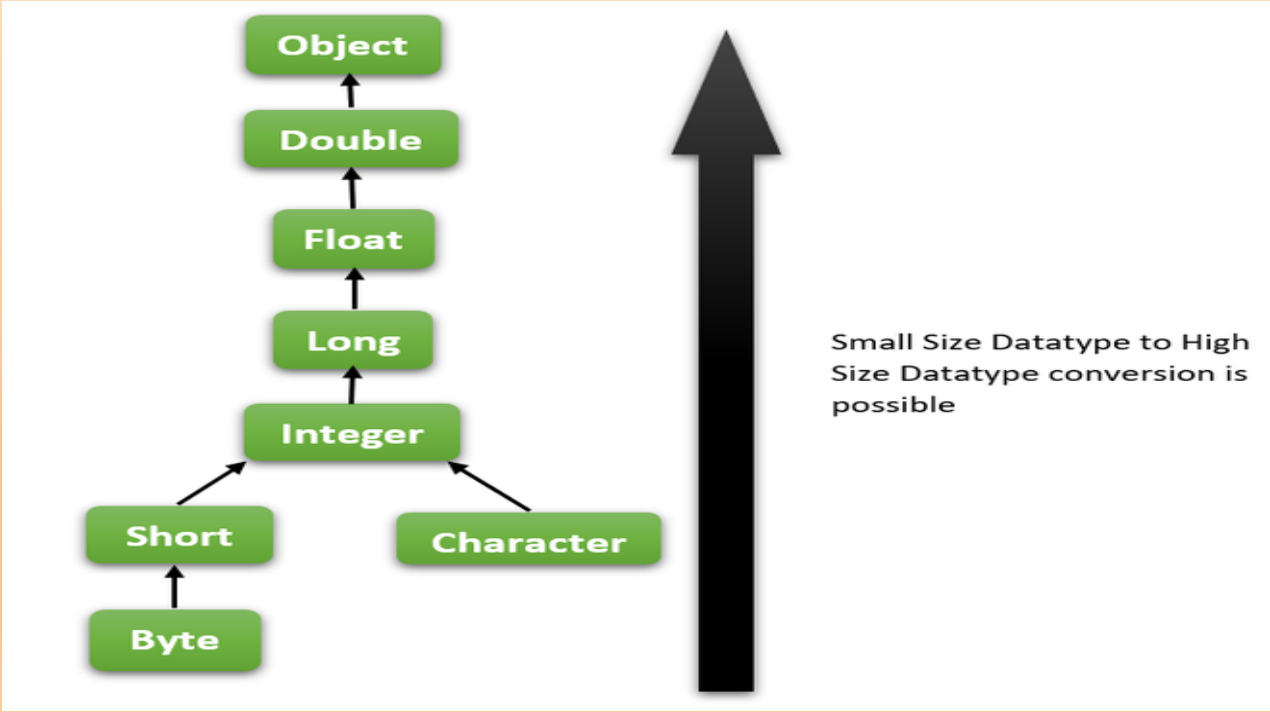
Output:

22

24.9

Method Overloading and Type Promotion

- One type is promoted to another implicitly if no matching datatype is found.



Example of Method Overloading with TypePromotion

```
class OverloadingCalculation1{
  void sum(int a,long 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);  //now second int literal will be promoted to long
  obj.sum(20,20,20);
  }
}
```

Output:

```
40
60
```

Example of Method Overloading with Type Promotion if matching found

- If there are matching type arguments in the method, type promotion is not performed.

```
class OverloadingCalculation2{
  void sum(int a,int b){
        System.out.println("int arg method invoked");
}
  void sum(long a,long b){
        System.out.println("long arg method invoked");
}
  public static void main(String args[]){
  OverloadingCalculation2 obj=new OverloadingCalculation2();
  obj.sum(20,20);  //now int arg sum() method gets invoked
  }
}
```

Output:

int arg method invoked

Example of Method Overloading with Type Promotion in case of ambiguity

- If there are no matching type arguments in the method, and each method promotes similar number of arguments, there will be ambiguity.

```
class OverloadingCalculation3{
  void sum(int a,long b){
        System.out.println("a method invoked");
        }
  void sum(long a,int b){
        System.out.println("b method invoked");
}
  public static void main(String args[]){
  OverloadingCalculation3 obj=new OverloadingCalculation3();
  obj.sum(20,20);  //now ambiguity
  }
}
```

Output:

Compile Time Error

Practice

1. Write a Java Program to Find Area of Square Using Method Overloading. (You have to find the area for both **int** and **double** values)

2. Suppose, you have to design a **Max** class where there will be method to find the maximum number. The user can provide two or three numbers as inputs to find the maximum among them. Now, write the code according to the description using method overloading.

Thank You