NORBIS

SENTER FOR
DIGITALT LIV
NORGE

Object-oriented design

David Grellscheid

UNIVERSITETET I BERGEN

The Abdus Salam
ICTP International Centre
for Theoretical Physics

Programming paradigm examples

Declarative / Imperative

Structured / Non-Structured

Procedural

Object-oriented

Functional

(Almost) any style can be implemented in any language

Grady Booch
“Object-oriented analysis and design”
2nd edition
Addison-Wesley, 1994

Main goal: manage complexity

Different approaches, OO is just one of them!

see e.g. Haskell for a completely different approach to complexity handling: functional programming

- Complexity of the problem domain

  external; requires software maintenance, evolution, preservation

- Development process

  impossible for one developer to understand large projects completely

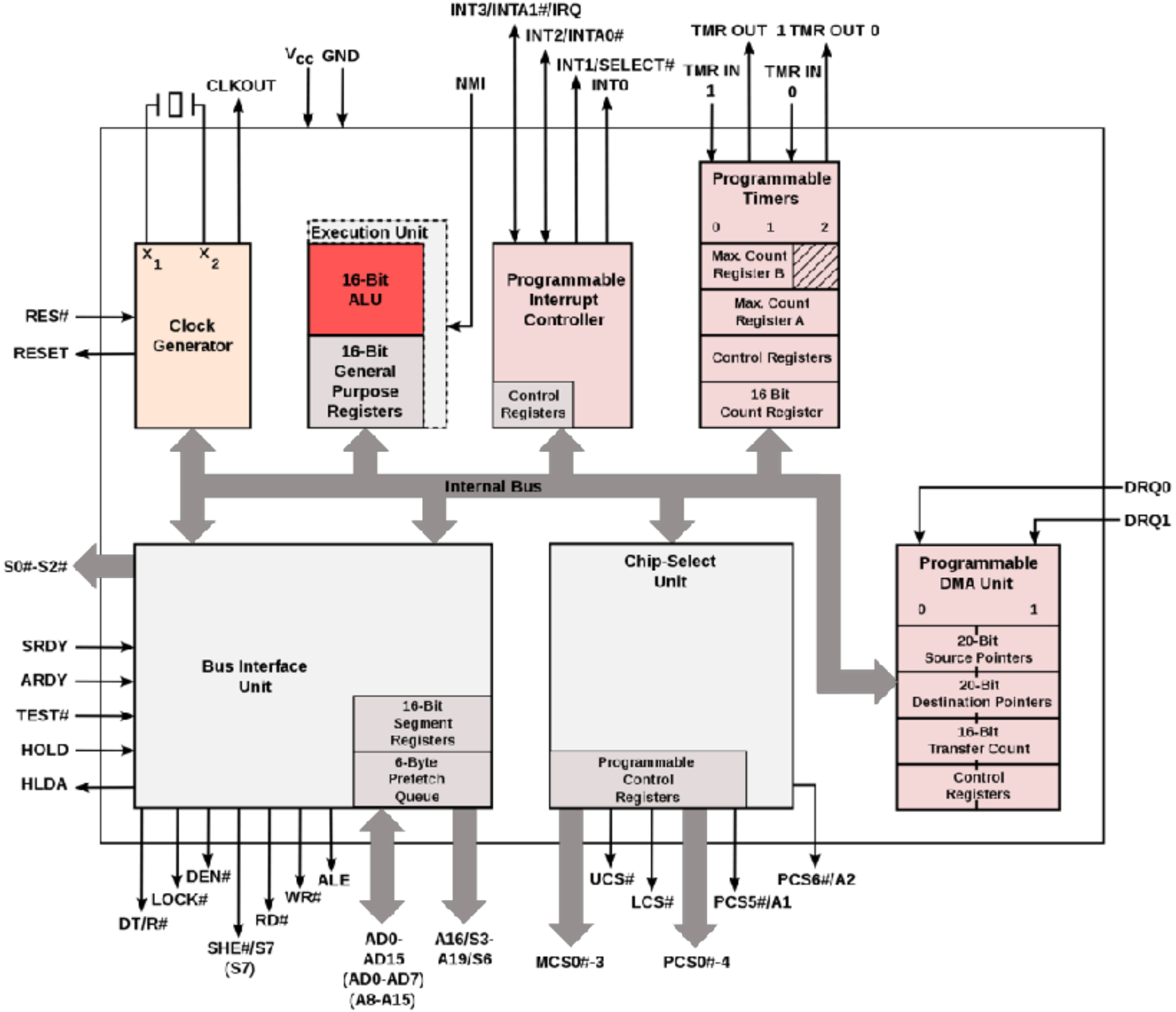
- Software is boundlessly flexible

  able to work at any level of abstraction; no fixed quality standards

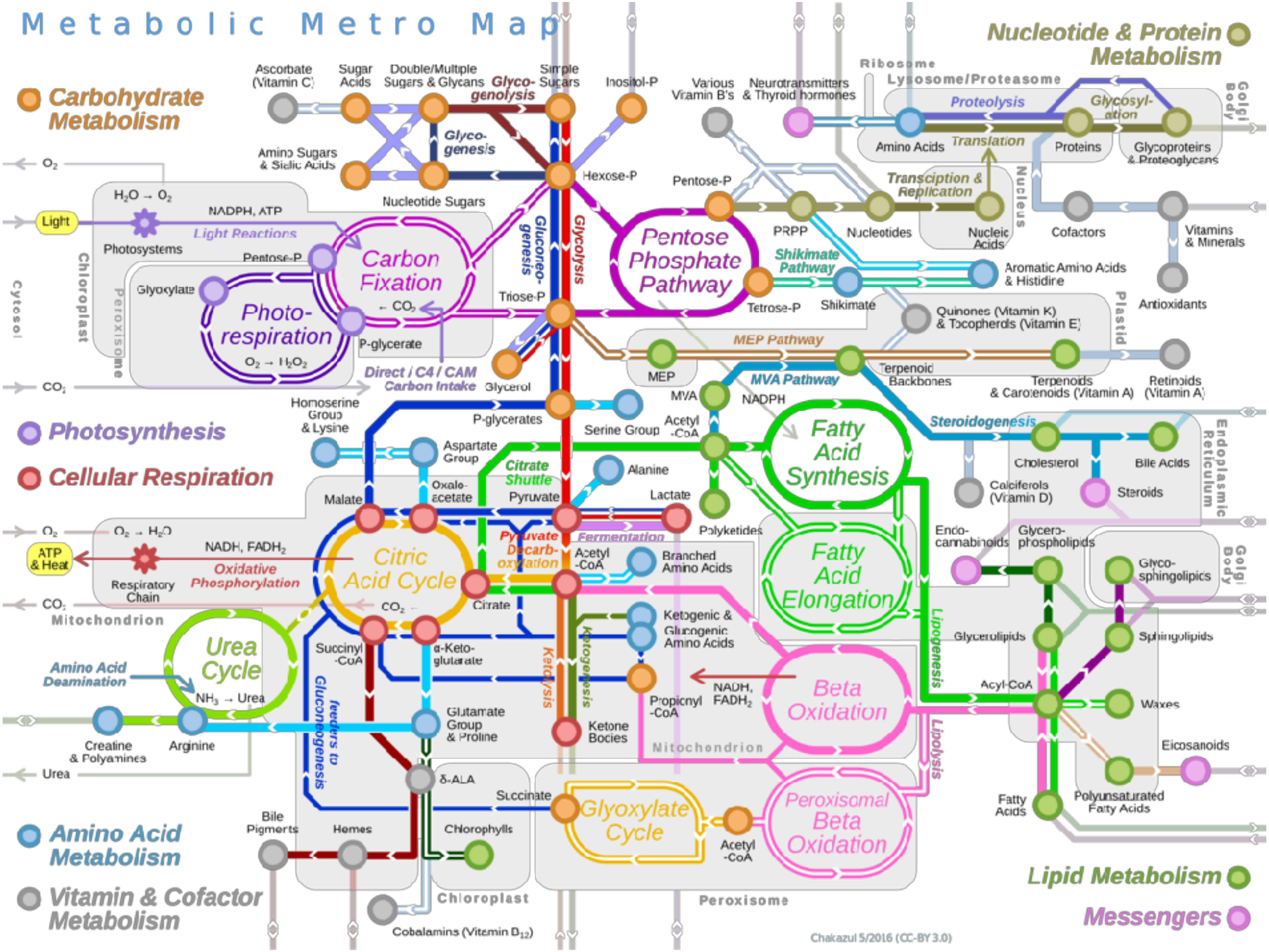
- Behaviour of discrete system

  natural world physics is local and continuous
  program state is not: combinatoric, small change -> large effect

Intel 80186 / 80188 architecture



Metabolic Metro Map



Chakazul 5/2016 (CC-BY 3.0)

- Complexity is hierarchical grouping of subsystems, down to elementary components
- Choice of elementary blocks is mostly arbitrary
- Links and interactions within a component are much stronger than between components
- Hierarchy uses only a few different subsystems in different combinations
- Working complex systems evolve from working simple systems

- Deal with complexity by decomposition
- Algorithmic decomposition:
  which steps in which order?
- OO decomposition:
  which "real-world" entities are involved?
  how do they relate to each other?

Object

- **State**: inner structure with current values
- **Behaviour**: external interaction and state changes (construct / destruct // modify / select / iterate)
- **Identity:** distinct to all other objects
  It's not the name, one object can have many names!
  Identity considerations are relevant when looking at copying, lifetime and ownership behaviour.

Class

Objects with common structure and behaviour belong to a **class**. The class defines both.

An object is an **instance** of a class.

Core features of OO design

- Abstraction
- Encapsulation
- Modularity
- Hierarchy

Abstraction

- Outside view of the object
- Focus on relevant details, ignore others
- Define distinction to other objects
- No surprises, no unexpected side behaviour

Abstraction

- Identify object invariants, properties that must be true at any time
- Operations have pre- and post-conditions, they must be satisfied
- Objects should never enter inconsistent state

Abstraction

- Implementation details do not matter here
- Define public member functions
- Private section doesn’t matter yet

Encapsulation

- separates object's tasks from each other
- actual implementation of the abstraction is hidden
- allows isolated implementation changes
- internal design changes in the objects do not impact the users of the objects

Encapsulation

- Abstractions only work well if implementation is encapsulated!

Modularity

- Grouping of classes into functionally related units. Modules should be loosely coupled externally.
- “Physical” collection of units in files, rather than abstract connections
- Difficult to get right first time, may need several redesigns during development

Hierarchy

- Abstractions form hierarchies
- Helps to think about the useful levels

Two main kinds:

- “is-a”: cat is an animal; oak is a plant
- “has-a”: car has an engine; house has a door

Hierarchy: “is-a”

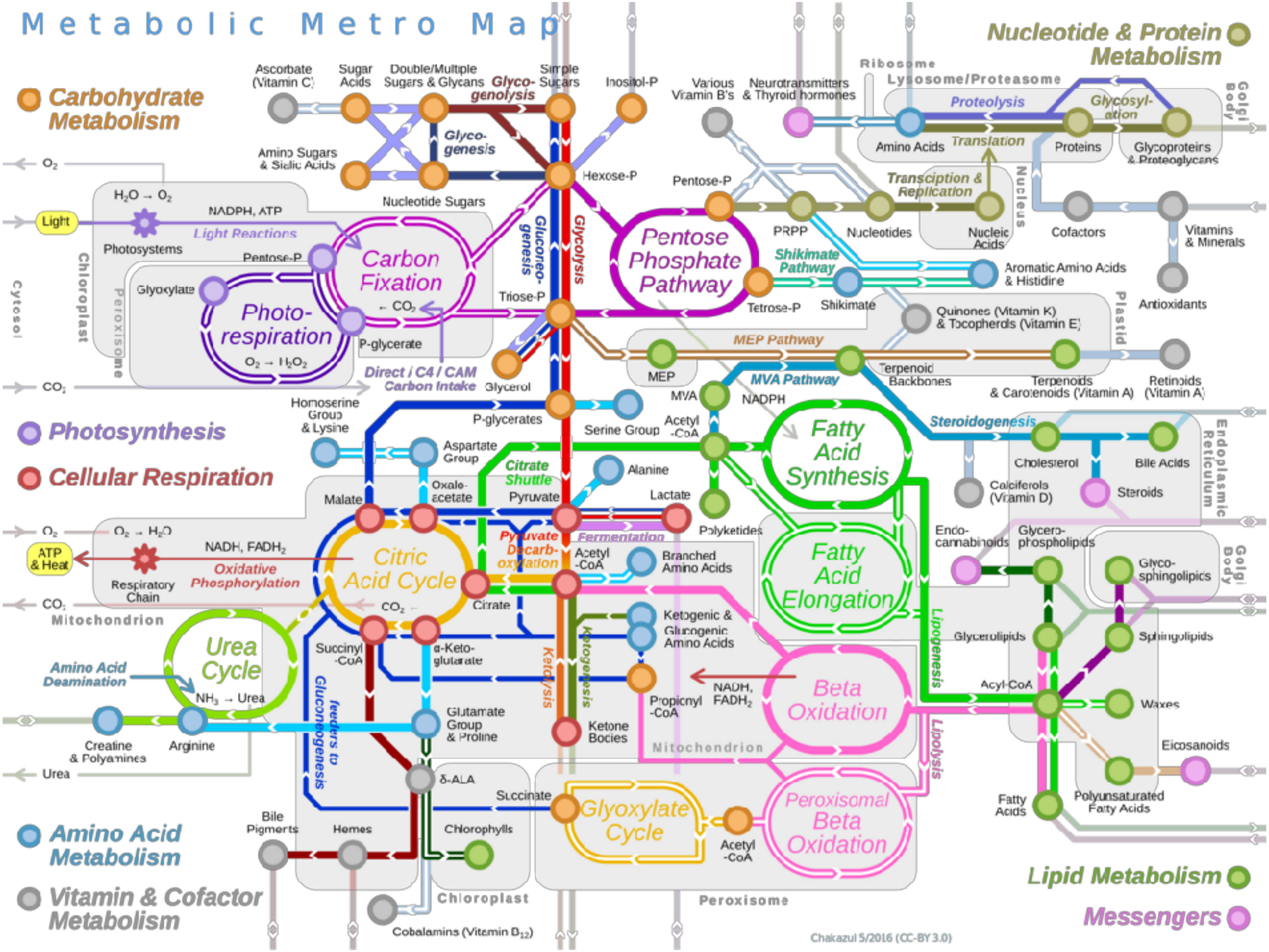
- Modelled by inheritance
- Common functionality moves to the top; applies to all classes down the hierarchy

Easy re-use of code alone is **not** a good reason for inheritance

Hierarchy: “has-a”

- Modelled by aggregation
- Objects have other objects as member variables

Metabolic Metro Map



Chakazul 5/2016 (CC-BY 3.0)

… main message …

Hierarchy

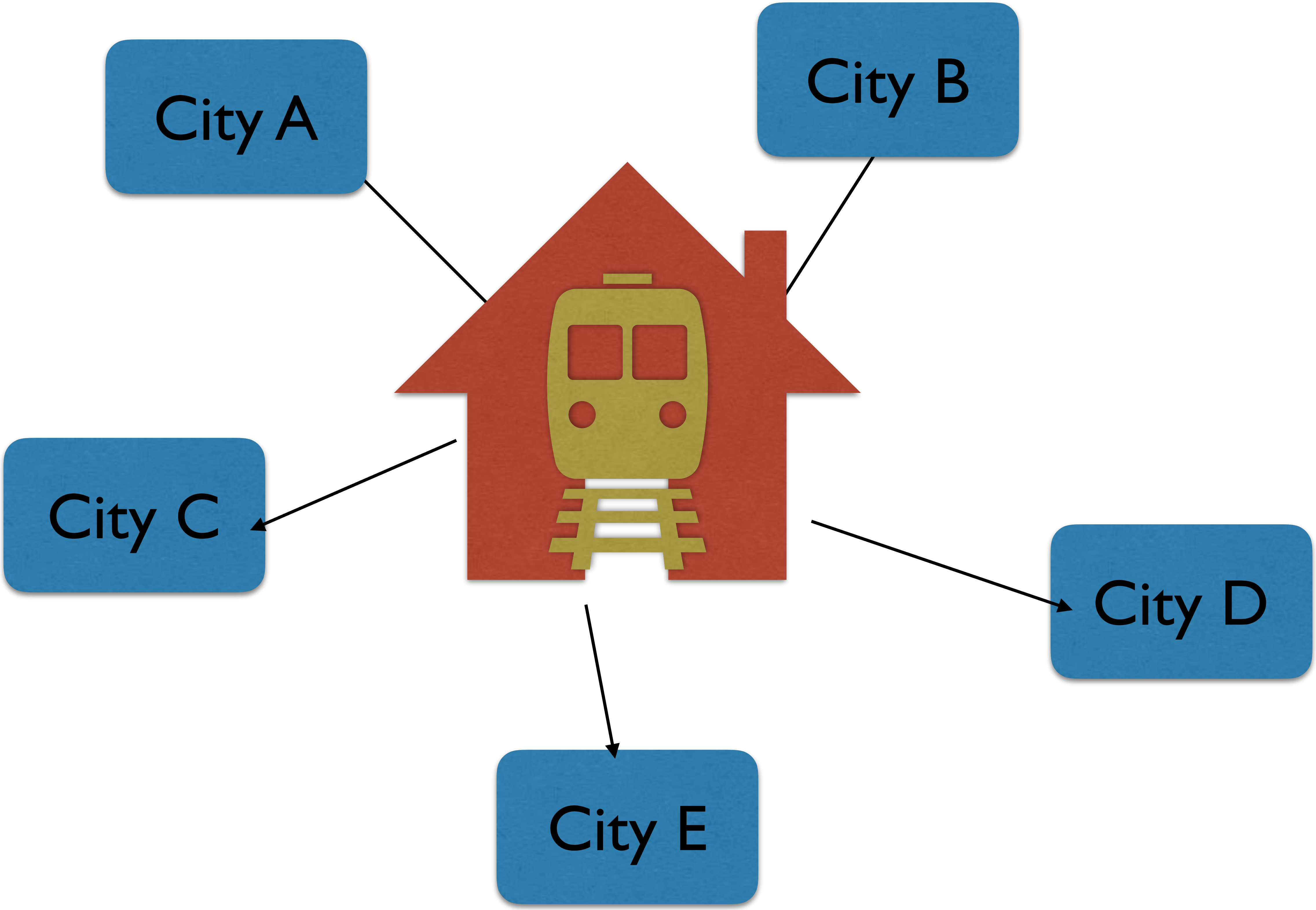
- Abstractions form hierarchies
- Helps to think about the useful levels

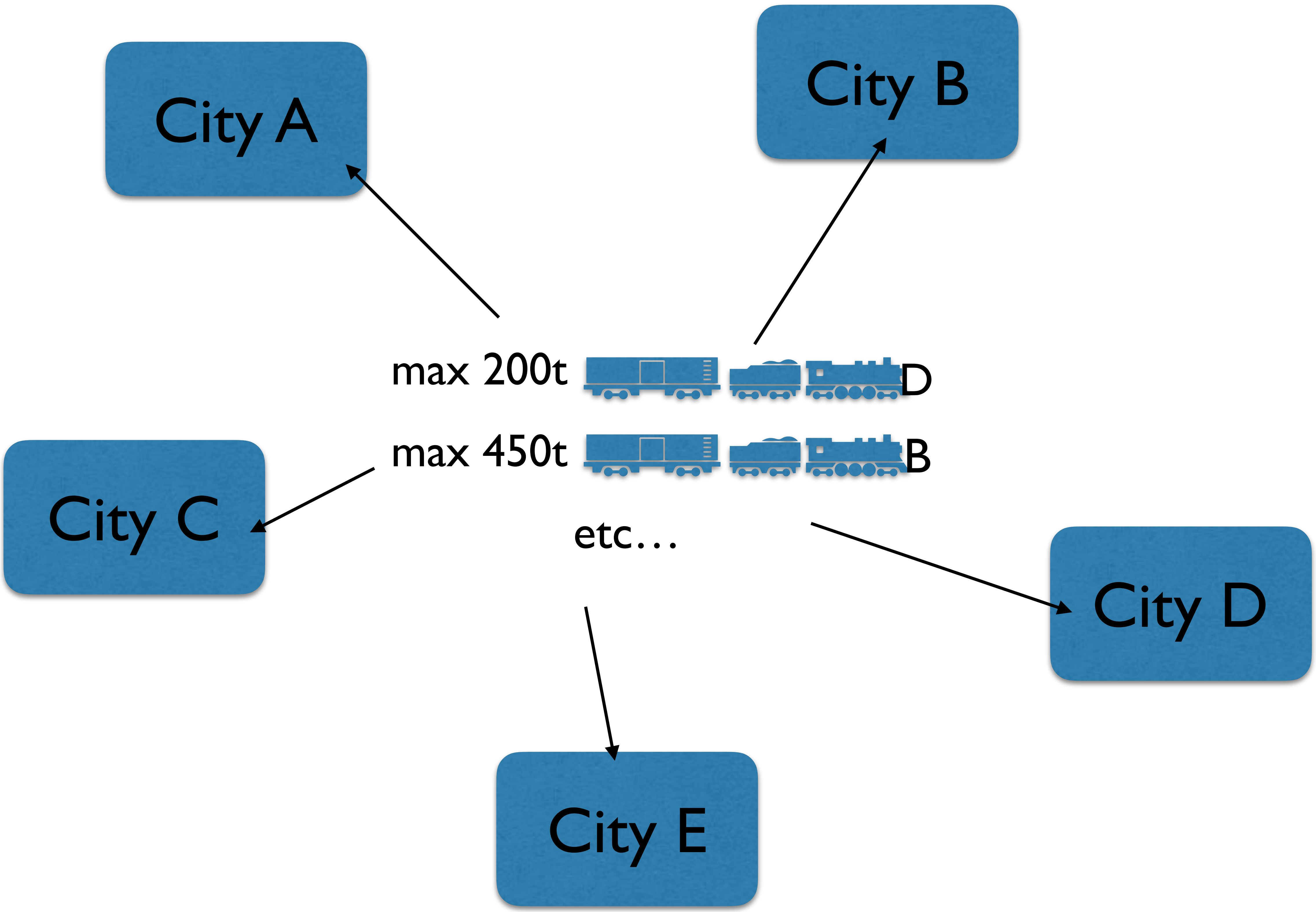
Two main kinds:

- “is-a”: cat is an animal; oak is a plant
- “has-a”: car has an engine; house has a door

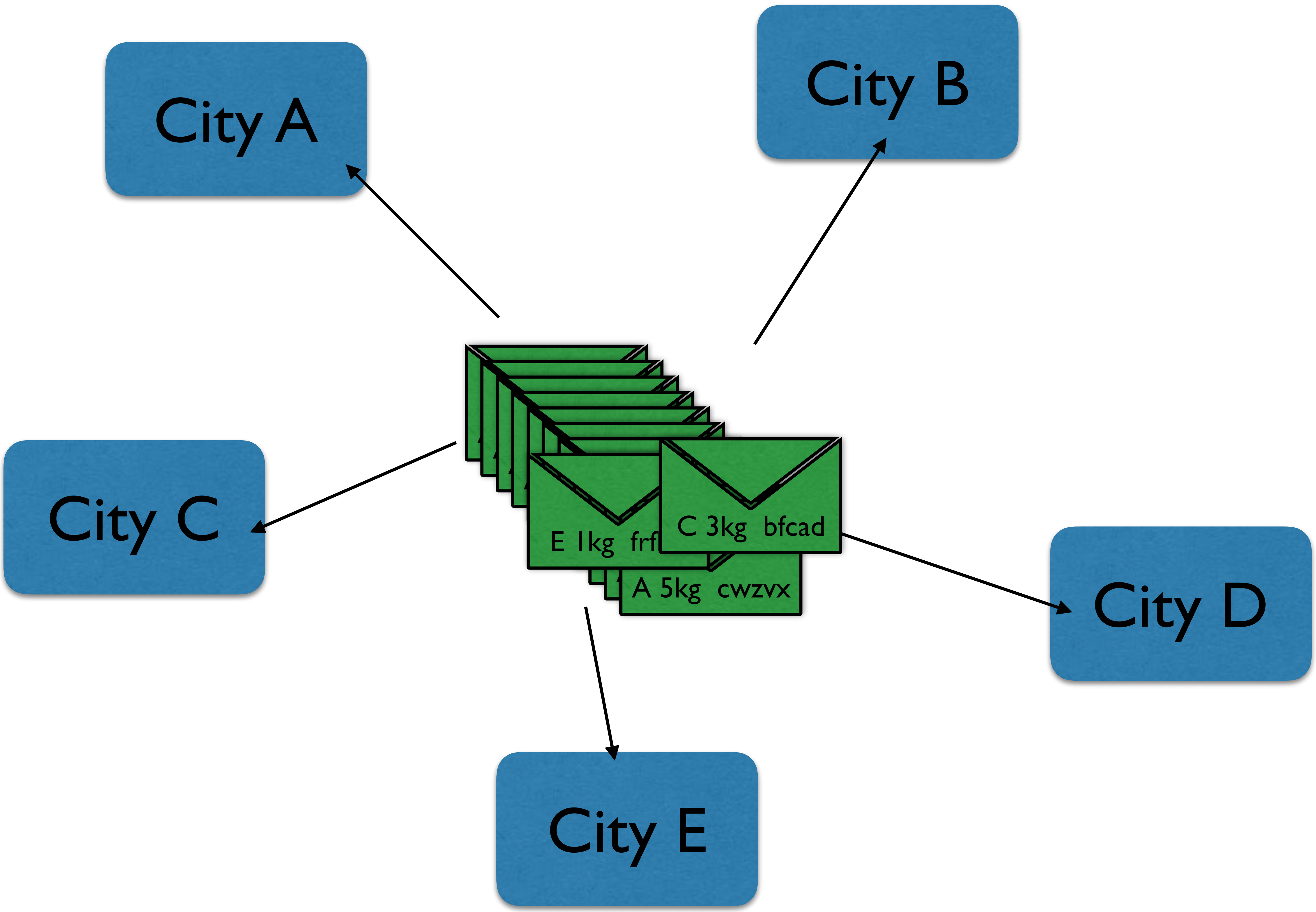
Exercise

Exercise: a freight station




City A
City B
max 200t
D
max 450t
B
etc…
City C
City D
City E

NORBIS SENTER FOR DIGITALT LIV NORGE

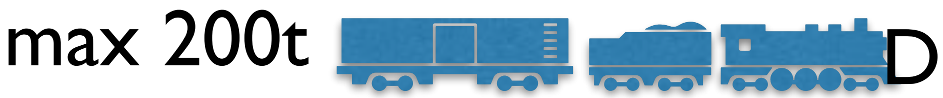


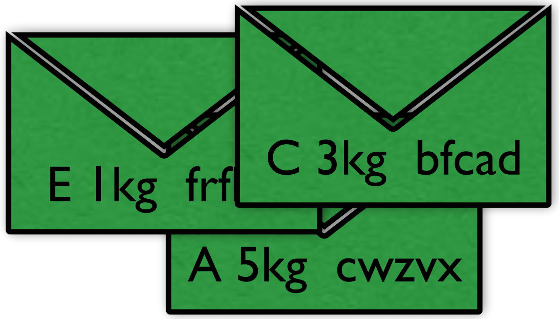
Design an OO model

classes, objects, interfaces, public/private, which methods/state

no implementation!

a random train arrives, is loaded with correct mail, leaves, and repeat




E 1kg frf
C 3kg bfcad
A 5kg cwzvx