Software design and Analysis

LECTURE 03

Outline

- Purpose of software
- Complexity of software
- Reason for software complexity
- Resolving Complexity

Purpose of software

- What is the purpose of software?
  - Perform complex tasks/calculations
  - Ease complexity
  - Reduce human intervention
  - Reduce errors
  - Perform monotonous work
  - …

Is developing software is complex?

Software Complexity

- Many objects in this world exhibit great complexity
  - Heart beating, photosynthesis weather etc.
- Applies to software as well
  - Database system
  - Financial system
  - Air traffic controller
- This complexity is too great for one person to understand
- Complexity can never be eliminated; but it can be reduced

Reasons for software Complexity

- Problem domain complexity
  - Some domains are complex e.g. finance, telecom etc.
  - Lends itself to software
  - Non-functional requirements, such as usability, performance, cost, etc. add to overall complexity

Reasons for software Complexity

- Communication between users and developers
  - May have vague idea of what they expect from software
  - Difficult for user to express their requirements
  - Developers expect requirement in a specific format (e.g. UML)
  - Both user and developers may lack domain experience
  - Users get a better idea of the system only after seeing prototypes or design documents
  - May request a change that is difficult to incorporate

Reasons for software Complexity

- Volatile nature of requirements
  - Requirements change
  - May be difficult, if not impossible to incorporate that change
  - Future changes must be anticipated

Reasons for software Complexity

- Unpredictable behavior of software
  - Software runs of systems with discrete components
  - Application may have multiple threads, variables, memory allocations
  - Large numbers of events and states
  - Cause combinatorial explosion
  - Interaction between components

Reasons for software Complexity

- Developmental process
  - Big teams
  - Geographical locations
  - Communication/coordination between developers
  - Resource shortage

Summary

- Software solves problem & reduces complexity
- Writing software is complex
- Factors such as domain, requirements, etc., contribute to complexity of the software

Resolving Complexity

- Human beings is built-in mechanism for dealing with complexity
- Decompose each part into smaller parts till each part makes sense individually
- Finally, integrate the parts to build the final system
- “divide et impera” – divide and rule
- Ancient rule to conquer the state or component and Same principle is applied to programming languages

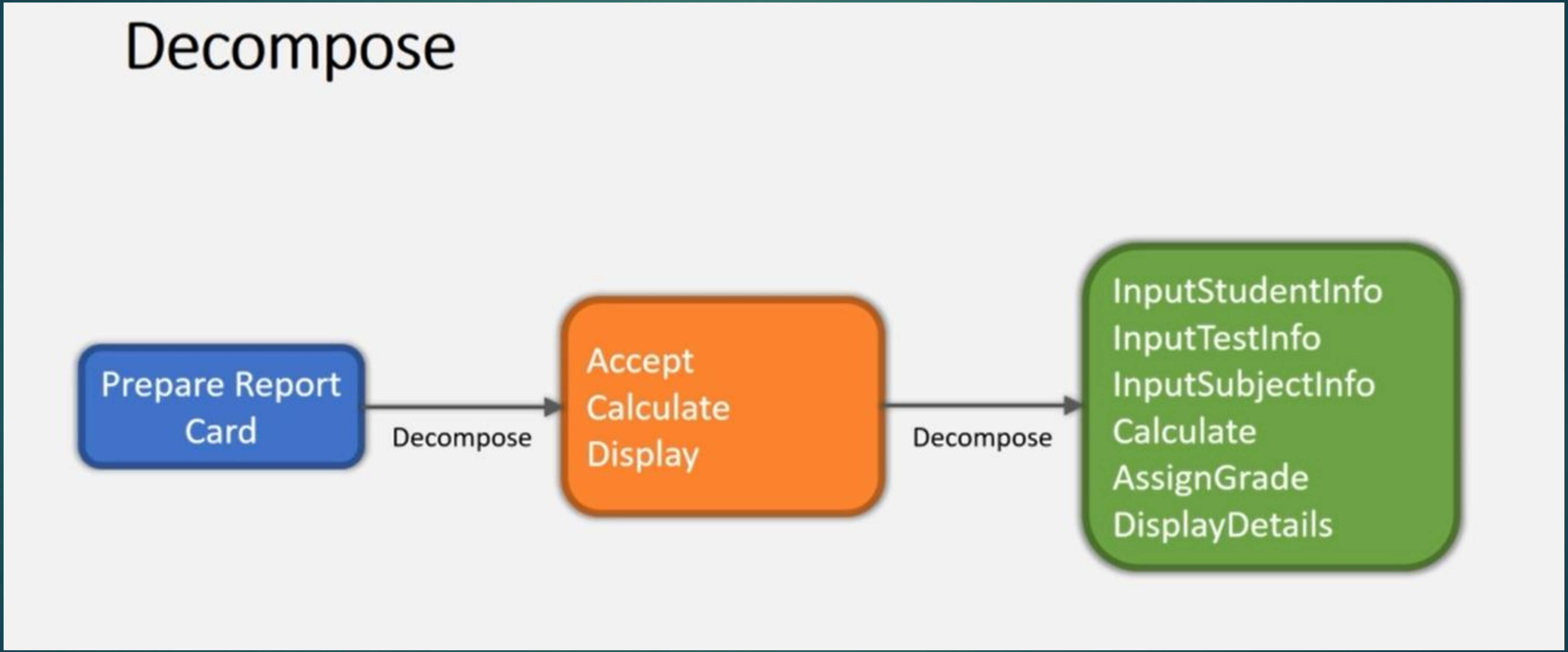
Algorithmic Decomposition

- Decompose problems into algorithms
- Each algorithm is part of series of steps (part of overall process)
- Typically a top-down approach
  - Start with a big picture
  - Break down into smaller chunks
- These lead to final result(like a flow chart)

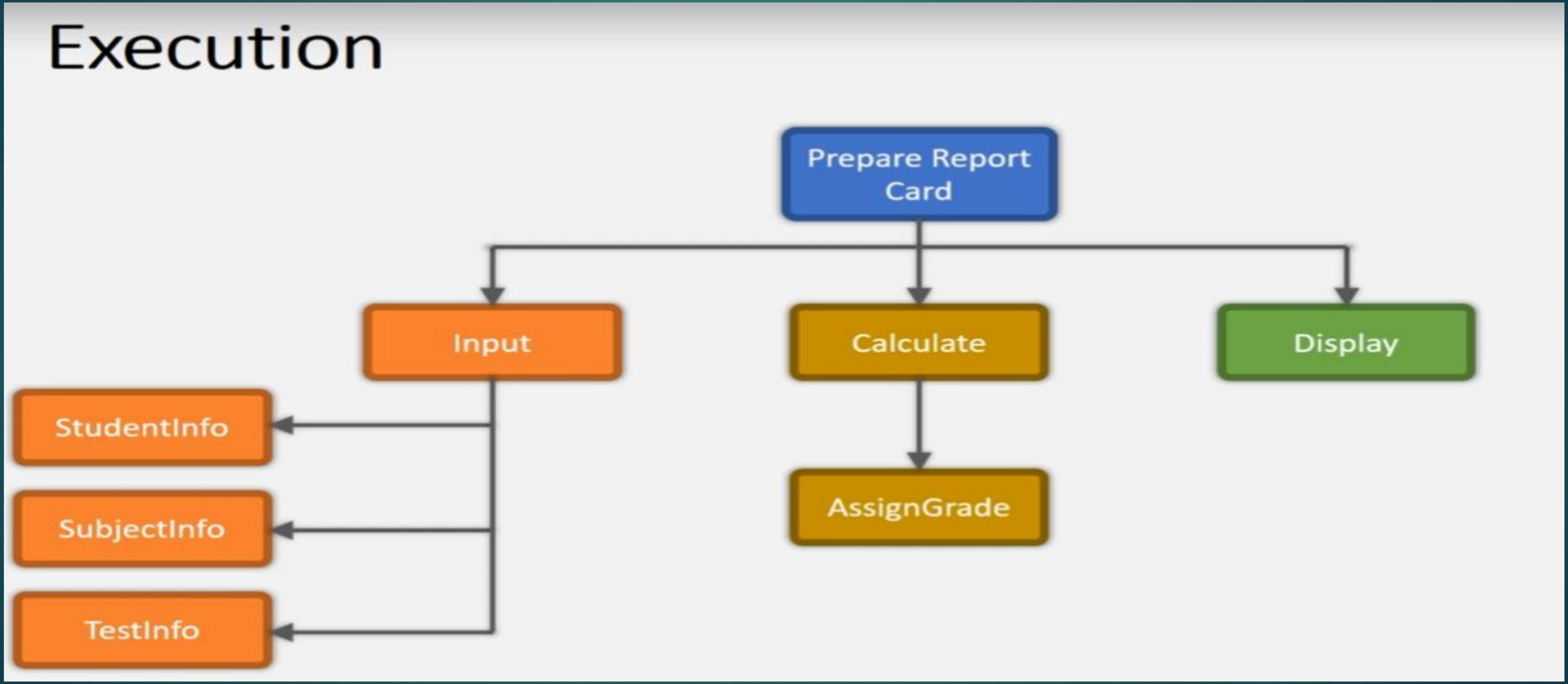
Example

- Write a software to help tutor prepare a report card for their students
- The software will
  - Accept students details (name, roll number , class)
  - Accept test details (semester/unit)
  - Accept subject name and score
  - Calculate total score
  - Calculate average score
  - Assign grade
  - Display lowest & highest scoring subjects (based on the threshold)

Decompose


Decompose
Prepare Report Card
Decompose
Accept
Calculate
Display
Decompose
InputStudentInfo
InputTestInfo
InputSubjectInfo
Calculate
AssignGrade
DisplayDetails

Execution


Execution
Prepare Report Card
Input
Calculate
Display
StudentInfo
SubjectInfo
TestInfo
AssignGrade

Disadvantages

- Development based on the high-level specification
- Algorithms are very specific to the application, thus, difficult to reuse
  - Often changes over time
  - Parts have to be rewritten
- Difficult to add new features
  - Algorithm are tightly wired to work together

Disadvantages

- Data is treated with low significance
  - Data is shared between algorithms
  - Unintentional modification can lead to disastrous results
- Focus on operations
  - No idea about the entity on which operation is performed
  - Overall understanding of the application becomes complicated
  - Doesn't map to real life entities

Languages

- Languages based on algorithmic decomposition
  - Fortran, Cobol, Pascal, c, etc.
- These are the initial programming languages that were created to solve scientific problem and performing mathematical operations.
- The system based on these languages are
  - Driven in size
  - Complex
  - Do not at risk with problem associated with reusability, scalability and maintainability

New way to resolve complexity

Object Oriented Decomposition

Summary

- Algorithmic decomposition
- Focus on operations rather than data
- Dosent scale well
- Doesn’t map to real world