Development Methodologies

Development Methodologies

- The following two methodologies are the primary ones in current use; both based on *divide-and-conquer*

Development Methodologies

- The following two methodologies are the primary ones in current use; both based on *divide-and-conquer*
  - » **Top-down design (TDD)**

Development Methodologies

- The following two methodologies are the primary ones in current use; both based on *divide-and-conquer*
  - » **Top-down design (TDD)**
    - > **Also known as functional decomposition**

Development Methodologies

- The following two methodologies are the primary ones in current use; both based on *divide-and-conquer*
  - » **Top-down design (TDD)**
    - > **Also known as functional decomposition**
  - » **Object-Oriented Design (OOD)**

Top-Down Design

- Consists of breaking the problem into a set of sub-problems called **modules**

Top-Down Design

- Consists of breaking the problem into a set of sub-problems called **modules**
- It creates a hierarchical structure of problems and sub-problems (modules)

Top-Down Design

- Consists of breaking the problem into a set of sub-problems called **modules**
- It creates a hierarchical structure of problems and sub-problems (modules)
- This process continues for as many levels as it takes to expand every task to the smallest details.

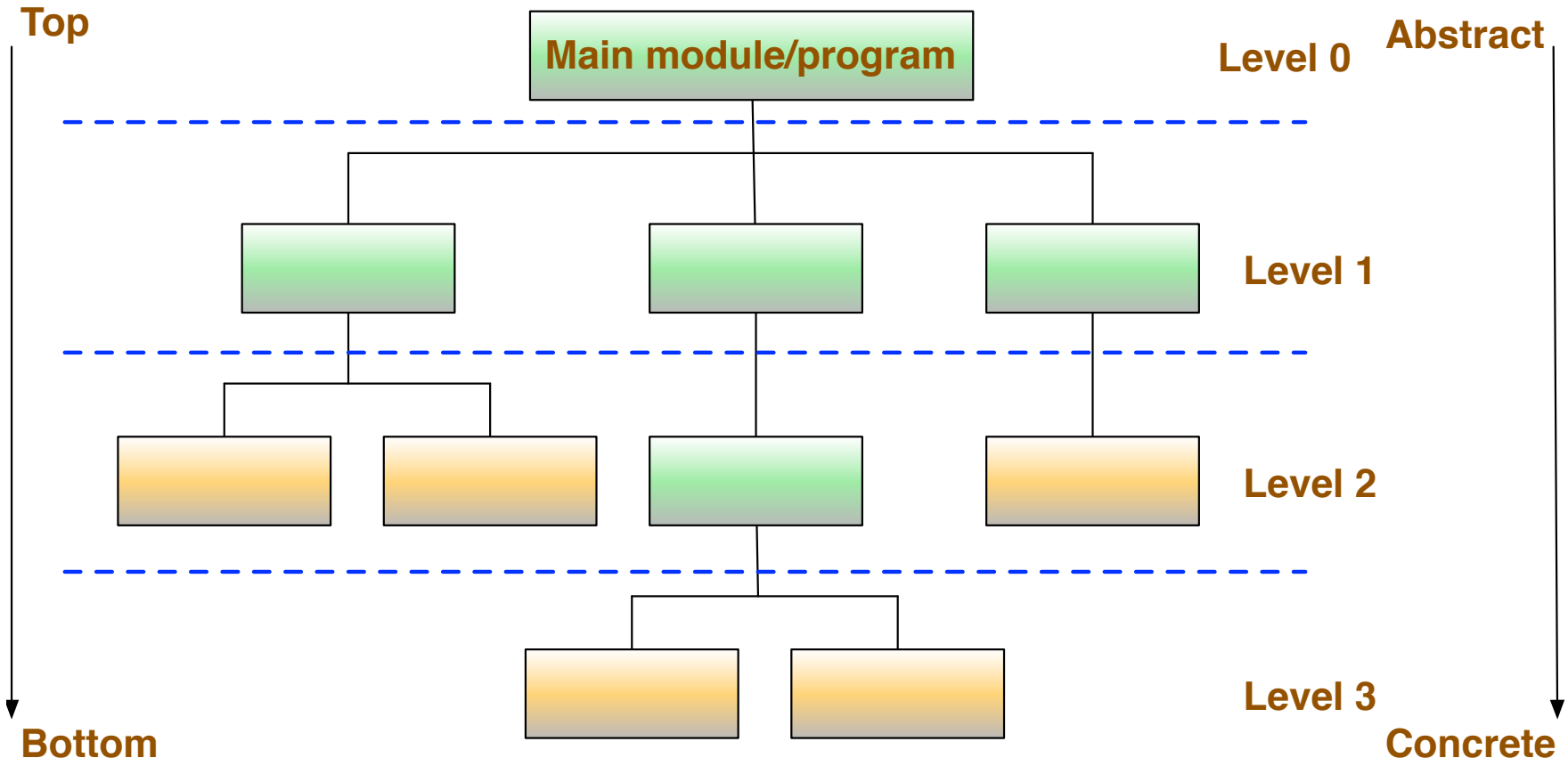
Top-Down Design

- Consists of breaking the problem into a set of sub-problems called **modules**
- It creates a hierarchical structure of problems and sub-problems (modules)
- This process continues for as many levels as it takes to expand every task to the smallest details.
- A step that needs to be expanded is called an **abstract step**

Top-Down Design

- Consists of breaking the problem into a set of sub-problems called **modules**
- It creates a hierarchical structure of problems and sub-problems (modules)
- This process continues for as many levels as it takes to expand every task to the smallest details.
- A step that needs to be expanded is called an **abstract step**
- A step that doesn't need expansion is a **concrete step**

Top-Down Design



Object-Oriented Design

- A **class** defines the abstract characteristics of a thing (**object**), including it's properties and the things it can do (**methods**)

Object-Oriented Design

- A **class** defines the abstract characteristics of a thing (**object**), including it's properties and the things it can do (**methods**)
  - » **One can say that a class is a *blueprint* that describes the nature of something**

Class, Object Example

- You can have a class **DOG** that defines all possible dogs by listing the characteristics and behaviours they have

Class, Object Example

- You can have a **class** **DOG** that defines all possible dogs by listing the characteristics and behaviours they have
  - » **The object** **Rover** **is one particular dog, with particular versions of the characteristics**
    - > **A Dog has fur; Rover has fur**

Class, Object Example

- Notice that fur is also an **object**

Class, Object Example

- Notice that fur is also an **object**
  - » **It has characteristics of its own**

Class, Object Example

- Notice that fur is also an **object**
  - » **It has characteristics of its own**
    - > **Rover's is long, brown and white**

Class, Object Example

- Notice that fur is also an **object**
  - » **It has characteristics of its own**
    - > **Rover's is long, brown and white**
- But fur is also a **class**!
  - » **A blueprint for anything that can be called fur**

Class, Object Example

- Notice that fur is also an **object**
  - » **It has characteristics of its own**
    - > **Rover's is long, brown and white**
- But fur is also a **class**!
  - » **A blueprint for anything that can be called fur**

OOD features

» **Information hiding**

OOD features

- Information hiding
- Data abstraction

OOD features

- Information hiding
- Data abstraction
- Encapsulation

OOD features

- Information hiding
- Data abstraction
- Encapsulation
- Modularity

OOD features

- Information hiding
- Data abstraction
- Encapsulation
- Modularity
- Polymorphism

OOD features

- Information hiding
- Data abstraction
- Encapsulation
- Modularity
- Polymorphism
- **Inheritance**

OOD features

- Information hiding
- Data abstraction
- Encapsulation
- Modularity
- Polymorphism
- Inheritance

We will see examples of these in the exercises

Testing the algorithm

- Regardless of the design methodology, algorithms must be tested

Testing the algorithm

- Regardless of the design methodology, algorithms must be tested
  - » **Testing at the algorithm development stage involves looking at each part of the design independently, and at how they are coupled**

Desk checking

Working through a design at a desk with a pencil and paper.

Walk-through

Manual simulation of the design, taking sample data values and tracing them through its steps to see what actually happens.

Testing the algorithm

- Large scale programming is conducted in teams

Testing the algorithm

- Large scale programming is conducted in teams
  - » **Divide-and conquer strategies make it possible to assign subtasks to team members, who independently develop solutions**

Testing the algorithm

- Large scale programming is conducted in teams
  - » **Divide-and conquer strategies make it possible to assign subtasks to team members, who independently develop solutions**
- In this environment testing is often done in team meetings using the **inspection** method

Design Inspection

On person (not the designer) reads the design (handed out in advance) line by line while the others point out problems.