Data Structures & Algorithms

Adil M. Khan
Professor of Computer Science
Innopolis University

1. Data Types vs. Data Structures

2. Interfaces

3. Collections

Data Types vs. Data Structures

Key Point

- ***Data Type:*** a set of values together with operations on that type

- ***Data Structure:*** a physical implementation of a data type

- Same data type can be realized using many different data structures

- Some realizations are "*good*"; others not so much

What do you think I mean by "good"?

- Data Structures:
  - Two fundamental kinds:
    1. arrays of contiguous memory locations
    2. linked structures
- We can even combine the two mechanisms

Implementing Types with Structures

- Example:

"**Stack**" — A type can be implemented as either a "**LinkedStack**" or an "**ArrayStack**" — which are the structures.

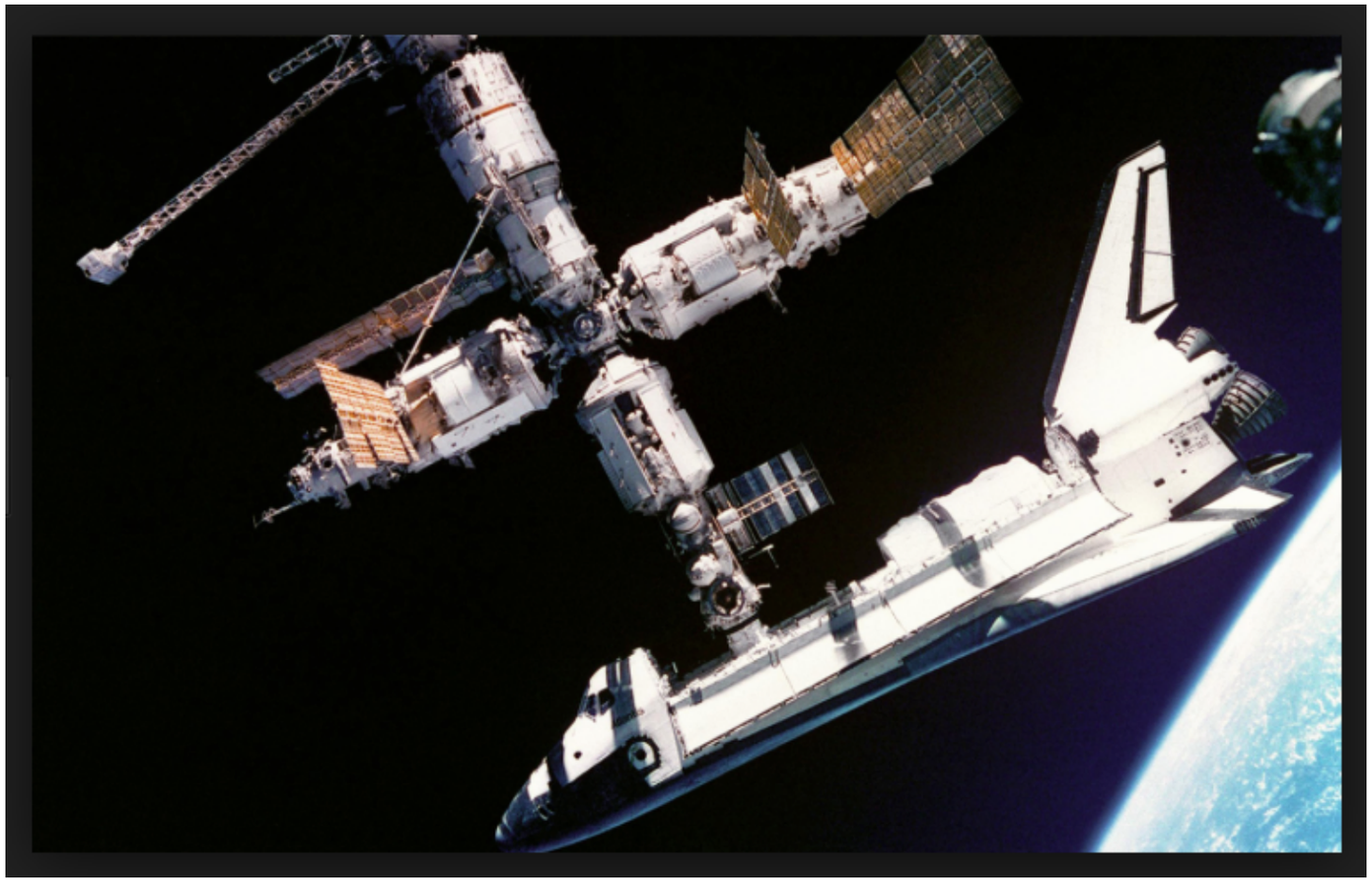
- In Java, Stack would be an interface, whereas LinkedStack and ArrayStack would be classes (fully-specified, concrete classes)

Interfaces

“Separating interface from implementation”

What is an Interface?

- ISS with its docking device on which space shuttles can dock to it
- What if you put the same device on the roof of your house?
- Will the space shuttle be able to dock to your house?
- Yes!



"Thus if you house has the device then it is ***Dockable***, just like the ISS. Both the ISS and your house can implement the Dockable interface."

But just having the device is not enough, the implementation behind it is also needed for proper working

Interfaces

- A kind of a contract that specifies behavior only
- Implementation is given in a real class

Interfaces in Java

- A collection of method declarations with no data and no bodies

- That is, the methods are empty

- Simply method signatures

- If a class wants this behavior, it must implement the interface and provide implementation

Interface Example

- An antique photograph that we want to sell
- So we go to an antique buyer
- The buyer needs the following for any item to be **Sellable**
  - Description of the item
  - Listed price
  - Lowest price

Interface Sellable

```
/** Interface for objects that can be sold. */
public interface Sellable {

  /** Returns a description of the object. */
  public String description();

  /** Returns the list price in cents. */
  public int listPrice();

  /** Returns the lowest price in cents we will accept. */
  public int lowestPrice();
}
```

A Sellable Photograph

```
/** Class for photographs that can be sold. */

public class Photograph implements Sellable {

  private String descript;                    // description of this photo

  private int price;                         // the price we are setting

  private boolean color;                      // true if photo is in color


  public Photograph(String desc, int p, boolean c) {  // constructor
    descript = desc;

    price = p;

    color = c;
  }


  public String description() { return descript; }

  public int listPrice() { return price; }

  public int lowestPrice() { return price/2; }

  public boolean isColor() { return color; }

}
```

So, what should you do if you have a cycle to sell?

Collections

What is a Collection?

- An object that holds a group of objects.
- Some collections
  - allow duplicates, others do not
  - are orders, others are not
  - restrict access to elements according to rules, others do not

Primary Kinds Of Collections	
Set	a group of unordered objects without duplicates
List	a sequence of objects with distinguished first object. Various restricted sequences: stacks, queues, priority queues
Map	a set of (key, value) pair with unique keys
Hierarchy (Tree)	a group of objects in which every object, except root, has a single parent.
Graph	a group of objects in which each object is related to, somehow or other, to zero or more members of the group

Iterating Collections

Iterating Over Collections

- Two ways to iterate over a collection:
  - a foreach statement
  - an iterator object

foreach Statement

- Iterating over a HashSet of Dog objects

```
Set<Dog> kennel = new HashSet<Dog>();

// The foreach way
for (Dog d : kennel) {
    d.bark();
}
```

Iterator Object

- Iterating over a HashSet of Dog objects

```
Set<Dog> kennel = new HashSet<Dog>();
// The explicit iterator way
    Iterator<Dog> it = kennel.iterator();
    while (it.hasNext()) {
        Dog d = it.next();
        d.bark();
    }
```

What is Iterator?

- It is an interface in Java!

```
public interface Iterator<E> {
    boolean hasNext();
    E next();
    void remove();
}
```