technische universität
dortmund

Department of Computer Science
Chair 11 / Algorithm Engineering

Dr. Carsten Gutwenger

Winter 2011/12

Object-oriented Programming
Assignment Sheet No. 9

Date: December 20

Exercise 9.1 (Inheritance and Virtual Functions)

Reimplement the classes in Exercise 8.2 using virtual functions for `area` and `circumference`. Declare these functions as pure virtual in `Shape` and implement them in derived classes. Use the following `print` function for testing your implementations:

```
void print(Shape &s)
{
    cout << "area          = " << s.area() << endl;
    cout << "circumference = " << s.circumference() << endl;
}
```

Exercise 9.2 (Inheritance and Virtual Functions)

Implement classes for vehicles using inheritance:

- A `Vehicle` has a name, a number of wheels, and a number of seats.
- A `Car` is a vehicle with at least 4 seats and 4 wheels; additionally cars have a color.
- A `Motorbike` is a vehicle with 2 wheels and 1 or 2 seats.

Make sure that all data members are only accessible vie get/set methods, which write errors to the console when invalid parameters have been passed.

Vehicles shall also have a current position, which is initialized to 0; when they drive this position is changed. Implement a member function `int drive(int dist)` which advances the current position by `dist` and returns it. Cars can drive forward and backward, motorbikes only forward. Override the `drive` method for motorbikes, such that an error message is printed when a negative argument is passed for parameter `dist`.

Implement an output operator for vehicles that prints all information about the vehicle including its type (vehicle, car, or motorbike). (*Hint:* Use a virtual `print` method.)

Exercise 9.3 (Inheritance, Virtual Functions and Pointers)

Implement a class hierarchy for arithmetic expressions.

- The base class of the hierarchy shall be the abstract class `Expression` with a pure virtual function `double evaluate()`.
- `ConstExp` shall represent just a constant; its constructor shall take a constant x and its value is x. Furthermore, a `ConstExp` shall have a set method for changing the constant.
- `AddExp` shall represent an addition of two expressions; its constructor shall take two pointers to expressions e_1 and e_2 and its value is the sum of the values of e_1 and e_2.
- Analogously, implement expressions `SubExp`, `MulExp`, and `DivExp`.
- `SqrtExp` shall represent the square root function; its constructor takes a pointer to an expression e and its value is the square root of the value of e.

Override `evaluate` in derived classes such that it returns the value of the expression. Take care to deal with potential errors (null pointers, division by zero, square root of a negative number).

Example: The following code sequence

```
ConstExp cexp1(10.0);
ConstExp cexp2(64.0);

SqrtExp sqrtexp(&cexp2);
AddExp addexp(&cexp1, &sqrtexp);

cout << addexp.evaluate() << endl;
cexp1.set(20.5);
cout << addexp.evaluate() << endl;
```

shall print:

```
18
28.5
```