OOP #4

OBJECT-ORIENTED PROGRAMMING #4

ŁUKASZ ZIOBROŃ

AGENDA

1. 4 pillars of objectivity
2. Abstraction
3. Encapsulation
4. Inheritance
5. Polymorphism

THE FOUR PILLARS OF OBJECTIVITY

THE FOUR PILLARS OF OBJECTIVITY

- Abstraction
- Encapsulation
- Inheritance
- Polymorphism

ABSTRACTION

ABSTRACTION

- Interface
  - The public part of a class
    - Member function - obvious
    - Non-member function
    - Member types
    - Member fields
    - Template parameters
    - Specializations
  - Example: `std::vector` on cppreference.com
  - The private part (implementation) is unknown
- Object Oriented Design (OOD)

> *Make interfaces easy to use correctly and hard to use incorrectly*
>
> *-- Scott Meyers, Effective C++*

BAD INTERFACE EXAMPLE

```
// A date class which is easy to use but also easy to use wrong.
class Date {
   public:
      Date(int month, int day, int year);
      ...
};

// Both are ok, but some european programmer may use it wrong,
// because european time format is dd/mm/yyyy instead of mm/dd/yyyy.
Date d(3, 4, 2000);
Date d(4, 3, 2000);
```

ENCAPSULATION

ENCAPSULATION

- Access specifiers
  - `public` - `struct` default
  - `protected`
  - `private` - `class` default
- Setters and getters
- Unnamed namespaces

INHERITANCE

INHERITANCE

- Constructors and destructors call order
  - Constructors - base class first, then derived
  - ideone.com
- Diamond problem
  - virtual inheritance
- `class` from `struct` inheritance is...
  - `private`
- `struct` from `class` inheritance is...
  - `public`

INHERITANCE ACCESS MODIFIERS

	`public`	`protected`	`private`
`public`	public	protected	private
`protected`	protected	protected	private
`private`	private	private	private

POLYMORPHISM

POLYMORPHISM

- Virtual functions
- Pure virtual functions (`=0`)
- Abstract classes
  - have at least one pure virtual function
- `vtable` and `vptr`
  - implementation of polymorphism
  - constructor of derived class overrides base class records in `vtable`

EXERCISE

CARS

1. Design proper abstraction (interfaces)
2. Apply inheritance
3. Fix encapsulation
4. Use polymorphism to represent every type of car, using a single pointer
5. Fix diamond problem
6. Fix potential memory leaks
7. Think about the way of keeping engines in cars. Should they be kept by a value, reference or a pointer (what kind of pointer)?
8. Is this code testable?

VIEW TASK IN REPO

POST-WORK

POST-WORK

You can work in groups or individually. Fork the Cars repo and submit a Pull Request after you have finished.

1. (4 XP) Create `InvalidGear` exception. It should be thrown when someone tries eg. change a gear from 5 to R. It should inherit from one of STL exceptions
2. (2 XP per fix) Fix interfaces to be easy to use correctly and hard to use incorrectly (like `accelerate(-999)`)
3. (10 XP - optional) Write a proper unit tests to this code
4. Read one of below articles. It will be useful for the next lesson
   - SOLID czyli dobre praktyki w programowaniu obiektowym (in Polish)
   - S.O.L.I.D: The First 5 Principles of Object Oriented Design (in English)

CODERS SCHOOL