Welcome to the

Java Course

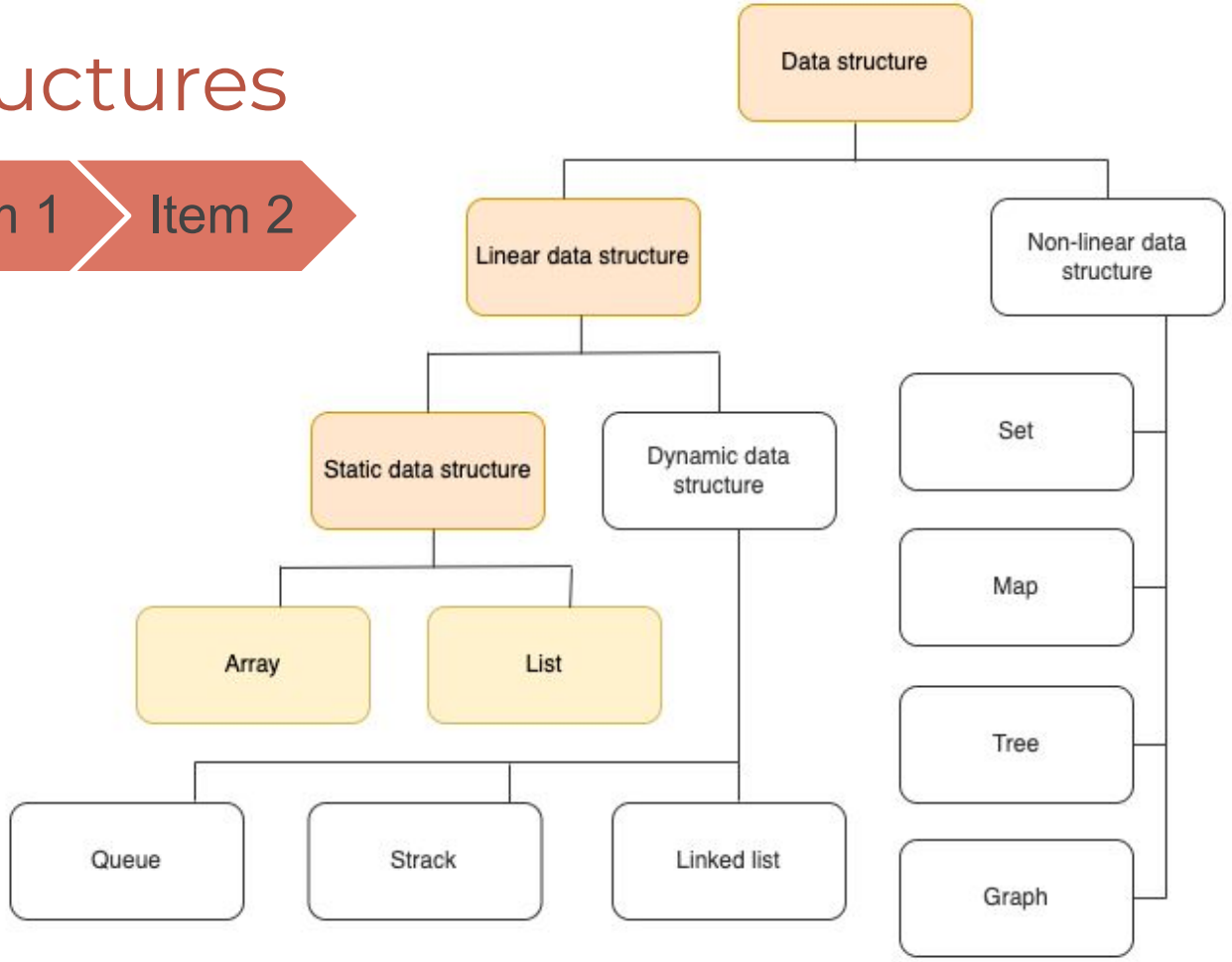
Module 2 – Day 05

Content of the course

- Functions and procedures
- Arrays and lists
- Search and sorting algorithms
- Data structures
- Computational complexity

Data structures

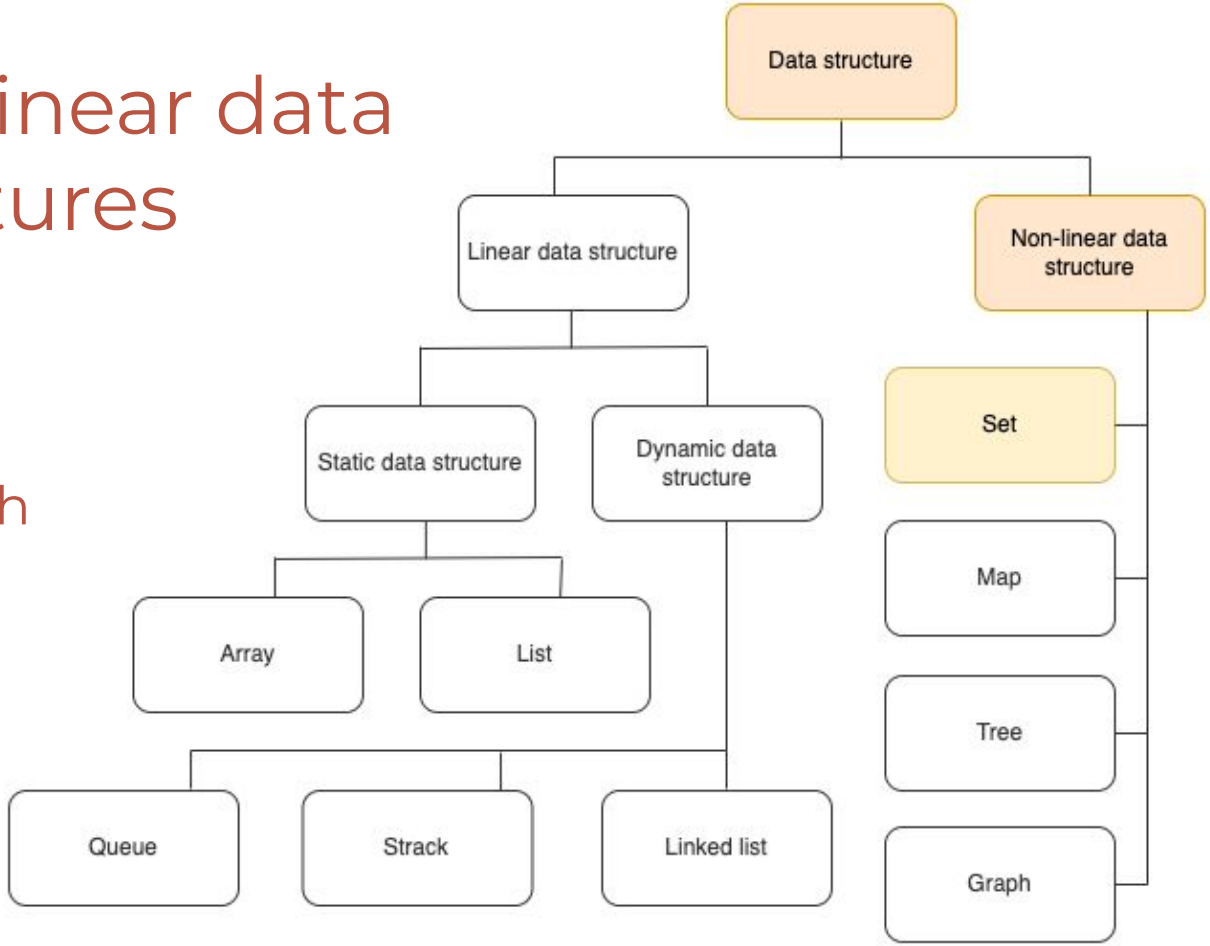
- Item 0
- Item 1
- Item 2



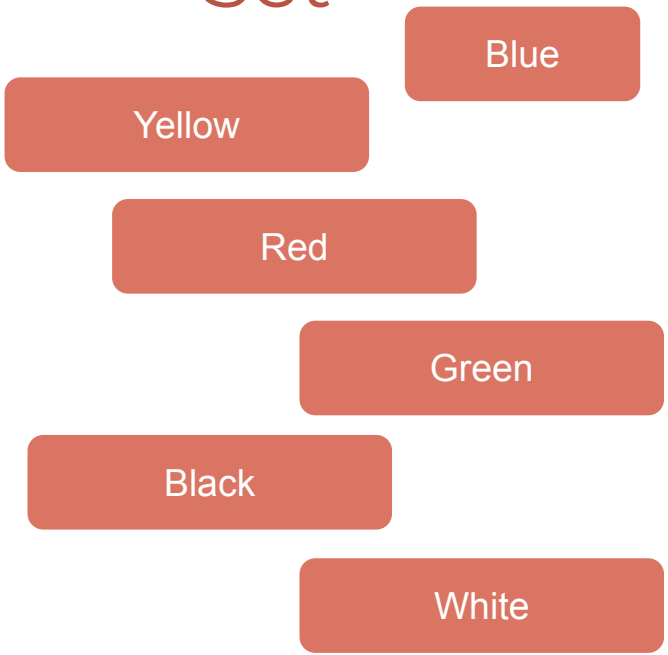
Non-linear data structures

Non-linear data structures

- Set
- Map
- Tree
- Graph



Set



Example

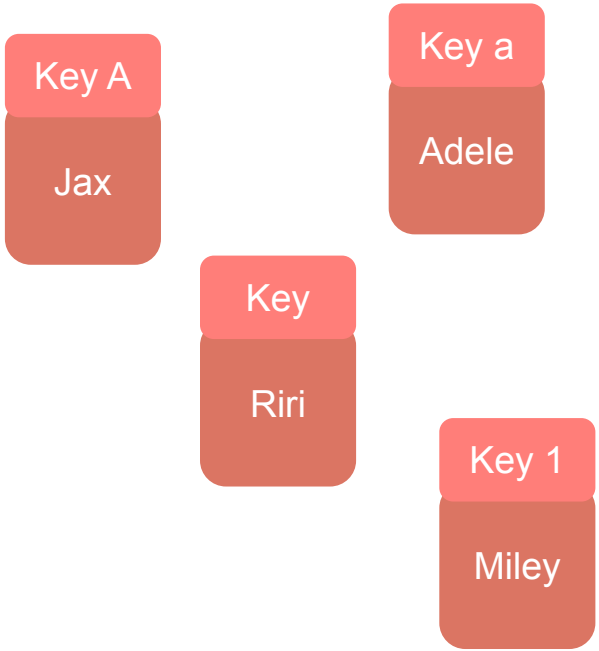
```
// Create a set to store unique colors
Set<String> colors = new HashSet<>();
// Add colors to the set
colors.add("White");
colors.add("Blue");
colors.add("Yellow");
// Attempt to add a duplicate color
colors.add("Blue");

// Print the set of colors
System.out.println("Colors: " + colors);
```

Output

```
Colors: [White, Blue,
Yellow]
```

Map



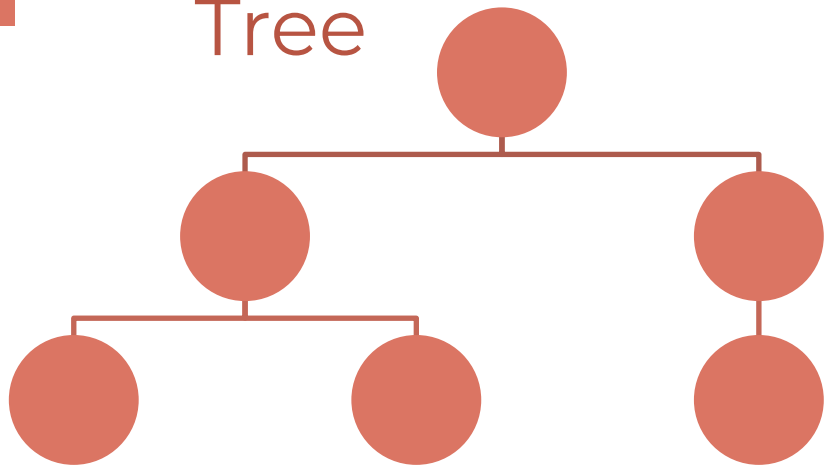
Example

```
// Create a map to store student IDs and their
names
Map<Integer, String> students = new
HashMap<>();
// Add students to the map
students.put(1035, "Alice");
students.put(1037, "Bob");
students.put(1038, "Charlie");
//Print the students
System.out.println("Students: " + students);
// Remove a student by their ID and print the
students again
students.remove(1037);
System.out.println("Students: " + students);
```

Output

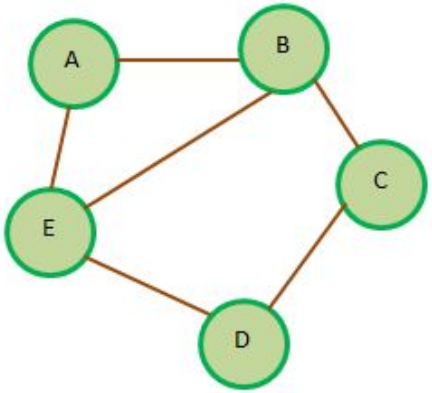
```
Students: {1035=Alice, 1037=Bob, 1038=Charlie}
Students: {1035=Alice, 1038=Charlie}
```

Tree



- A **tree** is a hierarchical structure with a single root from which nodes branch out, used to represent hierarchical relationships.
- **Library:** Trees are not directly provided by the Java Collections Framework, but javax.swing.tree and custom implementations are common.
- **Useful Methods:** Varies by specific tree type (e.g., binary search trees, AVL trees).
- **Example:** File systems on a computer, where each folder can contain files or other folders.

Graph



	A	B	C	D	E
A	0	1	0	0	1
B	1	0	1	0	1
C	0	1	0	1	0
D	0	0	1	0	1
E	1	1	0	1	0

- A **graph** consists of nodes (vertices) connected by edges, capable of representing complex networks.
- **Library:** Like trees, graphs are not part of the Java Collections Framework. Libraries like JGraphT provide graph implementations.
- **Useful Methods:** Varies by library and graph type (e.g., directed, undirected).
- **Example:** Road networks, where intersections are nodes and roads are edges, useful for route planning.

Now YOUR TURN !

Let’s do the exercises

Project Students - Step 9

Modify the program:

- To store each information of each student separately. Instead of storing each student in a String, we will use a Map, therefore, we will have a List of Maps.

- Allow the user to filter the list of students by the course they are registered to.

Project Students - Step 9

```
Options menu:
(a) add a student
(b) remove a student
(c) see the list of students
(d) search for one student
(e) exit
Select an option: a

>>> Student 1 <<<
Enter first name: Ana
Enter last name: Gaggero
Enter birthday (day of month): 22
Enter birth month: 10
Enter birth year: 1982
Enter course registered: Java
Student 1 added.
```

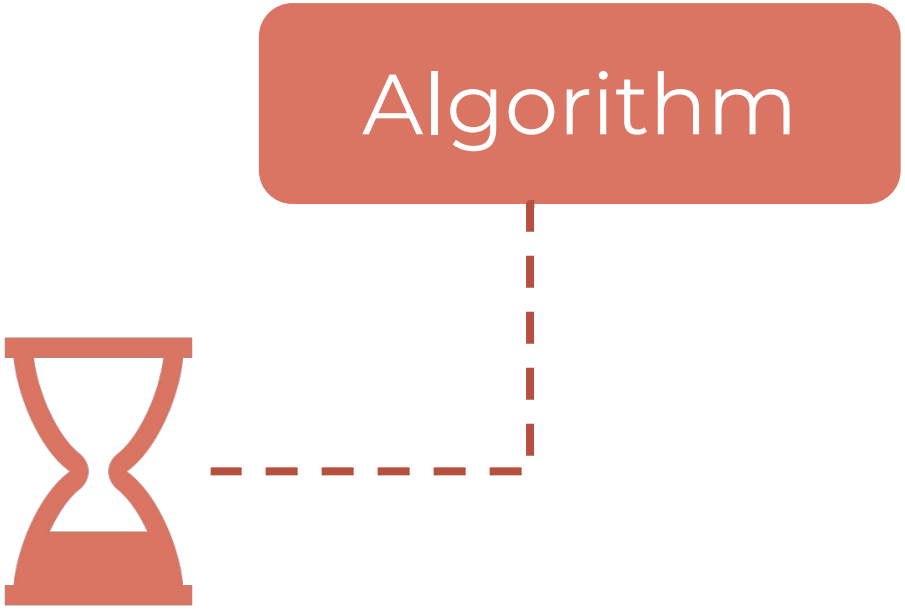
Project Students - Step 9

```
...
Select an option: c
Would you like to filter the list by course? Enter the name of the course
or enter “all” to see the students for all courses: Java
List of students:
Tom Grass born the 7 of January 1980. Registered to Java
Valerie Muller born the 12 of April 1990. Registered to Java
```

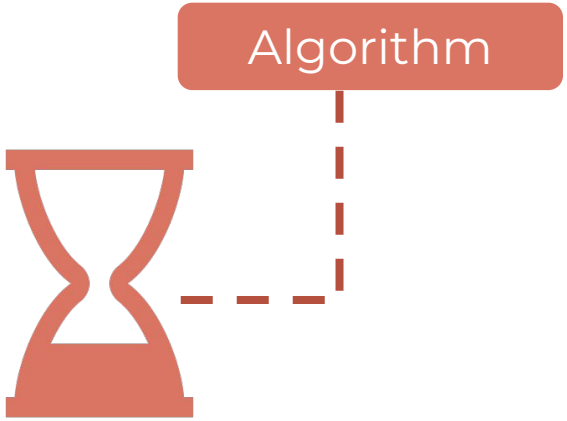
```
...
Select an option: b
Enter the name of the student to be removed: Tom Grass
Student removed.
```

Computational complexity

Complexity



Complexity




```
public void swap(int a, int b){
    int temp = a;
    a = b;
    b = temp;
}
```

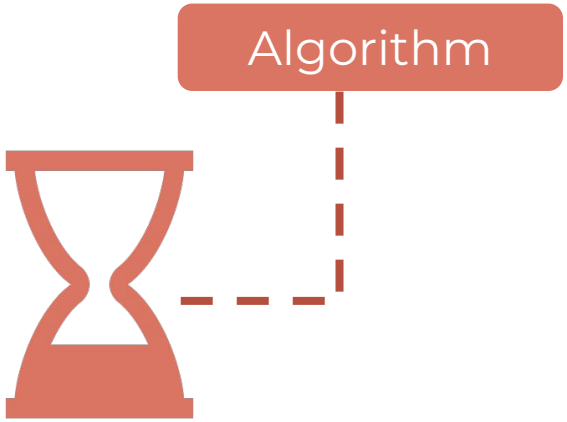
1

1

1

O(1)

Complexity




```
public int sum(int n){
  int sum = 0;
  for (int i = 1; i <= n; i++){
    sum += i;
  }
  return sum;
}
```

1

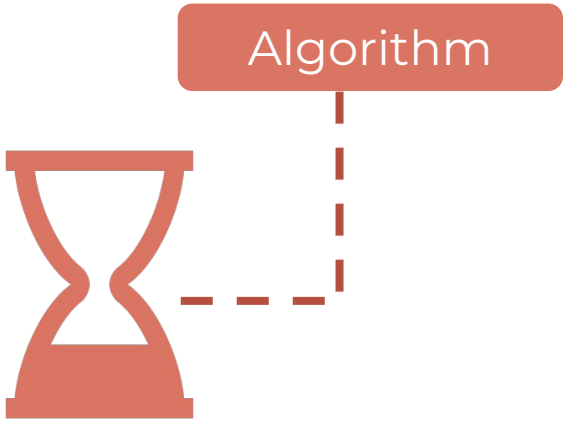
n + 1

n

1

O(n)

Complexity




```
public int sum(int arr[], int n){
  int sum = 0;
  for (int i = 1; i <= n; i++){
    sum += arr[i];
  }
  return sum;
}
```

1

n + 1

n

1

O(n)

Complexity

Algorithm

```
public int sum(int arr[][], int n){
  int sum = 0;
  for (int i = 1; i <= n; i++){
     for (int j = 1; j <= n; j++){
        sum += arr[i][j];
     }
  }
  return sum;
}
```

1

n + 1

n * (n + 1)

n * n

1

$O(n^2)$

Complexity

Algorithm

```
public void algo(int n){

  if (n % 2 == 0)
    #statement

  else {
    for(int i = 0; i< n; i++)
      #statement
  }

}
```

O(1) Best case

$O(n^2)$ Worst case