`BinaryOperator<T>`#

`BinaryOperator<T>` is a functional interface that inherits from `BiFunction<T, T, T>` interface. The `BinaryOperator<T>` interface takes only one parameter as compared to `BiFunction<T, T, T>`, which takes three parameters(two operands type and one result type).

Both the input objects and the result are of the same type in `BinaryOperator<T>`.

Below are the few interfaces that come under the `BinaryOperator<T>` category.

`BinaryOperator<T>`	Represents an operation upon two operands of the same type, producing a result of the same type as the operands (reference type)	`T apply(T t, T u)`
`DoubleBinaryOperator`	Accepts two double-value operands and produces a double-value result	`double applyAsDouble(double left, double right)`
`IntBinaryOperator`	Accepts two int-value operands and produces an int-value result	`int applyAsInt(int left, int right)`
`LongBinaryOperator`	Accepts two long-value operands and produces a long-value result.	`applyAsLong(long left, long right)`

Example#

```
import java.util.function.BinaryOperator;

public class BinaryOperatorDemo {


    public static void main(String args[]) {
        Person person1 = new Person("Alex", 23);
        Person person2 = new Person("Daniel", 56);
        BinaryOperator<Person> operator = (p1, p2) -> {
            p1.name = p2.name;
            p1.age = p2.age;
            return p1;
        };

        operator.apply(person1, person2);
        System.out.println("Person Name: " + person1.getName() + " Person Age: " + person1.getAge());
    }
}

class Person {
    String name;
    int age;

    Person() {
    }

    Person(String name, int age) {
        this.name = name;
```

Run Save Reset