ADVANCED FUNCTIONAL THINKING

DR SURBHI SARASWAT
SCHOOL OF COMPUTER SCIENCE

PROGRAMMING

- Key objectives of a Good Code:
  - Testability: Make it easy to ensure the software is working correctly
  - Maintainability: Make it easy to keep the software working (debugging, readability,)
  - Extendibility: Make it easy to add new functionality
  - Flexibility: Make it easy to adapt to new requirements
  - Reusability: Make it easy to reuse code in other projects

PROGRAMMING

- Modularity
  - Split up code into parts, e.g. functions, classes, modules, packages, …
    - You have done well if the parts are independent of each other and have clear responsibilities
    - You have done badly if the parts are very dependent on each other (changes in one part require changes in many others)
  - This has benefits for almost all of your goals:
    - Easier and more complete testability by using unit tests, better debugging
    - Confidence from unit tests allows for better maintainability and flexibility
    - Allowing to split responsibilities for different “modules” enhances collaboration and thereby maintainability Code reusability

PROGRAMMING

- Modularity
  - Split up code into parts, e.g. functions, classes, modules, packages, …
    - You have done well if the parts are independent of each other and have clear responsibilities
    - You have done badly if the parts are very dependent on each other (changes in one part require changes in many others)
  - This has benefits for almost all of your goals:
    - Easier and more complete testability by using unit tests, better debugging
    - Confidence from unit tests allows for better maintainability and flexibility
    - Allowing to split responsibilities for different “modules” enhances collaboration and thereby maintainability Code reusability

PROGRAMMING PARADIGMS

- A paradigm is a preferred approach to programming that a language supports.
- A programming style or way programming/thinking
  - Imperative.
  - Structured
  - Procedural
  - Object-Oriented
  - Functional

PROGRAMMING PARADIGMS

- Imperative Programming
  - Oldest approach.

It is one of the oldest approach.
It is closest to actual mechanical behavior of a computer.
A program is a list of instructions that change the memory state until the desired end state is achieved.
It is difficult to scale.
It is quite easy for simple programs.

  - Closest to the actual mechanical behaviour of a computer
  - Original imperative languages were abstractions of assembly language.
  - A program is a list of instructions that change a memory state until desired end state is achieved.
  - Useful for quite simple programs. Difficult to scale.
  - Soon it led to spaghetti code.

PROGRAMMING PARADIGMS

- Structured Programming
  - block structures, loops, and tests.
  - This is the paradigm we are likely to be most familiar with.
- Procedural Programming
  - Evolution of structured programming.
  - Divide the code into procedures: routines, subroutines, modules methods, or functions.
  - Advantages:
    - Division of work.
    - Debugging and testing.
    - Maintenance.
    - Reusability.

~Structured programming is solving a problem is approached by breaking it down into smaller sub-problems in a hierarchical manner.
~Breaking down a program into smaller, manageable modules or functions to enhance readability and maintainability.
~Code structures avoid multiple entry and exit points to improve clarity and ease of understanding.
~it is Machine-Independent, So it takes time to convert into machine code.
~Avoidance of unstructured control flow, especially the use of the GOTO statement.

~Procedural programming focuses on procedures or routines, which consist of a set of instructions to be executed.
~Encapsulation of data and procedures to create modular and reusable code.
~Data is typically scoped to the procedure, limiting its visibility and accessibility outside of the procedure.
~Execution follows a step-by-step process, and the emphasis is on the sequence of operations.

PROGRAMMING PARADIGMS

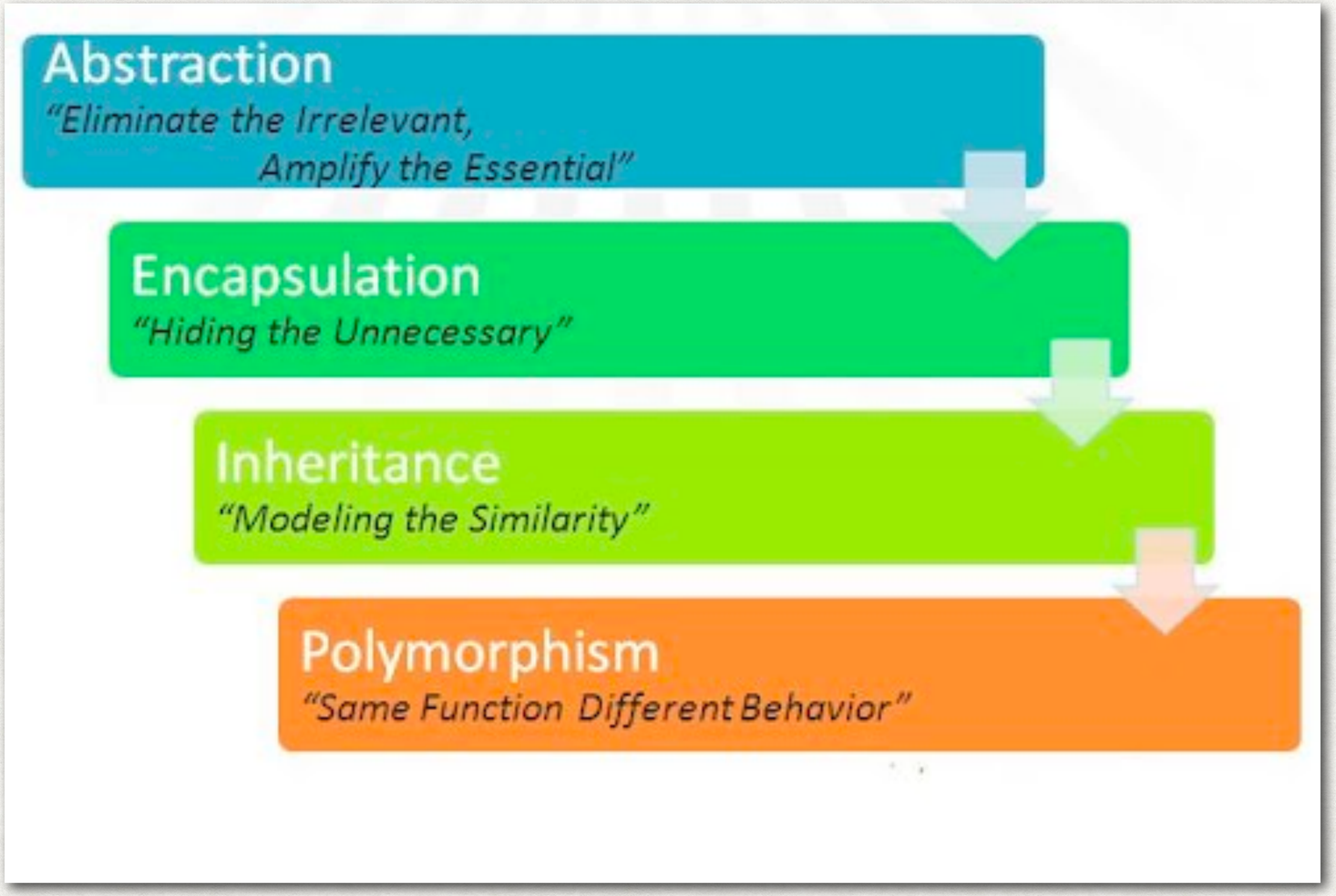
- Object Oriented Paradigm
  - Based on the concept of "objects", which can contain data and code.
  - The data is in the form of fields, and the code is in the form of procedures.
  - A common feature of objects is that procedures are attached to them and can access and modify the object's data fields.
  - Think in terms of objects that contain data and offer methods (functions that operate on objects)
  - Data and functions form a unit
  - Focus on object structure rather than manipulation logic Organize your code in classes (blueprints for objects)
  - Every object is instance of its class

PROGRAMMING PARADIGMS

- Object Oriented Paradigm work more efficiently than traditional techniques due to the following reasons.
    - The higher level of abstraction: Top-down approach support abstraction at the Functional level while object oriented approach support abstraction at the object level.
    - The seamless transition among different software development phases: It uses the same language for all phases, which reduces the level of complexity and redundancy makes software development clear and robust.
    - Good programming practice: The subroutine and attributes of a class are held together tightly.
    - Improves reusability: it supports inheritance due to which classes can be built from each other. So only difference and enhancement between classes need to be designed and coded. All the previous functionality remains as it is and can be used without change.

PROGRAMMING PARADIGMS

- Object Oriented Paradigm



FUNCTIONAL PROGRAMMING

- Functional Programming Paradigm
- FP is a programming paradigm and not a programming technique.
- FP is the philosophy of programming which dictates that the building blocks of code should be “pure functions”.
- The Idea is to model computational problem in terms of mathematical formulation.
- This usually means writing codes with functions.
- We know spreadsheets are good for working with functions.
    - You may have used a spreadsheet while working.
    - In a spreadsheet, there is a bunch of equations and we enter values in the given cells for these equations.
    - We get the answers through these equations.
    - When you enter the same values again, you get the same answer.
    - There are no fallouts.

FUNCTIONAL PROGRAMMING

- **Functional Programming Paradigm**
- FP is a programming paradigm and not a programming technique.
- FP is the philosophy of programming which dictates that the building blocks of code should be “pure functions”.
- The idea is to model computational problems in terms of mathematical formulation.
- This usually means writing codes with functions.
- We know spreadsheets are good for working with functions.
    - You may have used a spreadsheet while working.
    - In a spreadsheet, there is a bunch of equations and we enter values in the given cells for these equations.
    - We get the answers through these equations.
    - When you enter the same values again, you get the same answer.
    - There are no fallouts.

FUNCTIONAL PROGRAMMING

- Functional Programming Paradigm
  - expresses its computations in the style of mathematical functions
  - emphasizes expressions
    - (“is” something: a series of identifiers, literals and operators that reduces to a value)
  - over statements
    - (“does” something, e.g. stores value, etc.) —→ declarative nature
  - Data is immutable (instead of changing properties, I need to create copies with the changed property)
  - Avoids side effects (expressions should not change or depend on any external state)