Date:..............................

Item 24: Declare non-member functions when type conversions should apply to all parameters.

Object oriented principles + since we can apply multiplicat of rational numbers operator* should be member function of class Rational.

```
class Rational {
  public:
    Rational (int numerator = 0,
              int denominator = 1);
  //ctor is not explicit; allows implicit
    int to Rational conversions.

    int numerator() const.
    int denominator() const;

  ~~private~~:
     const Rational operator*(
         const Rational& rhs) const);
};

Rational OneEighth(1, 8);
Rational oneHalf(1, 2);

Rational result = oneHalf * oneEighth;

result = result * oneEighth; //both
                              are
                              fine
```

Date:..............................

Now, we can have mixed mode operations, where Rational can be multiplied with for example - ints

① result = oneHalf * 2; // fine

② result = 2 * oneHalf; // error

→ result = oneHalf.operator*(2);

→ result = 2.operator*(oneHalf);

→ Object oneHalf is an instance of Rational that contains operator* so compilers call that function while Integer 2 has no associated class.

→ Compilers will also look for non-member operator*

In ① implicit conversion takes place i.e. above line can be seen as

```
const Rational temp(2);

result = oneHalf * temp;

// same as oneHalf.operator*(temp);
```

Date:..............................

//compilers do this coz non-explicit Ctor is involved, if Rational ctor were explicit, neither of would have compiled.

```
result = oneHalf * 2;
result = 2 * oneHalf;    // Error.
```

=> Parameters are eligible for implicit type conversion only if they are listed in the parameter list.

that is why.

```
        result = oneHalf * 2 = oneHalf.operator*(2);
//works

result = 2 * OneHalf = 2 * operator*(oneHalf);
      //does not work.
```

=> only way is to make operator * a non-member function.

```
class Rational {
      ....        //contains no operator*

};
```

Date:...............................

```
const Rational operator*(const Rational& lhs,
                         const Rational& rhs)
{
  return Rational(lhs.numerator()*rhs.nume
  lhs.denominator() * rhs.denominator());
}  //non-member functn.

Rational oneFourth(1, 4);
Rational result;

result = oneFourth * 4; //fine

result = 4 * oneFourth; //works
                           aswell.
```

⇒ Next Questn is should operator* be friend functn?

Answer is we should avoid friend as much as we can.