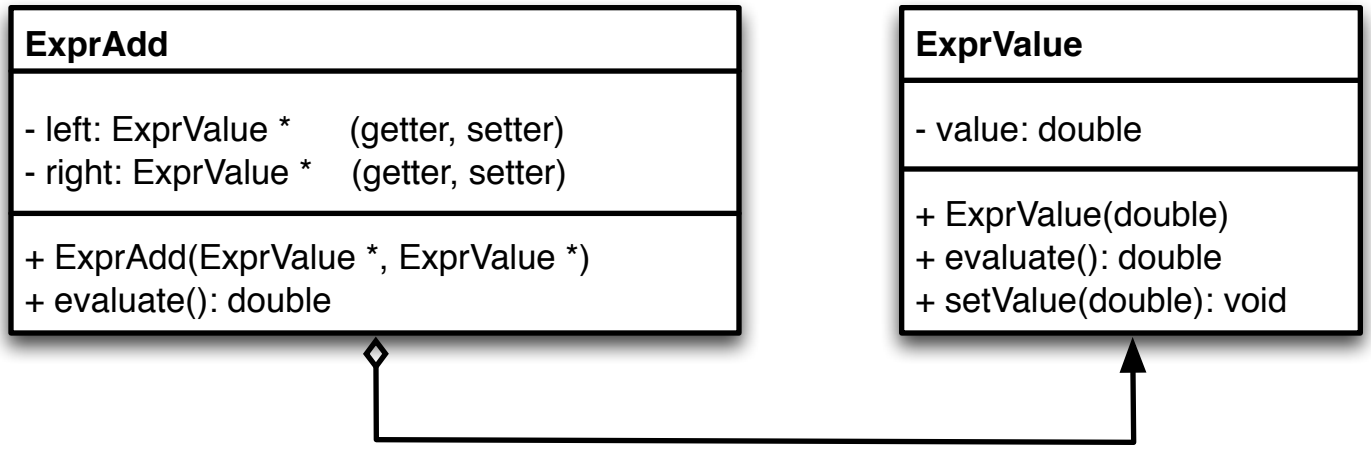
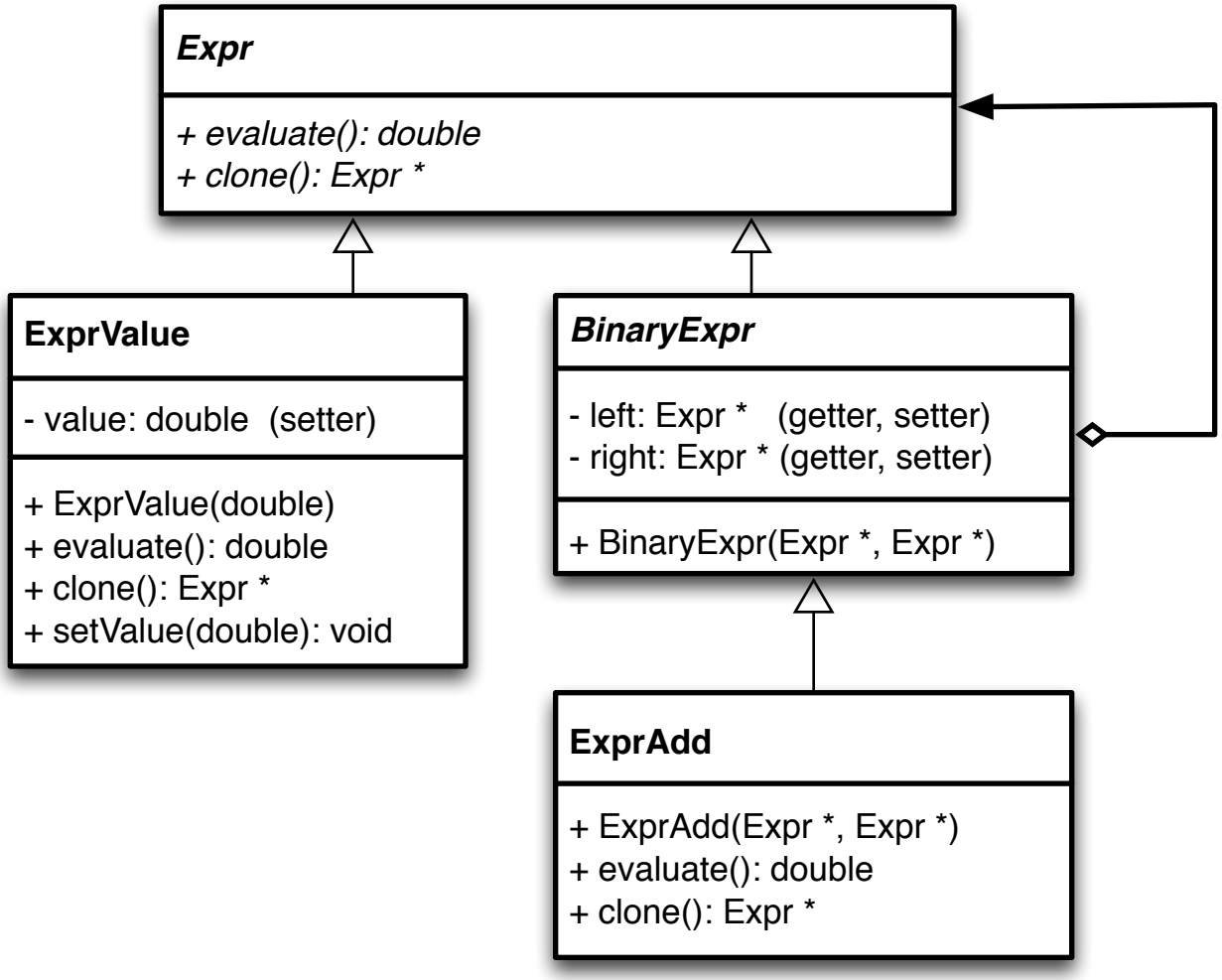
Current Version:





```
class ExprValue {
private:
    double value_;  // 8B big

public:
    ExprValue();
    ExprValue(double value);
    ExprValue(const ExprValue &toCopy);

    double getValue() const { return value_; }
    void setValue(double value) { value_ = value; }
    double operator+(const ExprValue &);
};
```

```
ExprValue::ExprValue(double value) : value_(value) {}

ExprValue::ExprValue() : ExprValue(0.0) {}

ExprValue::ExprValue(const ExprValue &toCopy) {
    value_ = toCopy.value_;
}

double ExprValue::operator+(const ExprValue &other) {
    return value_ + other.value_;
}
```

```
class ExprAdd {
public:
    ExprAdd(ExprValue *left, ExprValue *right);
    ExprAdd(ExprAdd &);
    virtual ~ExprAdd();

    ExprAdd &operator=(const ExprAdd&);
    ExprValue *getLeft() const { return left; }
    void setLeft(ExprValue *left) { ExprAdd::left = left; }
    ExprValue *getRight() const { return right; }
    void setRight(ExprValue *right) { ExprAdd::right = right; }

private:
    ExprValue *left;
    ExprValue *right;
};
```

```
ExprAdd::ExprAdd(ExprValue *left, ExprValue *right) :
        left(left), right(right) {}

ExprAdd::ExprAdd(ExprAdd &in) {
    left = new ExprValue(*in.left);
    right = new ExprValue(*in.right);
}

ExprAdd &ExprAdd::operator=(const ExprAdd &in) {
    left = new ExprValue(*(in.left));
    right = new ExprValue(*(in.right));
    return *this;
}

ExprAdd::~ExprAdd() {
    delete left;
    delete right;
}
```