Tutorial 3

Goals:

- Complexity & Sorting
  - Insertion Sort
  - Quick sort
- OPP Intro

Insertion Sort ($O(n^2)$, $\Omega(n)$, $\Theta(n^2)$)

Upper Bound: $O(n^2)$

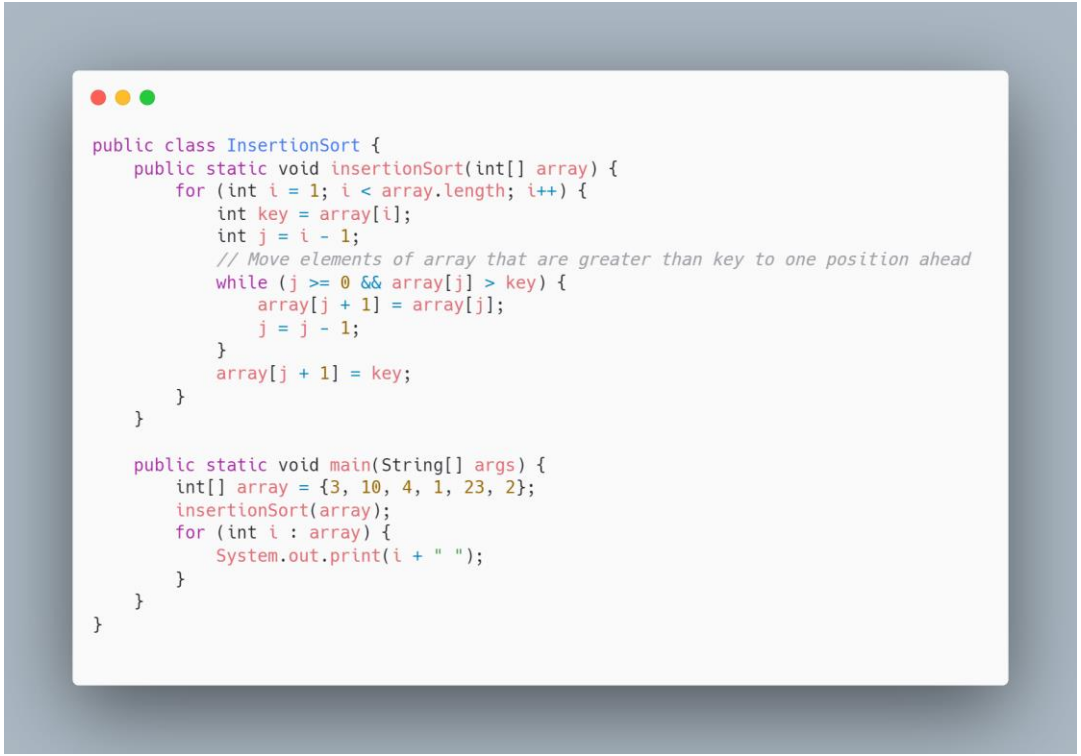
In the worst case, the array is sorted in reverse order, so every new element has to be compared to all previously sorted elements, resulting in n^2 comparisons.

Lower Bound: $\Omega(n)$

In the best case, the array is already sorted, and each element only needs to be compared once, resulting in a linear time of n.

Average Case: $\Theta(n^2)$

On average, the algorithm performs a number of comparisons close to quadratic as most elements will require shifting through multiple comparisons, especially for larger arrays.



```java
public class InsertionSort {
    public static void insertionSort(int[] array) {
        for (int i = 1; i < array.length; i++) {
            int key = array[i];
            int j = i - 1;
            // Move elements of array that are greater than key to one position ahead
            while (j >= 0 && array[j] > key) {
                array[j + 1] = array[j];
                j = j - 1;
            }
            array[j + 1] = key;
        }
    }

    public static void main(String[] args) {
        int[] array = {3, 10, 4, 1, 23, 2};
        insertionSort(array);
        for (int i : array) {
            System.out.print(i + " ");
        }
    }
}
```

Quick Sort (O(n^2),Ω(n log n),Θ(n log n))

Upper Bound: O(n^2)

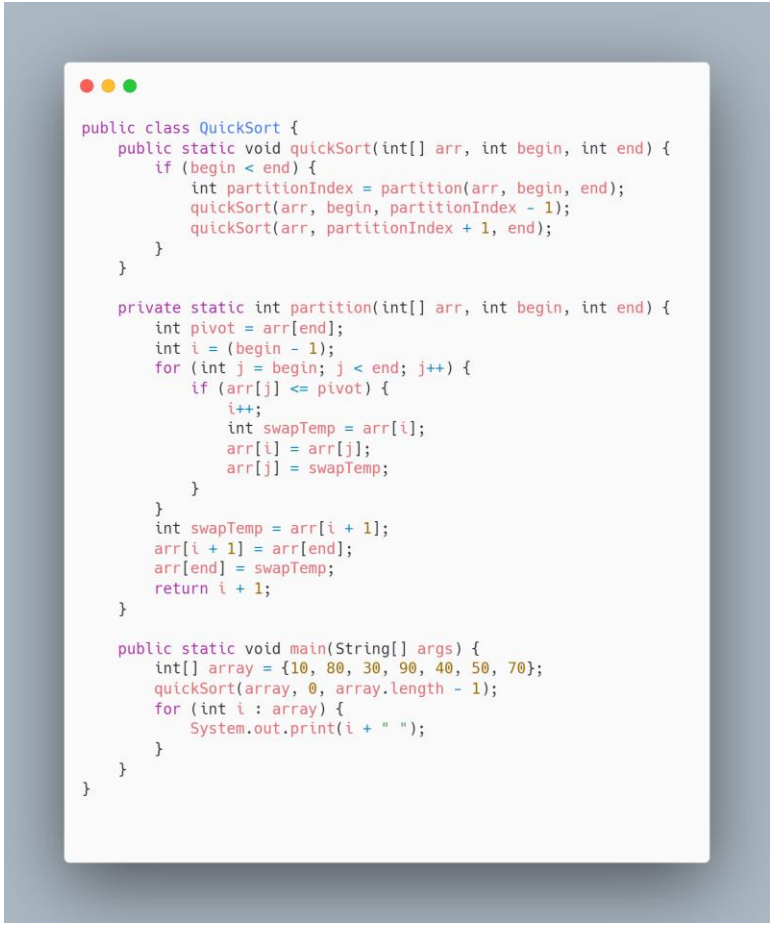
In the worst case, the algorithm will pick the smallest or largest element as the pivot in each iteration, leading to unbalanced partitions. This results in quadratic time complexity, O(n^2).

Lower Bound: Ω(n log n)

In the best case, the pivot divides the list into two equal-sized sub-lists, resulting in logarithmic depth and n comparisons per level. Thus, the time complexity is O(n log n).

Average Case: Θ(n log n)

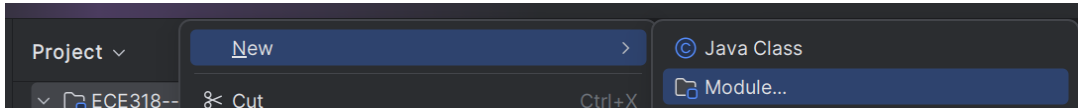
On average, Quick Sort performs well because the pivot tends to split the list into fairly balanced partitions, making the expected time complexity Θ(n log n). It's suitable for large datasets.



```java
public class QuickSort {
    public static void quickSort(int[] arr, int begin, int end) {
        if (begin < end) {
            int partitionIndex = partition(arr, begin, end);
            quickSort(arr, begin, partitionIndex - 1);
            quickSort(arr, partitionIndex + 1, end);
        }
    }

    private static int partition(int[] arr, int begin, int end) {
        int pivot = arr[end];
        int i = (begin - 1);
        for (int j = begin; j < end; j++) {
            if (arr[j] <= pivot) {
                i++;
                int swapTemp = arr[i];
                arr[i] = arr[j];
                arr[j] = swapTemp;
            }
        }
        int swapTemp = arr[i + 1];
        arr[i + 1] = arr[end];
        arr[end] = swapTemp;
        return i + 1;
    }

    public static void main(String[] args) {
        int[] array = {10, 80, 30, 90, 40, 50, 70};
        quickSort(array, 0, array.length - 1);
        for (int i : array) {
            System.out.print(i + " ");
        }
    }
}
```

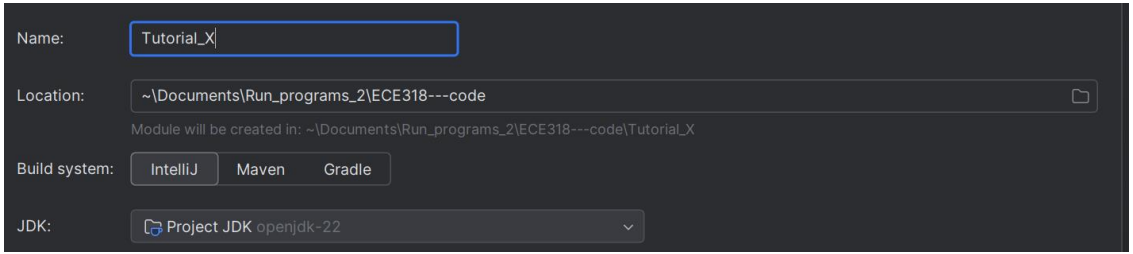
Create Modules

Right click on Project's name

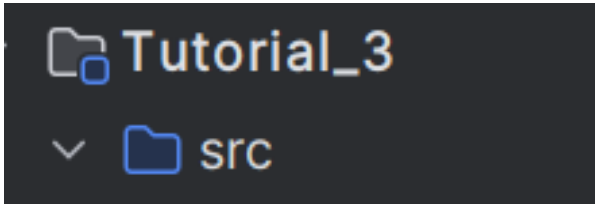


New -> Module

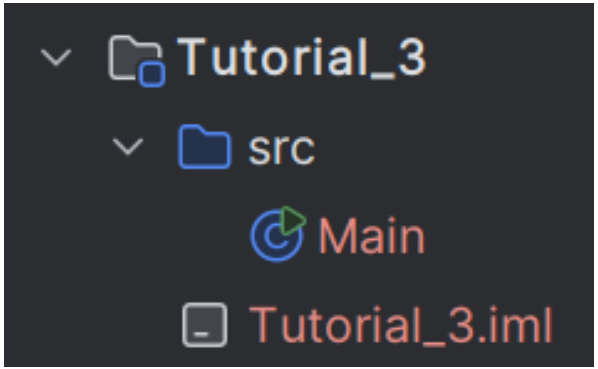
Give a Name and specify the SDK



Then the new Module file looks like:



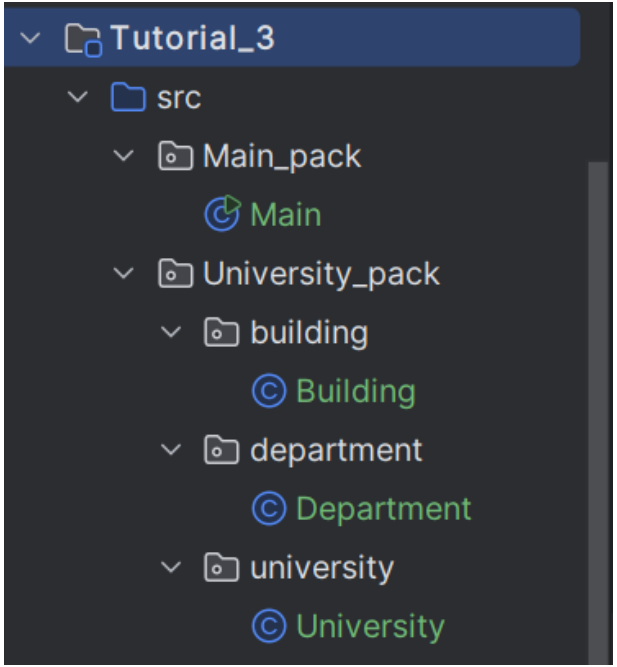
Class




```java
public class Main {
    //Attributes space
    private String title;
    private int version;

    // Constructors : Overload
    public Main() {
      this("Default Program", 1);
    }
    public Main(String title) {
      this(title, 1);
    }
    public Main(String title, int version) {
        this.title = title;
        this.version = version;
    }

    // Method Space
    public void displayInfo() {
        System.out.println(title + " v" + version);
    }

    // Main method
    public static void main(String[] args) {
        new Main().displayInfo();
        new Main("Custom Program").displayInfo();
        new Main("Advanced Program", 2).displayInfo();
    }
}
```

Create Packages

1. Right click on src folder
   New -> Package
2. Create the packages and sub-packages of image




```java
package Main_pack;
// making it accessible within the class.

import University_pack.building.Building;
import University_pack.department.Department;
import University_pack.university.University;
// Declare The package which the class resides.
```

Example 1: Package University_pack

```java
package University_pack.building;

public class Building {
    private String name;

    public Building(String name) {
        this.name = name;
    }

    public String getName() {
        return name;
    }

    @Override
    public String toString() {
        return name;
    }
}
```

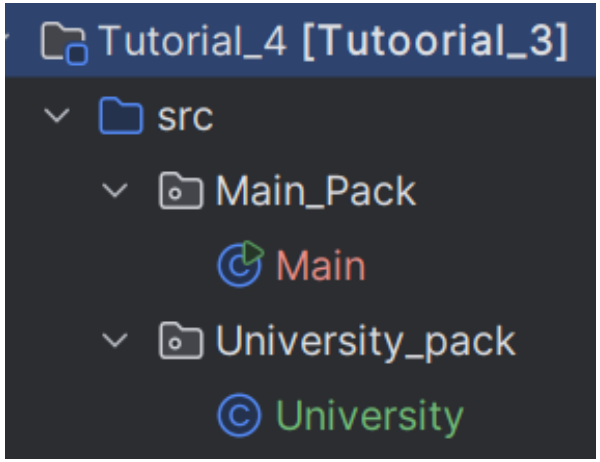
```java
package University_pack.department;

public class Department {
    private String name;

    public Department(String name) {
        this.name = name;
    }

    public String getName() {
        return name;
    }

    @Override
    public String toString() {
        return name;
    }
}
```

```java
package University_pack.university;

import University_pack.building.Building;
import University_pack.department.Department;

import java.util.ArrayList;
import java.util.List;

public class University {
    private String name;
    private List<Department> departments;
    private List<Building> buildings;

    public University(String name) {
        this.name = name;
        this.departments = new ArrayList<>();
        this.buildings = new ArrayList<>();
    }

    public String getName() {
        return name;
    }

    public void addDepartment(Department department) {
        departments.add(department);
    }

    public void addBuilding(Building building) {
        buildings.add(building);
    }

    public Department[] getDepartments() {
        return departments.toArray(new Department[0]);
    }

    public Building[] getBuildings() {
        return buildings.toArray(new Building[0]);
    }

    @Override
    public String toString() {
        return name;
    }
```

Example 1: Main

```java
package Main_pack;

import University_pack.building.Building;
import University_pack.department.Department;
import University_pack.university.University;

import java.util.Arrays;

public class Main {
    public static void main(String[] args) {
        // Create Departments
        Department csDepartment = new Department("Computer Science");
        Department eeDepartment = new Department("Electrical Engineering");

        // Create Buildings
        Building scienceHall = new Building("Science Hall");
        Building techTower = new Building("Tech Tower");

        // Create University 1
        University university1 = new University("University of Example");
        university1.addDepartment(csDepartment);
        university1.addBuilding(scienceHall);

        // Create University 2
        University university2 = new University("Example Institute of Technology");
        university2.addDepartment(eeDepartment);
        university2.addBuilding(techTower);

        // Display University 1 details
        System.out.println("University: " + university1.getName());
        System.out.println("Departments: " + Arrays.toString(university1.getDepartments()));
        System.out.println("Buildings: " + Arrays.toString(university1.getBuildings()));

        // Display University 2 details
        System.out.println("\nUniversity: " + university2.getName());
        System.out.println("Departments: " + Arrays.toString(university2.getDepartments()));
        System.out.println("Buildings: " + Arrays.toString(university2.getBuildings()));
    }
}
```

Next Example




```java
package Main_Pack;

import University_pack.University;

public class Main {
    public static void main(String[] args) {
        University uni = new University(10, 5, 15);

        // Accessible because publicCourses is public
        System.out.println("Accessing public: " + uni.publicCourses);

        // Error: packageCourses is package-private, not accessible from different package
        // System.out.println("Accessing package-private: " + uni.packageCourses);

        // Error: privateCourses is private, not accessible outside University class
        // System.out.println("Accessing private: " + uni.privateCourses);
        // Calling the displayInfo method
        uni.displayInfo();
    }
}
```

```java
package University_pack;

public class University {
    int packageCourses;            // Package-private (accessible within the same package)
    private int privateCourses;    // Private (accessible only within this class)
    public int publicCourses;      // Public (accessible from any class)

    // Constructor
    public University(int packageCourses, int privateCourses, int publicCourses) {
        this.packageCourses = packageCourses;
        this.privateCourses = privateCourses;
        this.publicCourses = publicCourses;
    }

    // Method to display information
    public void displayInfo() {
        System.out.println("Package-private courses: " + packageCourses);
        System.out.println("Private courses: " + privateCourses);
        System.out.println("Public courses: " + publicCourses);
    }
}
```

Task

Exercise: ATM and Customer Interaction with BankAccount

Overview This exercise simulates a real-world banking system where a Customer interacts with their BankAccount through an ATM. Only the ATM can modify the balance, allowing Customer requests for deposits and withdrawals while maintaining controlled access to sensitive account data.

Objectives Understand the concept of encapsulation and controlled access using access modifiers (public, private, package-private). Implement a system where only specific classes (ATM) can modify sensitive data (balance). Demonstrate interaction between multiple classes using method calls and access control.

```
ATMExampleProject
└── src
    ├── mainPackage
    │   └── Main.java
    ├── customerPackage
    │   └── Customer.java
    └── bankPackage
        ├── BankAccount.java
        └── ATM.java
```

** Tip: look at Teams for the structure(optional)