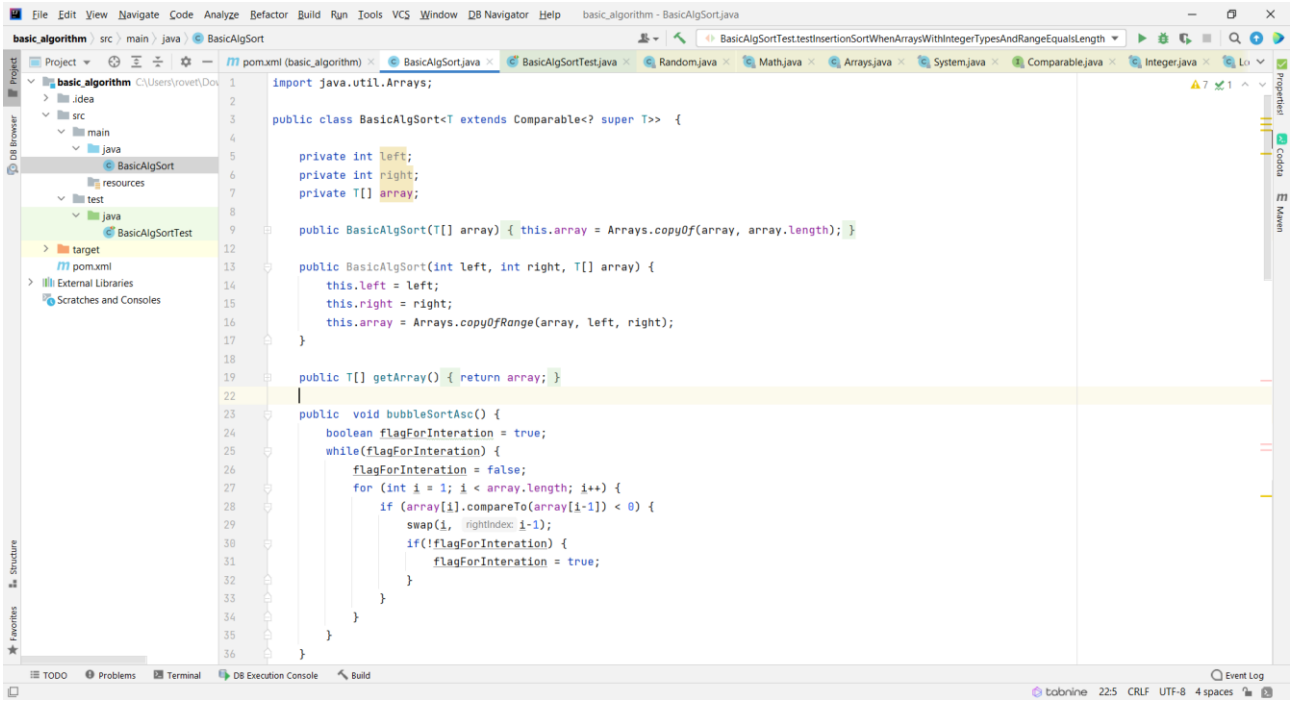
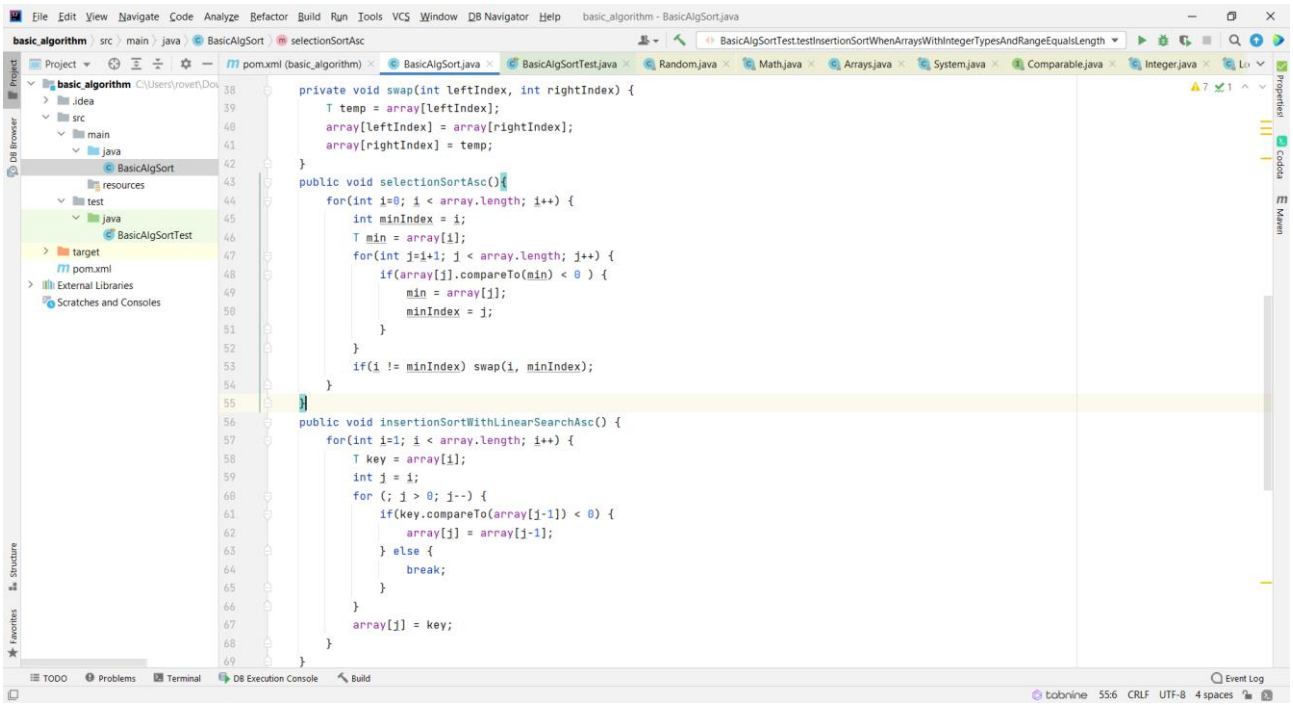
Робота №1. Базові алгоритми сортування

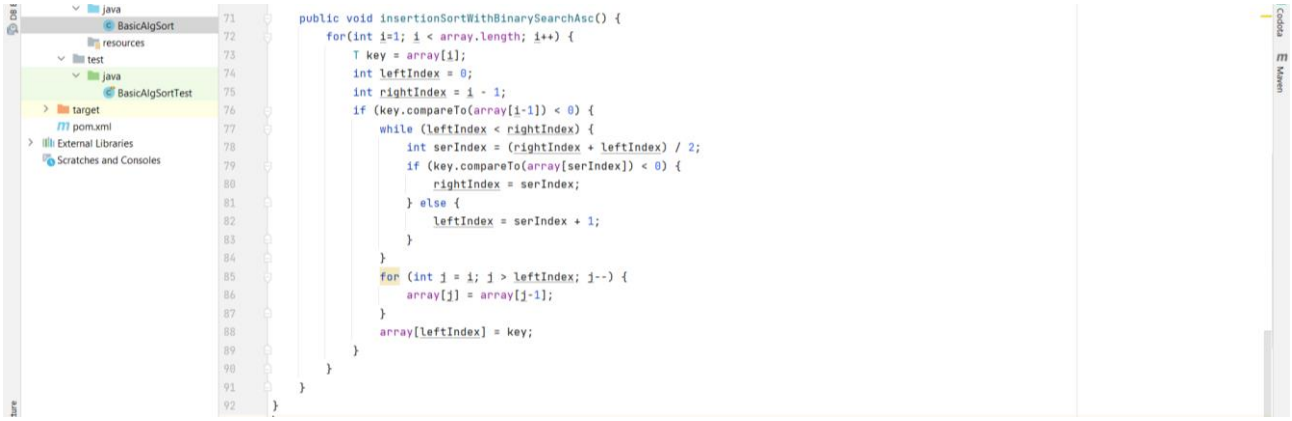
Бібліотека для сортування за зростанням масиву елементів на основі алгоритмів “бульбашкового” сортування, сортування вибором і включенням.



```java
import java.util.Arrays;

public class BasicAlgSort<T extends Comparable<? super T>>  {

    private int left;
    private int right;
    private T[] array;

    public BasicAlgSort(T[] array) { this.array = Arrays.copyOf(array, array.length); }

    public BasicAlgSort(int left, int right, T[] array) {
        this.left = left;
        this.right = right;
        this.array = Arrays.copyOfRange(array, left, right);
    }

    public T[] getArray() { return array; }

    public  void bubbleSortAsc() {
        boolean flagForInteration = true;
        while(flagForInteration) {
            flagForInteration = false;
            for (int i = 1; i < array.length; i++) {
                if (array[i].compareTo(array[i-1]) < 0) {
                    swap(i,  rightIndex: i-1);
                    if(!flagForInteration) {
                        flagForInteration = true;
                    }
                }
            }
        }
    }
```

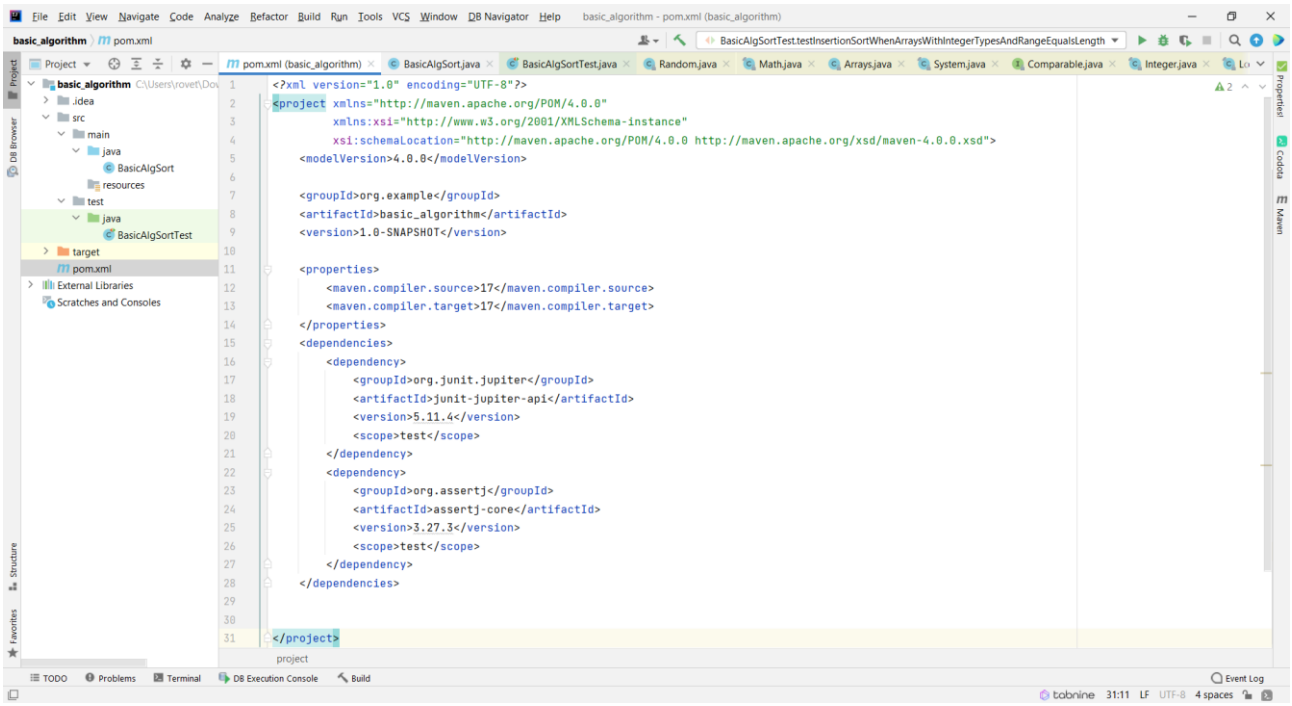


```java
    private void swap(int leftIndex, int rightIndex) {
        T temp = array[leftIndex];
        array[leftIndex] = array[rightIndex];
        array[rightIndex] = temp;
    }
    public void selectionSortAsc(){
        for(int i=0; i < array.length; i++) {
            int minIndex = i;
            T min = array[i];
            for(int j=i+1; j < array.length; j++) {
                if(array[j].compareTo(min) < 0 ) {
                    min = array[j];
                    minIndex = j;
                }
            }
            if(i != minIndex) swap(i, minIndex);
        }
    }
    public void insertionSortWithLinearSearchAsc() {
        for(int i=1; i < array.length; i++) {
            T key = array[i];
            int j = i;
            for (; j > 0; j--) {
                if(key.compareTo(array[j-1]) < 0) {
                    array[j] = array[j-1];
                } else {
                    break;
                }
            }
            array[j] = key;
        }
    }
```



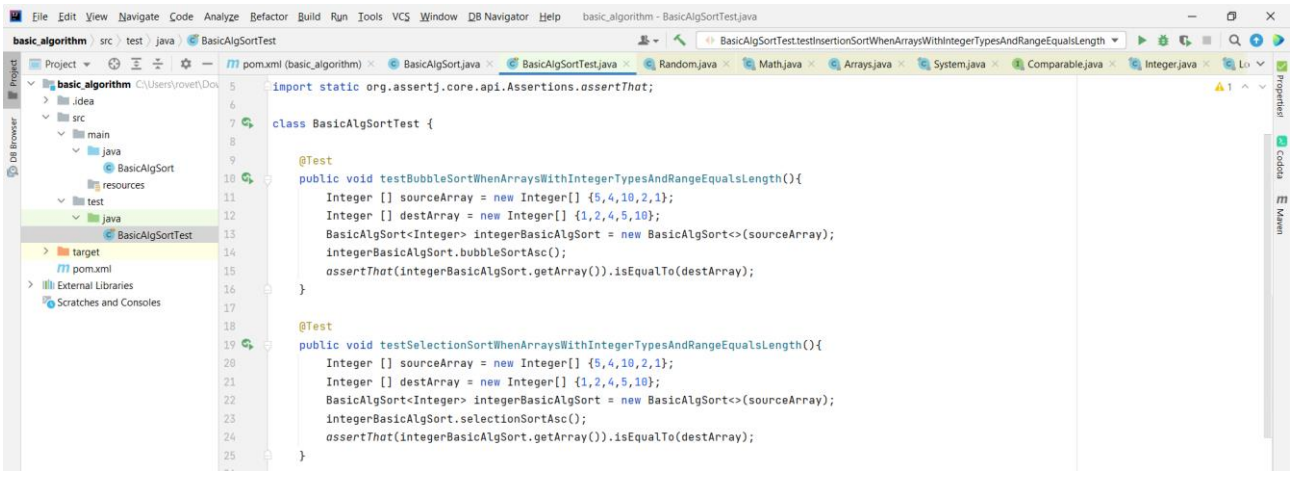
```
    public void insertionSortWithBinarySearchAsc() {
        for(int i=1; i < array.length; i++) {
            T key = array[i];
            int leftIndex = 0;
            int rightIndex = i - 1;
            if (key.compareTo(array[i-1]) < 0) {
                while (leftIndex < rightIndex) {
                    int serIndex = (rightIndex + leftIndex) / 2;
                    if (key.compareTo(array[serIndex]) < 0) {
                        rightIndex = serIndex;
                    } else {
                        leftIndex = serIndex + 1;
                    }
                }
                for (int j = i; j > leftIndex; j--) {
                    array[j] = array[j-1];
                }
                array[leftIndex] = key;
            }
        }
    }
}
```

Залежності, необхідні для тестування

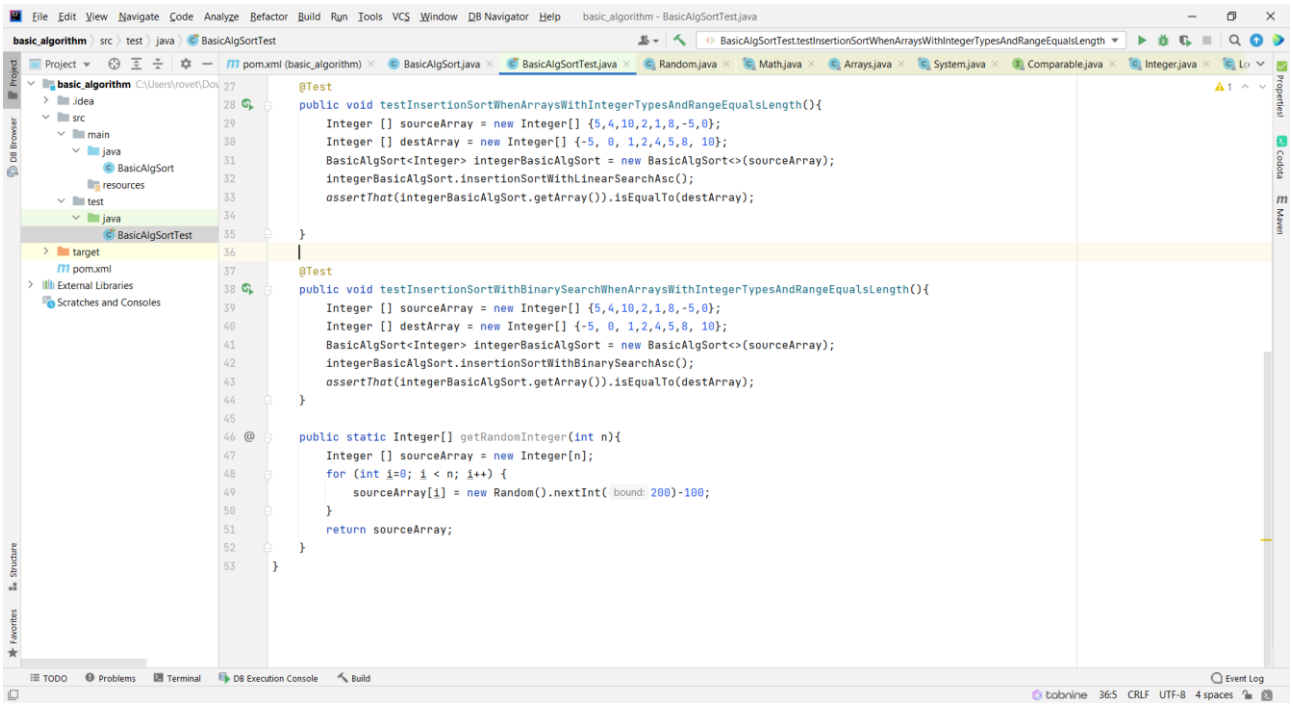


```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>org.example</groupId>
    <artifactId>basic_algorithm</artifactId>
    <version>1.0-SNAPSHOT</version>

    <properties>
        <maven.compiler.source>17</maven.compiler.source>
        <maven.compiler.target>17</maven.compiler.target>
    </properties>
    <dependencies>
        <dependency>
            <groupId>org.junit.jupiter</groupId>
            <artifactId>junit-jupiter-api</artifactId>
            <version>5.11.4</version>
            <scope>test</scope>
        </dependency>
        <dependency>
            <groupId>org.assertj</groupId>
            <artifactId>assertj-core</artifactId>
            <version>3.27.3</version>
            <scope>test</scope>
        </dependency>
    </dependencies>


</project>
```

Тестування бібліотеки для сортування масиву елементів



```
import static org.assertj.core.api.Assertions.assertThat;

class BasicAlgSortTest {

    @Test
    public void testBubbleSortWhenArraysWithIntegerTypesAndRangeEqualsLength(){
        Integer [] sourceArray = new Integer[] {5,4,10,2,1};
        Integer [] destArray = new Integer[] {1,2,4,5,10};
        BasicAlgSort<Integer> integerBasicAlgSort = new BasicAlgSort<>(sourceArray);
        integerBasicAlgSort.bubbleSortAsc();
        assertThat(integerBasicAlgSort.getArray()).isEqualTo(destArray);
    }

    @Test
    public void testSelectionSortWhenArraysWithIntegerTypesAndRangeEqualsLength(){
        Integer [] sourceArray = new Integer[] {5,4,10,2,1};
        Integer [] destArray = new Integer[] {1,2,4,5,10};
        BasicAlgSort<Integer> integerBasicAlgSort = new BasicAlgSort<>(sourceArray);
        integerBasicAlgSort.selectionSortAsc();
        assertThat(integerBasicAlgSort.getArray()).isEqualTo(destArray);
    }
```



```
    @Test
    public void testInsertionSortWhenArraysWithIntegerTypesAndRangeEqualsLength(){
        Integer [] sourceArray = new Integer[] {5,4,10,2,1,8,-5,0};
        Integer [] destArray = new Integer[] {-5, 0, 1,2,4,5,8, 10};
        BasicAlgSort<Integer> integerBasicAlgSort = new BasicAlgSort<>(sourceArray);
        integerBasicAlgSort.insertionSortWithLinearSearchAsc();
        assertThat(integerBasicAlgSort.getArray()).isEqualTo(destArray);

    }
    @Test
    public void testInsertionSortWithBinarySearchWhenArraysWithIntegerTypesAndRangeEqualsLength(){
        Integer [] sourceArray = new Integer[] {5,4,10,2,1,8,-5,0};
        Integer [] destArray = new Integer[] {-5, 0, 1,2,4,5,8, 10};
        BasicAlgSort<Integer> integerBasicAlgSort = new BasicAlgSort<>(sourceArray);
        integerBasicAlgSort.insertionSortWithBinarySearchAsc();
        assertThat(integerBasicAlgSort.getArray()).isEqualTo(destArray);
    }

    public static Integer[] getRandomInteger(int n){
        Integer [] sourceArray = new Integer[n];
        for (int i=0; i < n; i++) {
            sourceArray[i] = new Random().nextInt( bound: 200)-100;
        }
        return sourceArray;
    }
}
```

Завдання 1. Доповнити бібліотеку методами для сортування за спаданням масиву випадково згенерованих [-100, 100] елементів в деякому заданому діапазоні індексів [leftIndex, rightIndex], де leftIndex, rightIndex = 0,….,n-1, n - кількість елементів масиву і leftIndex < rightIndex. Протестувати отримані методи.

Завдання 2. Порівняти часи виконання алгоритмів сортування включенням з лінійним пошуком та включенням з бінарним пошуком для сортування масиву випадково згенерованих [-100, 100] елементів з кількістю n=1000, 10_000, 50_000, 100_000. Результати представити графічно, як залежність часу виконання алгоритмів від кількості елементів масиву.