OBJECT ORIENTED PROGRAMMING

IT PCC-CS503 TH

TOPIC

ABSTRACT DATA TYPES

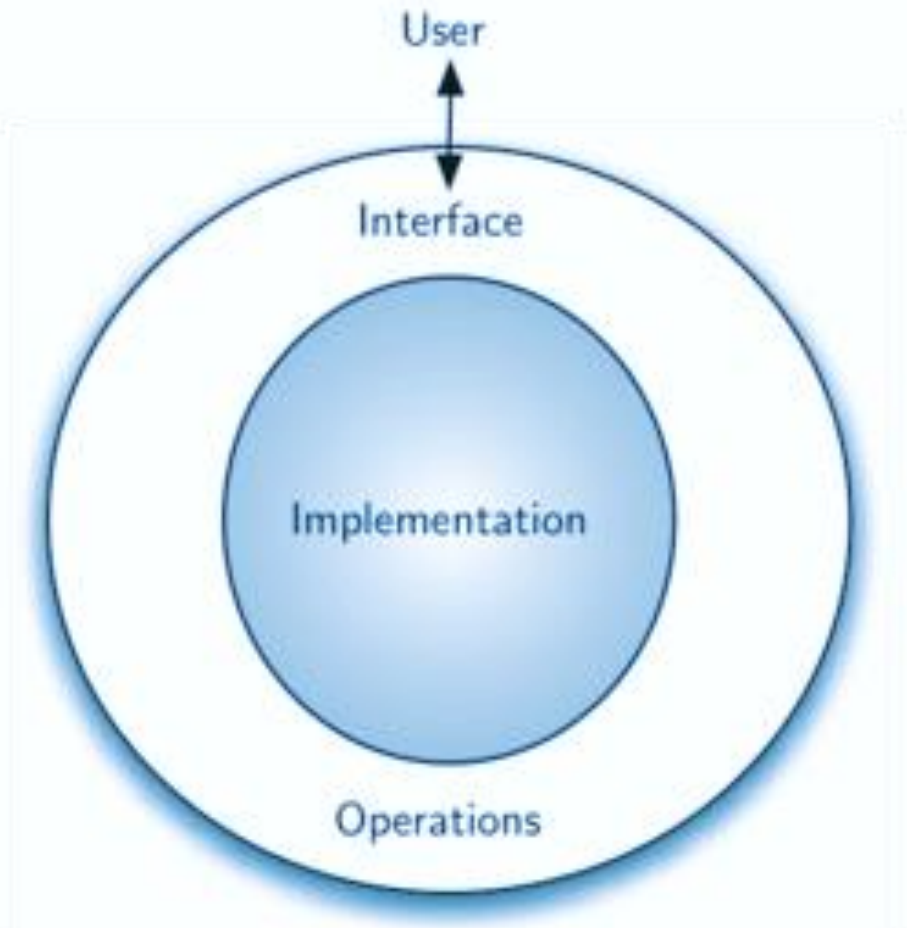
GROUP-8

Presented By:

- Pinaki Subhra Bhattacharya (11500219053)
- Sayon Islam (11500219054)
- Soham Nandi (11500219057)
- Randrita Sarkar (11500219058)

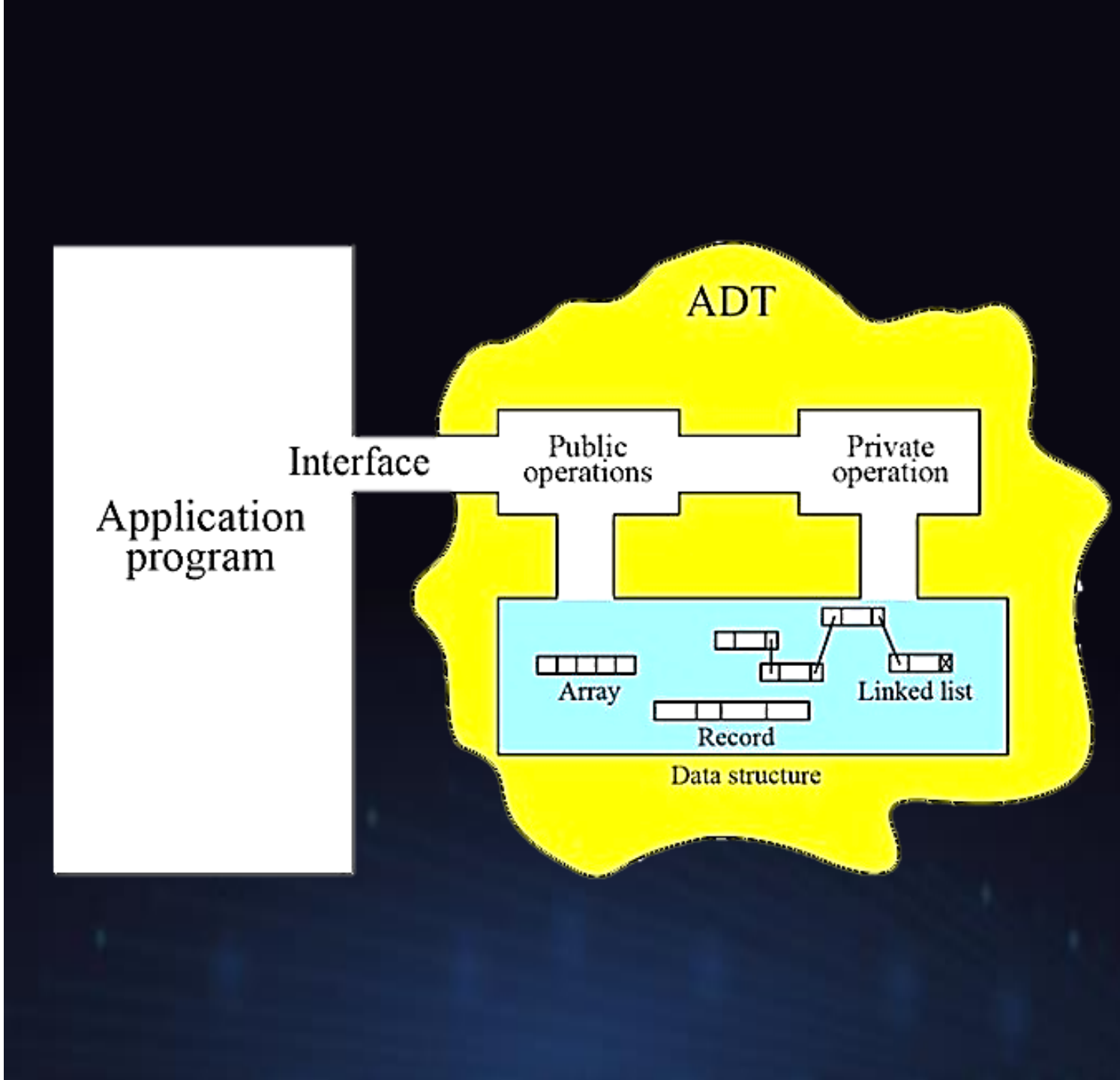
ABSTRACT DATA TYPE(ADT)

- To process data with a computer, we need to define the **data type and the operation** to be performed on the data.
- The definition of the data type and the definition of the operation to be applied to the data is part of the idea behind an **abstract data type** (ADT)
- ADT means to hide how the **operation is performed** on the data
- In other words, the user of an ADT needs only to know that a set of operations are available for the data type, but **does not need to know how they are applied**.



WORKFLOW

An **abstract data type(ADT)** is an abstraction of a data structure which provides only the interface to which a data structure must adhere to. The interface does not provide any specific details about how things are getting implemented. It will be more clear if we see some examples.

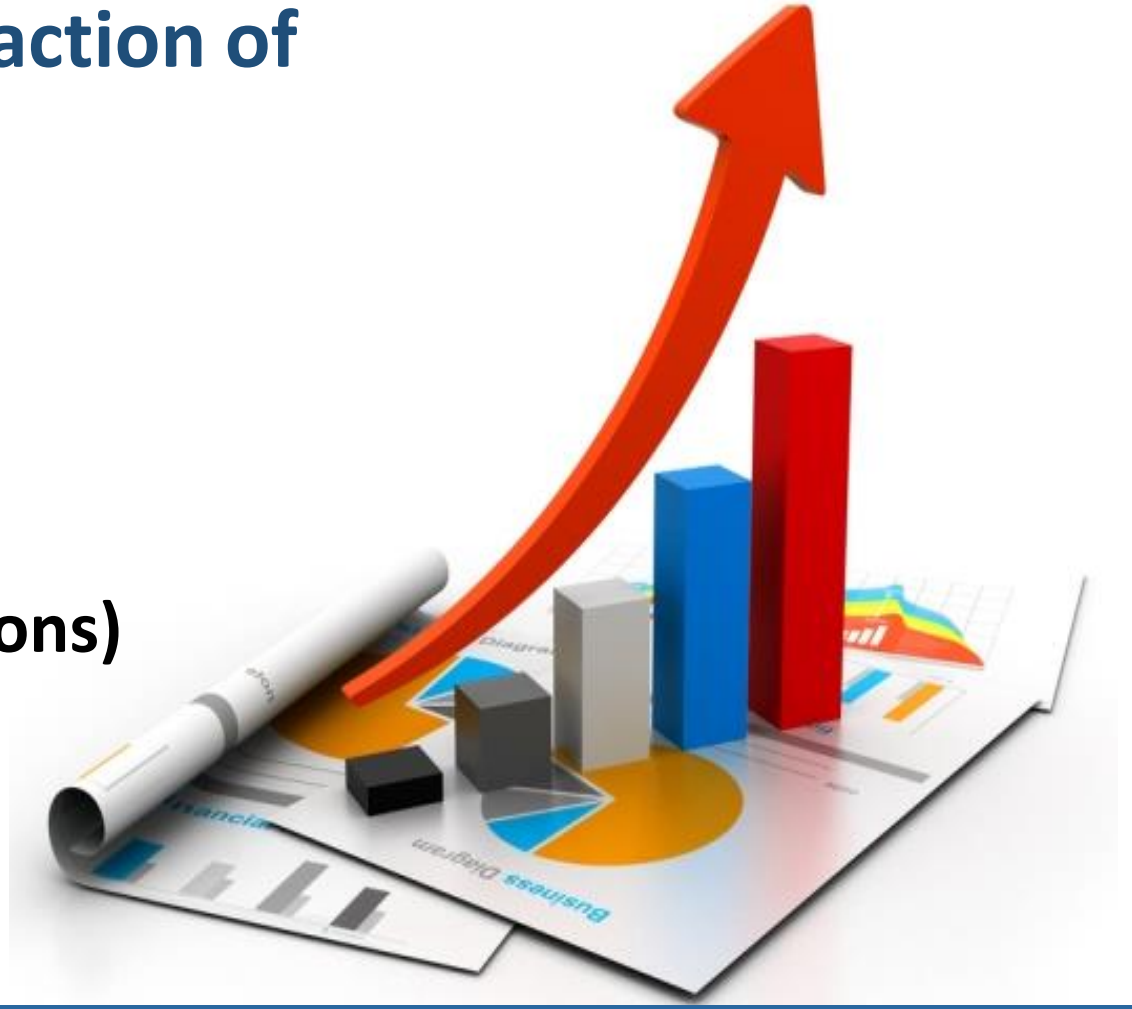


REAL LIFE EXAMPLE

ADT MODELING

A Simple Stock Trading System (Abstraction of Data Structure)

- **Operation:**
  - Order buy(Stock, Share, Price)
  - Order Sell(Stock, Share, Price)
  - Void cancel(order)
- **Error Condition(associated with operations)**
  - Buy/Sell a non existent stock
  - Cancel a non existent order



Abstract Data Type : Operations

- **Creators:** Creators create new objects of the type. It may take an object as an argument.
- **Producers:** Producers create new objects from old objects of the type. For example, the concat() method of the String is a producer that takes two strings and produces a new String representing their concatenation.
- **Observers:** Observers take the objects of the abstract type and return objects of a different type. For example, the size() method of the **List** returns an **int**.
- **Mutators:** Mutators change objects. For example, the add() method of List changes a list by adding an element to the end.

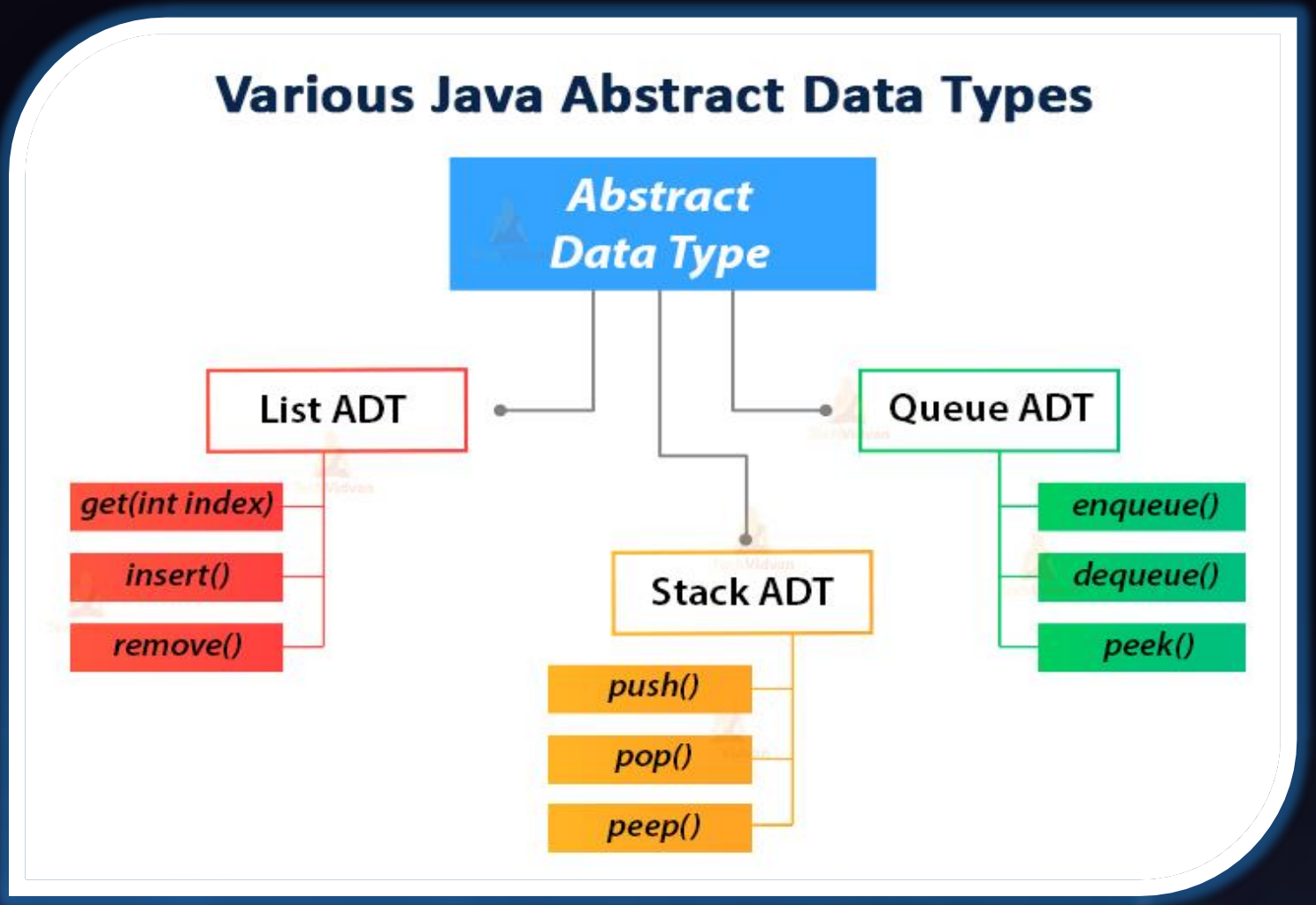
The **list** is an interface of Java List. The list is **mutable**. Its operations are:

creators: ArrayList and LinkedList constructors, Collections. singletonList

producers: Collections.unmodifiableList

observers: size, get

mutators: add, remove, addAll, Collections.sort

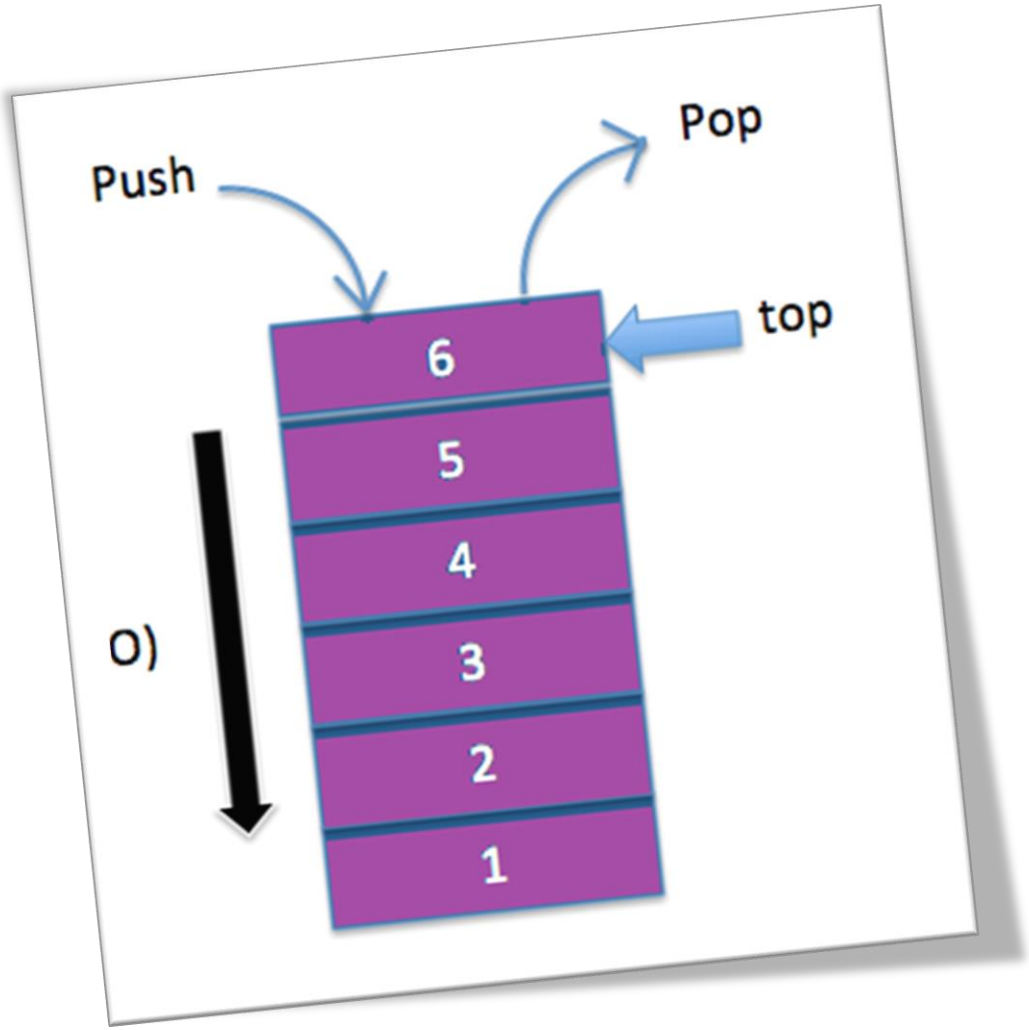


THE STACK ADT

Objects:

- A finite sequence of elements of the same type
- Additions restricted to one end of the sequence called the top of the Stack
- Deletions also restricted to the same end

- **Operations:**
  - Initialise
  - Push
  - Pop
  - Empty
  - Full



DESIGNING AN ABSTRACT DATA TYPE IN JAVA

Here are a few rules for designing an ADT.

- It’s better to combine simple and few operations in powerful ways, rather than a lot of complex operations.
- Each operation in an Abstract Data Type should have a clear purpose and should have a logical behavior rather than a range of special cases.
- The set of operations should be adequate so that there are enough kinds of computations that users likely want to do.
- The type may be either generic, for example, a graph, a list or a set, or it may be domain-specific for example, an employee database, a street map, a phone book, etc. But there should not be a combination of generic and domain-specific features.

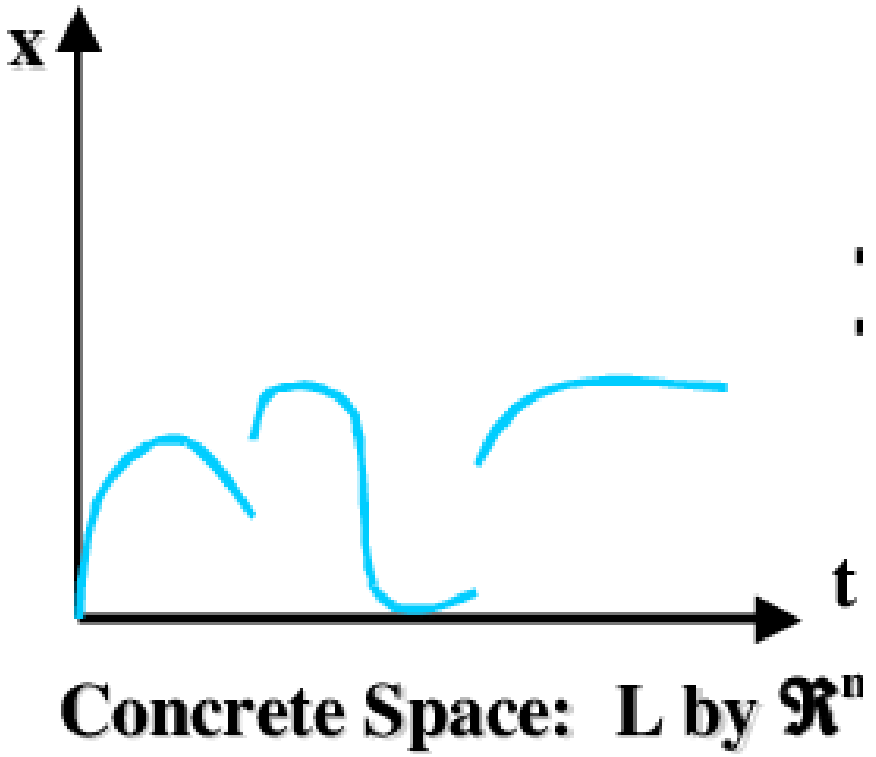
CONCRETE STATE SPACE

A set of all possible math model values of the data representation. In a commutative diagram, bottom half represents concrete state space.

CONCRETE INVARIANT

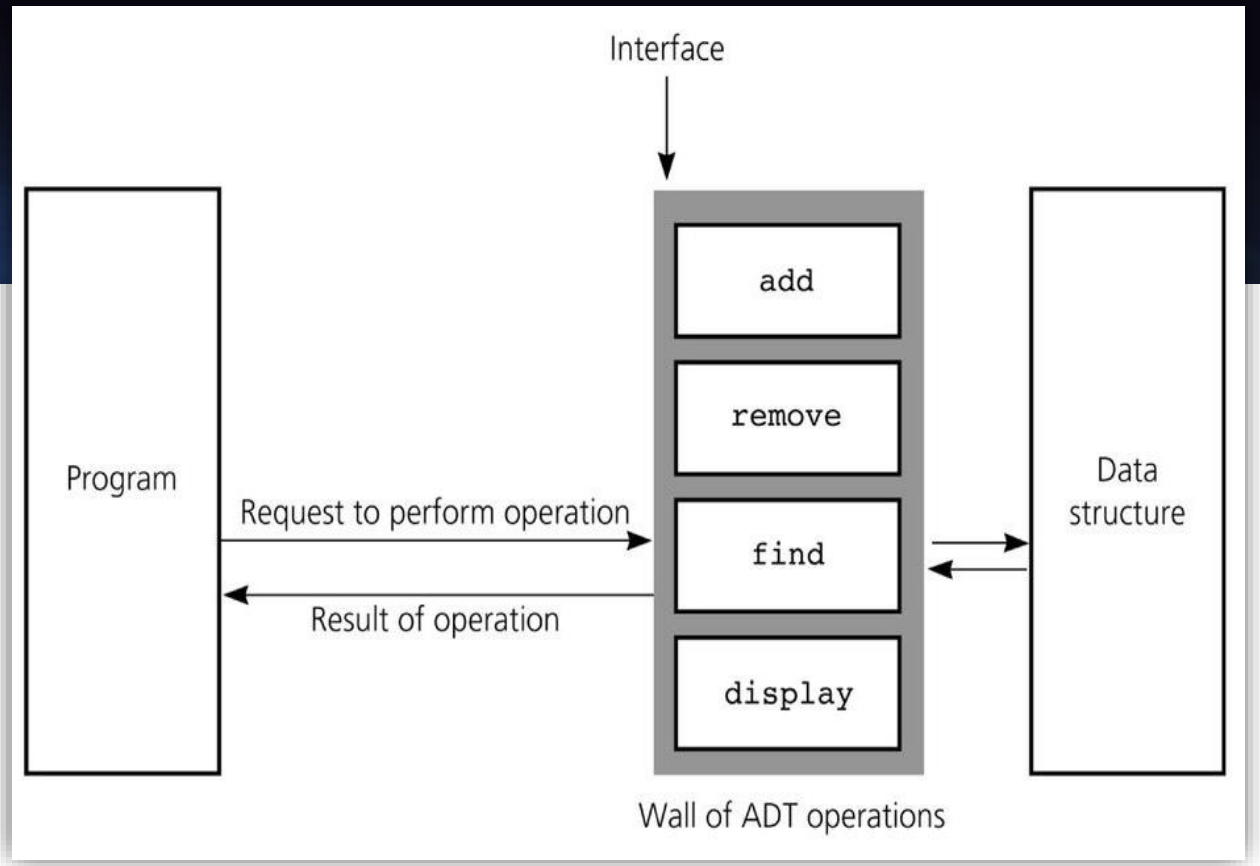
An invariant is a property that is always true of an ADT object instance, for the lifetime of the object.

- A good ADT preserves its own invariants. Invariants must be established by creators and producers, and preserved by observers and mutators.
- The rep invariant specifies legal values of the representation, and should be checked at runtime with checkRep().
- The abstraction function maps a concrete representation to the abstract value it represents.Representation exposure threatens both representation independence and invariant preservation.



WHY ABSTRACT DATA TYPES?

- User of the ADT **"sees"** only the interface to the objects; the implementation details are "hidden" in the definition of the ADT
- The user constrained to manipulate the object solely through the **functions (operations)** that are provided
- The designers may still alter the representation as long as the **new implementations** of the operations do not change the user interface. This means that users will not have to recode their algorithms.



BIBLIOGRAPHY

- https://techvidvan.com/tutorials/java-abstract-data-type/
- https://www.geeksforgeeks.org/abstract-data-types/
- https://stackoverflow.com/questions/10267084/what-is-adt-abstract-data-type

Thank You..!