15CSE374 INTRODUCTION TO DATA STRUCTURES AND ALGORITHMS

Sarath tv

Last Lecture

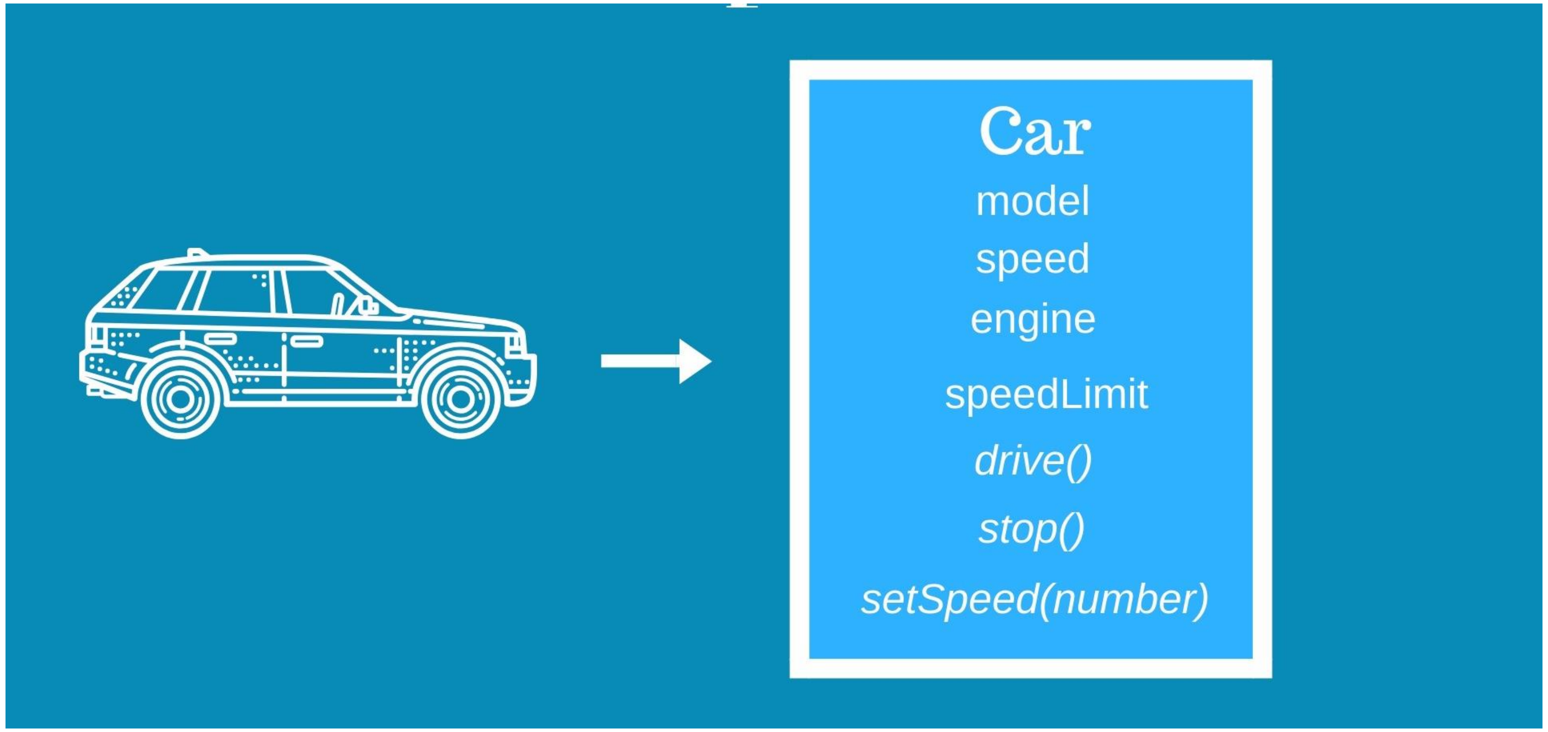
- Overview of Data structures.
- Need for Data structures.
- Cost and Benefits.

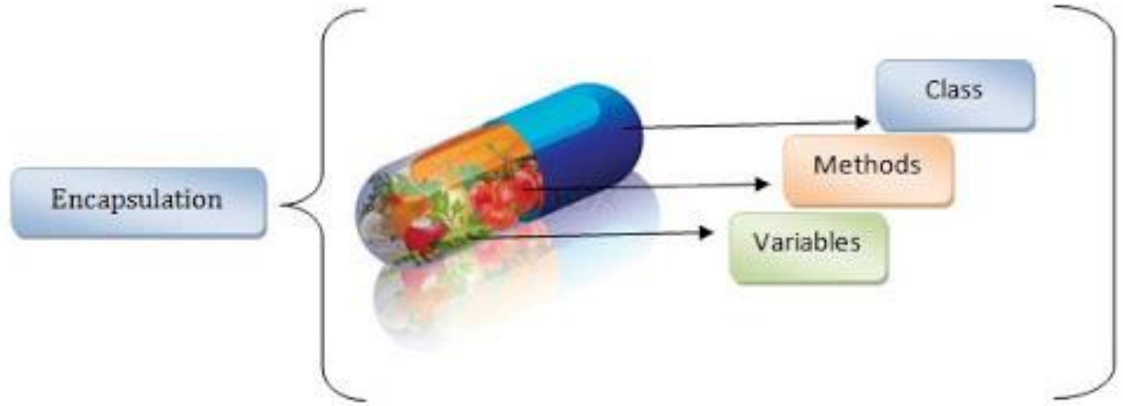
ADT & DS

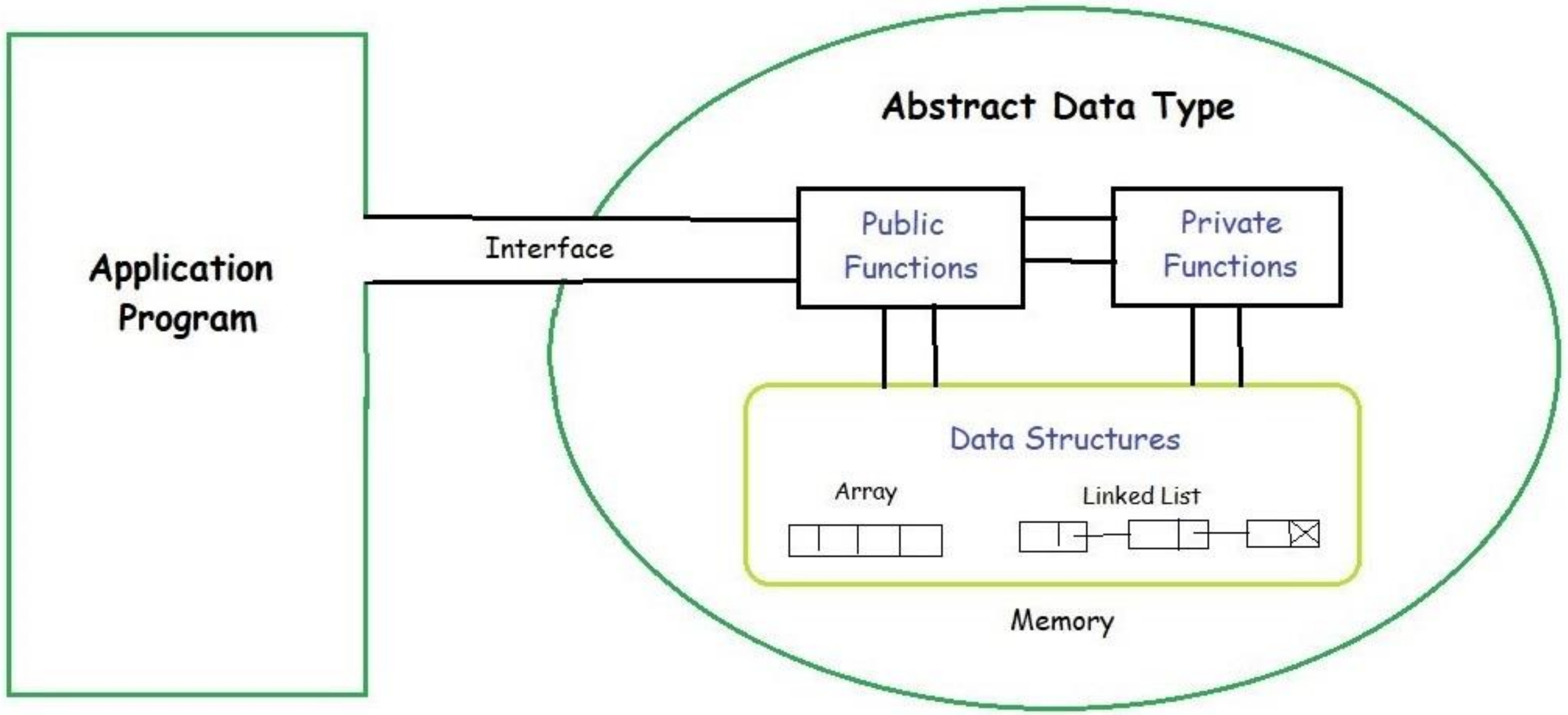
- Type- collection of values.
- Boolean, Integers-Simple types(doesn’t contain subparts).
- Aggregate/Composite type-Contains several information.
- A part of this aggregate type(a value) -Data item
- Data item → Member of type.
- Data type = type +operations
- Integer variable- member of integer data type.
- Supporting arithmetic operations(+,- *,/).

Abstract Data Type

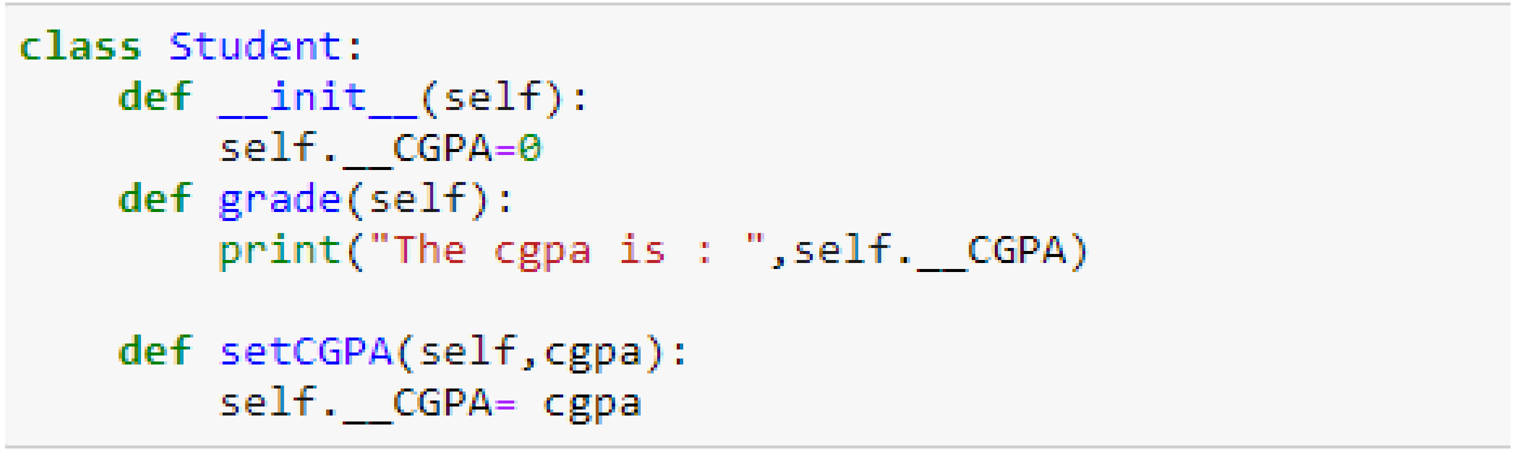
- Realization of a data type.
- To interact with the ADT- Interfaces- functions- Inputs and Outputs
- No specification on how the data type is implemented.
- Hidden from the user and protected from outside- Encapsulation in OOP.
- DS- Implementation for an ADT.
- In Object Oriented Language – Class are utilized.
- Operation associated → member function or methods.
- Data items → Data members in class


Car
model
speed
engine
speedLimit
drive()
stop()
setSpeed(number)


Encapsulation
Class
Methods
Variables


Application Program
Interface
Abstract Data Type
Public Functions
Private Functions
Data Structures
Array
Linked List
Memory

Abstraction Python code snippet



```
class Student:
    def __init__(self):
        self.__CGPA=0
    def grade(self):
        print("The cgpa is : ",self.__CGPA)

    def setCGPA(self,cgpa):
        self.__CGPA= cgpa
```

ADT

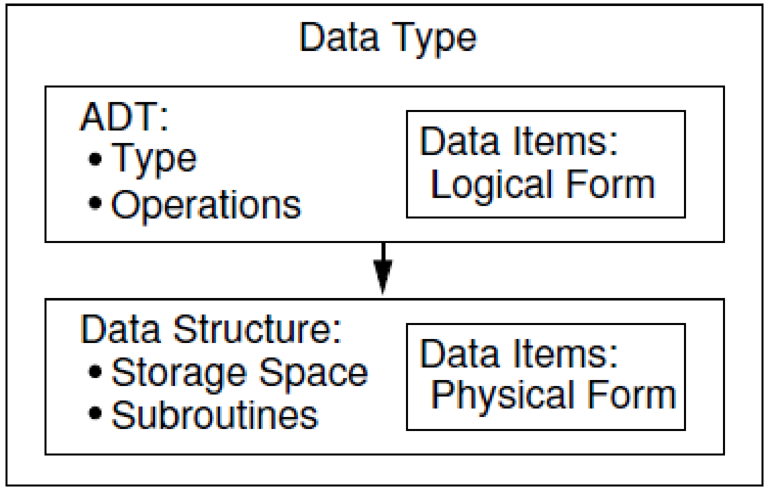
- Stores data
- Allows various operations on data to manipulate it.
- Specify the operation of the data structure and leave implementation for later.
- Interface doesn’t give any specific details about how something should be implemented or in what programming language.

Core Operations

- Interface for
  - Add
  - Remove
  - Find, retrieve or access an item
- More interfaces….
  - Empty or not?
  - Slice into subsets.

Logical & Physical Form.

- Definition of the data type wrt ADT- Logical Form.
- Implementation of data type as DS- Physical Form.



ADT	DS
List	Linked list ,Array
Queue	LL, Array
Map	Tree, Hash map
Vehicle	Cycle, car, bus

Design Patterns

- Flyweight
- Visitor
- Composite
- Strategy

Flyweight

- Software design pattern.
- Used to reduce the number of objects created and to decrease memory footprint and increase performance.
- Ways to decrease object count thus improving the object structure of application.
- Reuse already existing similar kind objects by storing them and creates new object when no matching object is found.
- A flyweight is an object that minimizes memory usage by sharing as much data as possible with other similar objects;
- Some parts of the object state can be shared,
- Hold them in external data structures and pass them to the objects temporarily when they are used.

Visitor

- Describe how to solve recurring design problems to design flexible and reusable object-oriented software,
- Objects that are easier to implement, change, test, and reuse.
- Represent an operation to be performed on the elements of an object structure.
- Visitor lets you define a new operation without changing the classes of the elements on which it operates.
- Way of separating an algorithm from an object structure on which it operates.
- The visitor allows adding new virtual functions to a family of classes, without modifying the classes.

Composite

- The **composite pattern** is a partitioning design pattern. The composite pattern describes a group of objects that are treated the same way as a single instance of the same type of object.
- The intent of a composite is to "compose" objects into tree structures to represent part-whole hierarchies.
- Implementing the composite pattern lets clients treat individual objects and compositions uniformly
- Used where we need to treat a group of objects in similar way as a single object.

Strategy

- In computer programming, the **strategy pattern** (also known as the **policy pattern**) is a software design pattern that enables selecting an algorithm at runtime.
- Instead of implementing a single algorithm directly, code receives run-time instructions as to which in a family of algorithms to use
- Strategy lets the algorithm vary independently from clients that use it.

Algorithm

- Problem- task to be performed.
- In terms of inputs and outputs.
- In mathematical sense → Problems as functions :: Matching between inputs and outputs.
- Algorithm :: Method or process to solve a problem.
- Implementation for function(problem).
- Different algos for a problem.
- An algorithm in a programming language → Program.

- Two questions
- How much time does the algorithm need to finish?
- How much space does this algorithm need for its computation?

Algorithm Analysis

- To estimate the resource consumption of an algorithm.
- Helps to compare two or more algorithms.
- Cost of algorithms for same problem.
- Growth rate, upper and lower bounds.
- How to compare 2 algos???
- Implement both and run!!!!!
- Unsatisfactory approach….
    - Effort
    - Better written.
    - Bias
    - Out of bound in terms of resource budget.

THANK YOU!!!!!