`operator<<` AND `operator<<` WITH WHOLE-PART

Overloading the I/O operators with

composition and aggregation

Delroy A. Brinkerhoff

COMPOSITION AND AGGREGATION: REVIEWING WHOLE-PART

COMPOSITION

- One way – the whole "knows about" the part
- Programs build *composition* relationships by nesting the part object inside the whole
- Established in a constructor
- Unchangeable after construction

AGGREGATION

- One way – the whole "knows about" the part
- Programs build *aggregation* relationships by defining a pointer member in the whole pointing to the part
- Established at any time
- Changeable at any time

COMPOSITION

PART

```
class part
{
    private:
        string    name;
        double    cost;
    public:
};
```

WHOLE

```
class whole
{
    private:
        part      my_part;
        int       simple;
    public:
};
```

INSERTER WITH COMPOSITION

```
friend ostream& operator<<(ostream& out, part& me)
{
        out << me.name << endl;
        out << me.cost << endl;
        return out;
}

friend ostream& operator<<(ostream& out, whole& me)
{
        out << me.my_part << endl;
        out << me.simple << endl;
        return out;
}
```

EXTRACTOR WITH COMPOSITION

```
friend istream& operator>>(istream& in, part& me)
{
        in >> me.name;
        in >> me.cost;
        return in;
}

friend istream& operator>>(istream& in, whole& me)
{
        in >> me.my_part;
        in >> me.simple;
        return in;
}
```

AGGREGATION

PART

```
class part
{
    private:
        string    name;
        double    cost;
    public:
};
```

WHOLE

```
class whole
{
    private:
        part*    my_part;
        int      simple;
    public:
};
```

INSERTER WITH AGGREGATION

```
friend ostream& operator<<(ostream& out, part& me)
{
        out << me.name << endl;
        out << me.cost << endl;
        return out;
}

friend ostream& operator<<(ostream& out, whole&  me)
{
    if (me.my_part != nullptr)
        out << *me.my_part << endl;
    out << me.simple << endl;
    return out;
}
```

EXTRACTOR WITH AGGREGATION

```
friend istream& operator>>(istream& in, part& me)
{
        in >> me.name;
        in >> me.cost;
        return in;
}

friend istream& operator>>(istream& in, whole& me)
{
    if (me.my_part != nullptr)
        in >> *me.my_part;
    in >> me.simple;
    return in;
}
```