

ISOM Project Cluster 2005

Exercise Sheet 2

Søren Debois
debois@itu.dk

September 12, 2005

Here are basic exercises in the C++ class concept. Feel free to modify the exercises as you see fit. Also, please e-mail me if you have any questions, or feel that the exercises below are not enough.

Exercise 1. Add all special members to the `Stack` class that you can think of. Also do this for the `Stack` with dynamic allocation.

Exercise 2. The class `aptr` from todays lecture did contained neither default constructor, copy constructor or copy assignment. Explain why this is bad. Add these members. (Hint: it will be necessary to change the destructor. More hint: You can safely assume that a valid pointer is never equal to 0.)

Exercise 3. Here is a base class for the implementation of arithmetic expressions using the “Composite” pattern:

```
class Expr {
public:
    virtual int eval();
};
```

Make this class abstract. Point out what’s wrong with its destructor.

Here are two derived classes:

```
class Num {
public:
    Num(int n) : n_(n) {}
    virtual int eval() { return n_; }
private:
    int n_;
};

class Mult {
public:
    Mult(Expr* e1, Expr* e2) : e1_(e1), e2_(e2) {}
    virtual int eval() { return e1->eval() * e2->eval(); }
private:
    Expr* e1_;
```

```
    Expr* e2_;  
};
```

The idea is that the expression $2 \times 3 \times 8$ is represented by the statement

```
Expr* p = new Mult(new Mult(new Num(2), new Num(3)), new Num(8));
```

Does `Mult` need a destructor? Does `Num`? Implement any destructors you think are missing. Implement a class for addition. Implement a member `void print()` for printing the expression (`p->print()` should display `"((2 * 3) * 8)"`).