Better Object-Oriented Design

OOP

- Object-oriented programming
- Object-oriented design

Cohesion

- Cohesion – how close is the information and actions in a class – classes should have single purpose
- Give examples???

- Orthogonality in a system
  - Positive - change the radio station does not change the volume
  - Negative – helicopter – changing speed, changes direction

Coupling

- Coupling – how a class is related to others
- It must be loose
- Why???
- Reusability
- Examples??

- Inheritance – the ability to get everything from a parent class.
- What is that ‘get everything’?
  - Get members
  - Get member-functions implementations
  - Get the type (the abstraction)
- When to use inheritance? To reduce copy-paste?

- Polymorphism – the ability to look at the child object like it is parent one
- Liskov principle
- JS, C++, Java, C#, Ruby, Python - how to do it? (virtual, override, interface, abstract, etc.)

- Abstraction – represent one thing, hide implementation, correct name and interface.
- Rules to follow:
  - Move not related methods (and group only the related ones) out of the class
  - Define the operations and think about their opposite
  - Think about levels of abstraction – DB layer, Model layer when you design your interfaces
  - When evolving an interface – think twice

- Encapsulation – is it only the visibility of the internal data of a class?
- Hide implementation details
- Keep state clean – validation in constructors
- Mutability – good or bad thing?

Encapsulation

- Rules
  - Never declare something public – do it private and think how to use it
  - If a method calls only public methods – make it public – wrong?
  - don’t depend on documentation (or implementation)
- Examples??

Inheritance or Composition

- Has-a vs is-a
- Final, sealed, abstract, virtual

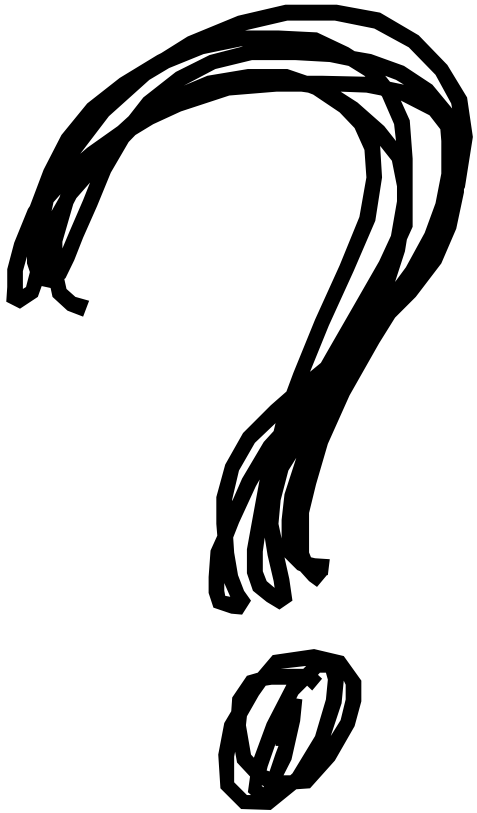
Inheritance

- Do you need an interface for a single implementation?
- When you design – draw your hierarchy and think about commonly used functionalities – where they should be? Higher = closer to the root? Think about the depth of that tree.
- Don’t override without adding specifics in implementation
- How to use ‘protected’?

Reasons to create a class

- For a real-world things
- Technical things
- Reduce/isolate/hide implementation details/ complexity


СВ. КЛИМЕНТ ОХРИДСКИ



THANK YOU