EE 441 Data Structures Lecture 1

Ilkay Ulusoy

Data Structures

- Computer memory is not an infinite source and cannot be accessed immediately.
- A data structure is a systematic way of organizing and accessing data in the memory so that it can be used efficiently in terms of size and time.
  - Examples: queue, stack, linked list, tree
- Most data structures have associated algorithms to perform operations that maintain the properties of the data structure.
  - Examples: search, insert, delete,

- A well-designed data structure allows a variety of critical operations to be performed on using as little resources as possible.
- Sources:
  - execution time
  - memory space

Procedural Programming

- Programs are divided into pieces which can be combined later.
- These pieces are written by programmers.
- Other users construct their own programs using these pieces.

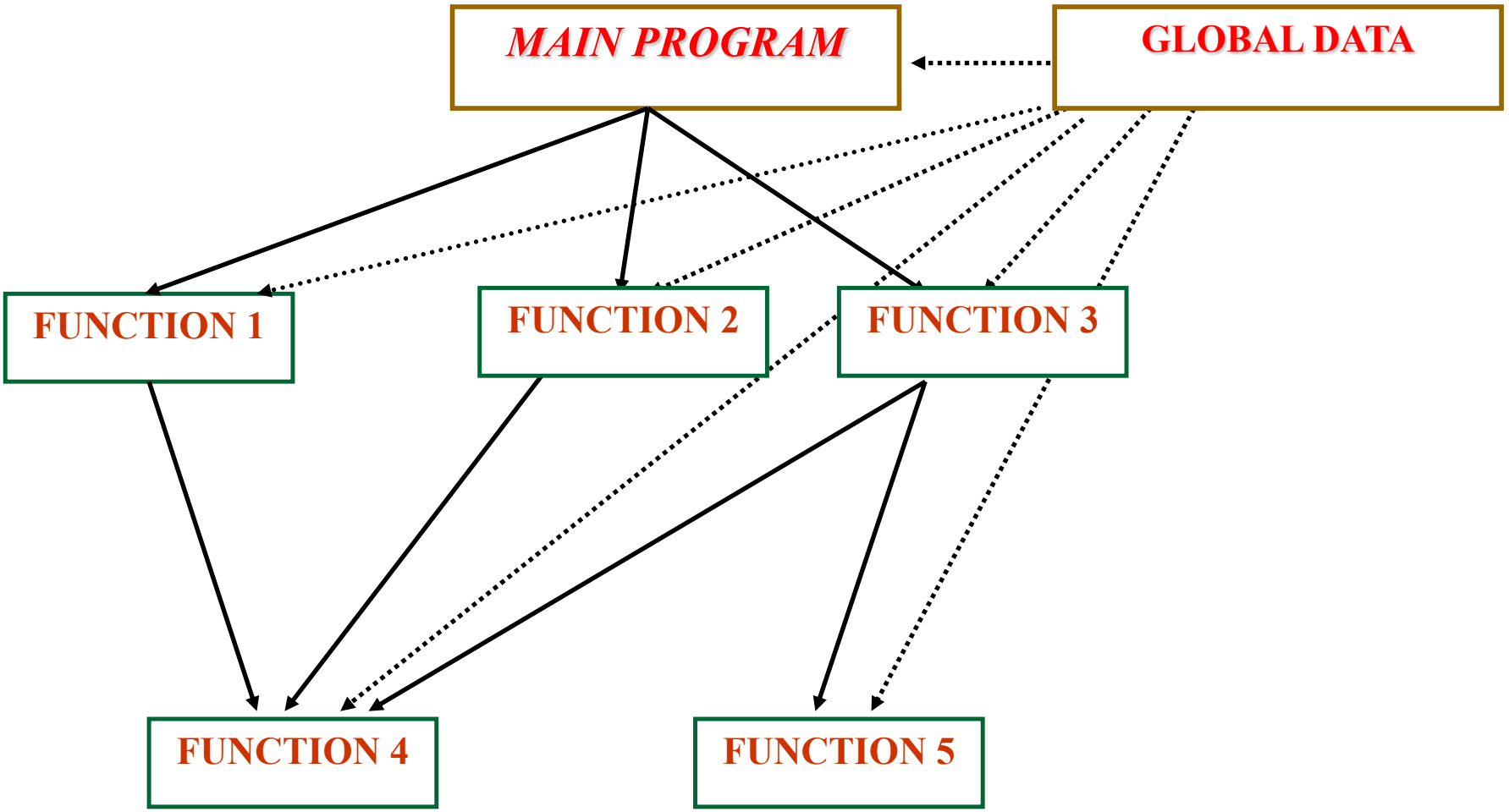
- Abstraction:
  - separates what the user needs to know and the programmer needs to know

- users can think in high-level terms
- users don't need low-level details about the piece implementations

Different ways of programming:
1. Structured Programming (SP)
2. Object-oriented Programming (OOP)

- Structured Programming: procedural abstractions
  - Using function
  - Function & program is divided into modules
  - Every module has its own data and function, which can be called by other modules
  - Focus on arguments and return values

STRUCTURED PROGRAMMING:



Object-oriented Programming (OOP) :

- procedural abstractions,
- data abstractions,
- encapsulation.

- Ignore the way data is represented in memory
- Focus on operations that can be performed on data
- Encapsulation (information hiding) aids the software designer
- The implementation details are hidden from the user of that object

Object oriented

- Produce highly maintainable software,
  - where changing one part of a program would not break the rest!
- Produce reusable software,
  - where similar programming tasks could share common elements, without having to reinvent the wheel!
- Make complex programming problems easier,
  - Where debugging is simpler!

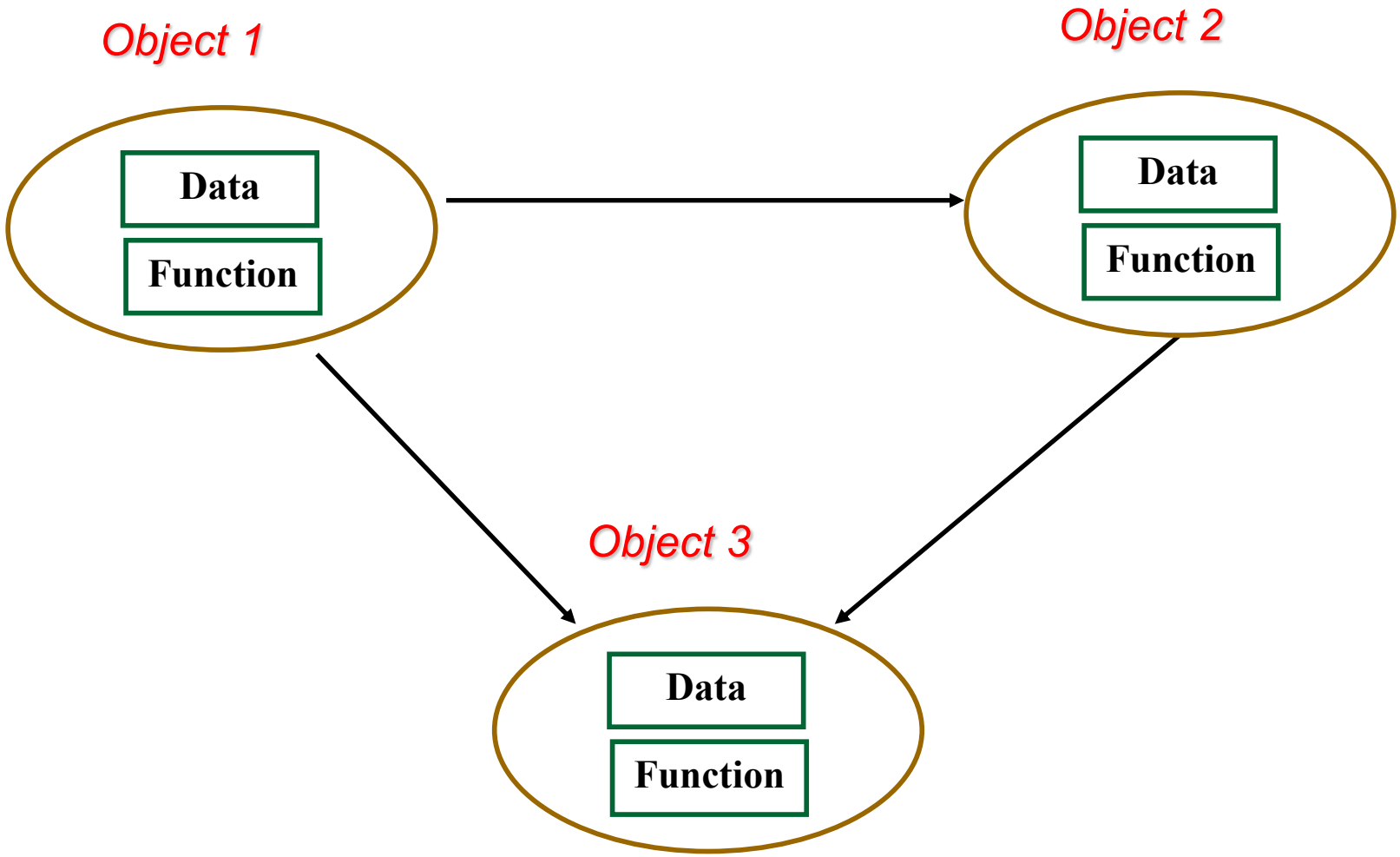
OO Approach

- OO development starts by thinking about the *problem*, and not about the *program* that will implement the solution
  - In the real world, "things" have *characteristics* and *behavior*.
    - e.g. a car is a "thing" that has a make, model, registration number and can accelerate, steer, brake, change gear etc.
  - We think about the objects involved in the problem, their *characteristics* (*attributes*) and *behaviors*

An OO language allows us to

model these attributes and behaviors, and

construct a solution to the problem from these objects



Some OOP Terminology

- **object** - usually a person, place or thing (a noun)
- **method** - an action performed by an object (a verb)
- **class** - a category of similar objects (such as automobiles)
  - Objects of the same class have the same data elements and methods

Examples of Objects

- What the objects are, will be determined by the problem domain:
  - In a banking application:
    - customers, accounts, sums of money etc.
  - In a library application:
    - books, journals, CD-Roms, members, etc.
  - In a communication network simulator:
    - network nodes, queues, channels, links, packets etc.
  - In computer vision:
    - Moving blobs in a frame of a video

OO Programming

- Objects send and receive messages to invoke actions
- The real world can be accurately described as a collection of objects that interact with each other
- Pure OO Languages: Smalltalk, Eiffel, Actor, Java
- Hybrid OO Languages: C++, Object-Pascal

Design Principles of OOP

- Encapsulation
- Polymorphism
- Inheritance

Encapsulation - data hiding

- software can be easily used without knowing the details of how it works.
  - Only object's function which perform parts of the objects behavior can modify information in the object.

- An analogy:
  - When you drive a car, you don't have to know the details of how many cylinders the engine has or how the gasoline and air are mixed and ignited.
  - Instead you only have to know how to use the controls.

Polymorphism

- the same word or phrase can mean different things in different contexts

- Analogy:
  - in English, bank can mean side of a river or a place to put money

- Function Overloading:
  - The operation of one function depends on the argument passed to it.
  - Example: Fly(), Fly(low), Fly(150)

Inheritance

- a way of organizing classes
  - Classes with properties in common can be grouped so that their common properties are only defined once.
  - Term comes from inheritance of traits like eye color, hair color, and so on.
- ***Superclass:*** inherit its attributes & methods to the subclass(es).
- ***Subclass:*** inherit all its superclass attributes & methods besides having its own unique attributes & methods.

An Inheritance Hierarchy