



Introduction to ADT

CS212:Data Structure

Data Types & Data Structures

- ▶ Applications/programs read data, store data temporarily, process it and finally output results.
- ▶ What is data? Numbers, Characters, etc.



Data Types & Data Structures

- ▶ Data is classified into data types. e.g. char, float, int, etc.
- ▶ A data type is:
 - (i) a **domain** of allowed values and
 - (ii) a set of **operations** on these values.
- ▶ Compiler signals an error if wrong operation is performed on data of a certain type.
 - For example,
 - `char x, y, z;`
 - `z = x * y` is not allowed.



Data Types & Data Structures

► Examples

| Data Type | Domain | Operations |
|-----------|-----------------------|-----------------------------|
| boolean | true, false | &&, , =, etc. |
| char | ASCII | =, !=, <, etc. |
| integer | -maxint to +maxint | +, -, =, ==, !=, <, etc. |

Data Types & Data Structures

- ▶ `int i, j;` → `i, j` can take only integer values and only integer operations can be carried out on `i, j`.
- ▶ **Built-in types:** defined within the language e.g. `int, float, etc.`
- ▶ **User-defined types:** defined and implemented by the user e.g. using `typedef` or `class`



Data Types & Data Structures

- ▶ **Simple Data** types: also known as atomic data types
→ have no component parts. E.g. int, char, float, etc.

21

3.14

'a'



Data Types & Data Structures

- ▶ **Structured Data types:** can be broken into component parts. E.g. an object, array, set, file, etc. Example: a student object.

| | | | | | |
|--------|----|---|---|---|---|
| Name | A | H | M | A | D |
| Age | 20 | | | | |
| Branch | C | S | C | | |

A Component part

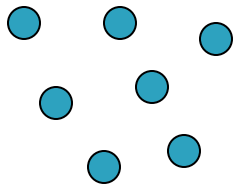
Data Types & Data Structures

- ▶ **A data structure** is a data type whose values
 - (i) can be decomposed into a set of component elements each of which is either simple (atomic) or another data structure
 - (ii) include a structure involving the component parts.



Data Types & Data Structure

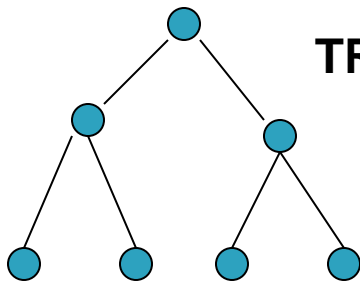
Possible Structures: Set, Linear, Tree, Graph.



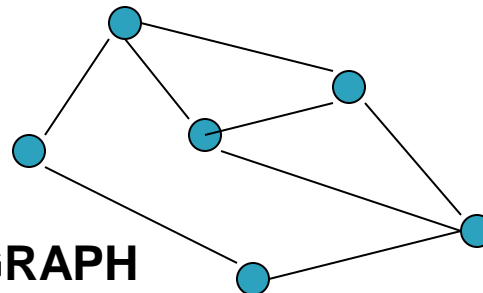
SET



LINEAR



TREE

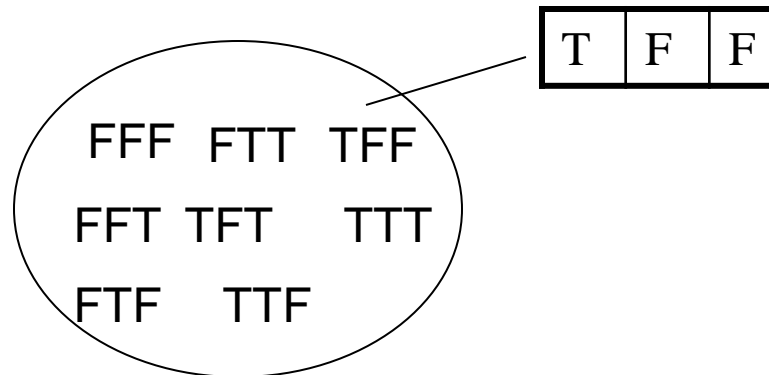


GRAPH

Data Types & Data Structures

- ▶ What is the domain of a structured data type?
Operations?
- ▶ Example: `boolean[] Sample = new boolean[3];`

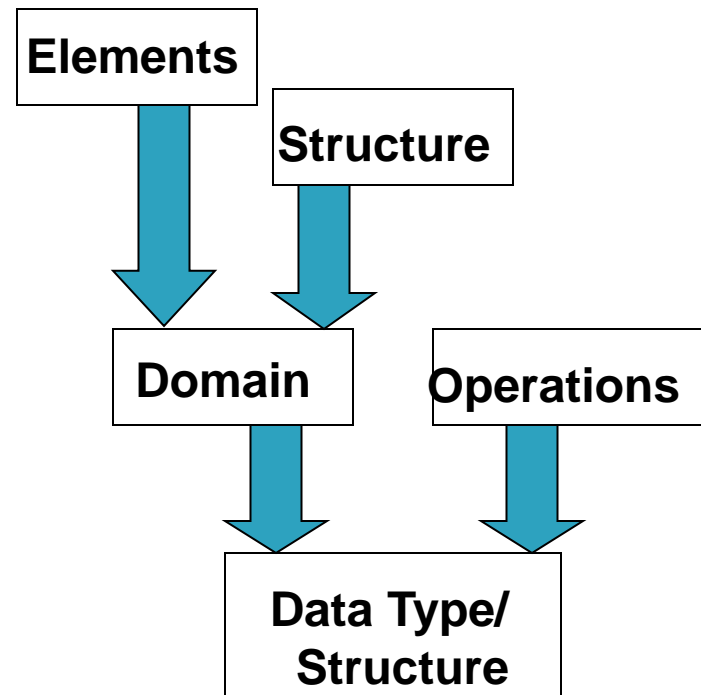
Domain



Data Types & Data Structures

▶ Example: Operations:

```
Sample[0] = true;  
boolean C = Sample[1];
```



Abstract Data Types (ADTs)

- ▶ **Abstraction?** Anything that hides details & provides only the essentials.
- ▶ Examples: an integer $165 = 1 \times 10^2 + 6 \times 10^1 + 5 \times 10^0$,
procedures/subprograms, etc.
- ▶ **Abstract Data Types (ADTs):** Simple or structured data types whose implementation details are hidden...



ADTs

- ▶ While designing ADTs, a designer has to deal with two types of questions:
 - (i) **What** values are in the domain? **What** operations can be performed on the values of a particular data type?
 - (ii) **How** is the data type represented? **How** are the operations implemented?

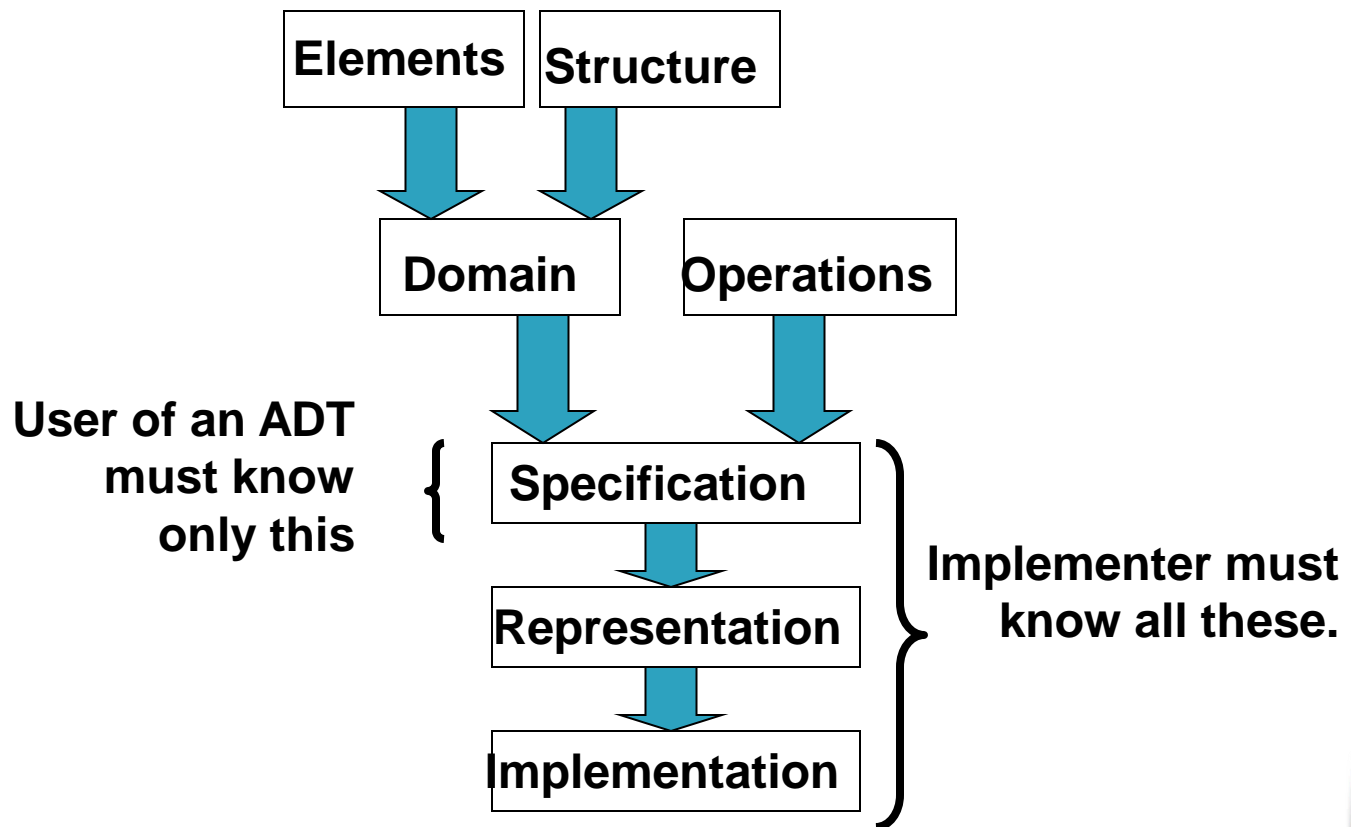


ADTs

- ▶ ADTs **specification** answers the ‘what’ questions. Specification is written first.
- ▶ ADTs **implementation** answers the ‘how’ questions. Done after specification.
- ▶ Users & Implementers:
 - Users of an ADT need only know the specification No implementation details. ← advantage
 - Programmer (Implementer) who implements ADT is concerned with..specification, representation, implementation.



ADTs



ADT: Example

ADT String1

Specification:

Elements: type char.

Structure: elements (characters) are linearly arranged.

Domain: type String, finite domain, there are 0 to 80 chars in a string, therefore $128^0 + 128^1 + 128^2 + \dots + 128^{80}$ possible strings in the domain.

Operations: Assume that there is a string S.

1. **Procedure** Append (c: char)

Requires: $\text{length}(S) < 80$.

Results: c is appended to the right end of S.



ADT: Example

2. Procedure Remove (c: char)

Requires: $\text{length}(S) > 0$.

Results: The rightmost character of S is removed and placed in c, S's length decreases by 1.

3. Procedure MakeEmpty ()

Results: all characters are removed.

4. Procedure Concatenate (R: String)

Results: String R is concatenated to the right of string S, result placed into S.

5. Procedure Reverse ()

6. Procedure Length (L: int)

7. Procedure Equal (S: String, flag: boolean)

8. Procedure GetChar (int i)



Remember

- ▶ In Java the *class* construct is used to declare new data types.
- ▶ In Java operations are implemented as function members of classes or methods.



ADT String: Implementation

```
public class String1 extends Object {  
    private char[] str;  
    private int    size;  
  
    public String1 () {  
        size = -1;  
        str = new char[80];  
    }  
  
    public void Append (char c) {  
        size++;  
        str[size] = c;  
    }  
}
```

Representation

Implementation



ADT String: Implementation

```
public char Remove () {  
    char c = str[size];  
    size--;  
    return(c);  
}  
public char GetChar(int i) {  
    return(str[i]);  
}  
public void MakeEmpty () {  
    size = -1;  
}  
public int Length () {  
    return(size);  
}
```



ADT String: Implementation

```
public void Concatenate (String1 s) {  
    for (int i = 0; i<=s.Length(); i++) {  
        char c = s.GetChar(i);  
        Append(c);  
    }  
}  
  
public boolean Equal (String1 s) {  
}  
  
public void Reverse () {  
}  
}
```



Using ADT String

```
import java.lang.*;
public class Test {
    public static void main(String[] args) {
        String1 s = new String1();
        String1 s1 = new String1();
        System.out.println("Hello, World");
        s.Append('a');
        s1.Append('b');
        s.Concatenate(s1);
        System.out.print(s.GetChar(0));
        System.out.println(s.GetChar(1));
    }
}
```



ToDo

- ▶ Read 2.1, 2.2, 2.3 of the Textbook.
- ▶ Program the String1 ADT.
- ▶ Implement the reverse and equals operations.
- ▶ Test This ADT using a test Class.

