C++ Inheritacne

Sascha Zelzer

MBI@DKFZ

11.05.2011

C++ Inheritance

Fundamental features in OOP:

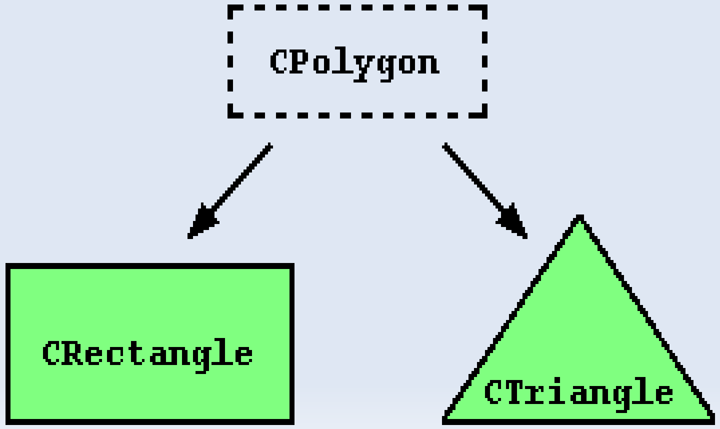
- Encapsulation
- Polymorphism
- **Inheritance**

Class Relationships

Two kinds of relationships between classes:

- ”part-of”
- ”kind-of”

Inheritance allows to create classes which are derived from other classes (”kind-of” relationship).



Class Relationships (Example)

- ”part-of” (composition, aggregation)

```
class Engine {};
class Car {
    private: Engine _engine;
};
```

- ”kind-of” (inheritance)

```
class Vehicle {};
class Car : public Vehicle { … };
```

Access Specifiers

- Inheritance access specifiers:

  public, *protected*, *private*

```
class Car : private Engine {
    ...
};
```

$\longrightarrow$ syntactic variant of composition

Access	public	protected	private
Members of the same class	yes	yes	yes
Members of derived classes	yes	yes	no
Non-members	yes	no	no

Use composition when you can,
private inheritance when you have to!

Substitution Principle

If **S** is a derived type of **T**, then objects of type **T** in a program may be replaced with objects of type **S** without altering any of the desirable properties of that program.

```
class Rectangle {
public:
  int getWidth() const; int getHeight() const;
  void setWidth();       void setHeight();
};

class Square : public Rectangle {};
```

Guidelines

- Favor composition over inheritance
- Never hide member functions from base classes in your derived class
- Use Abstract Base Classes (ABC) to create *interfaces*

Questions?