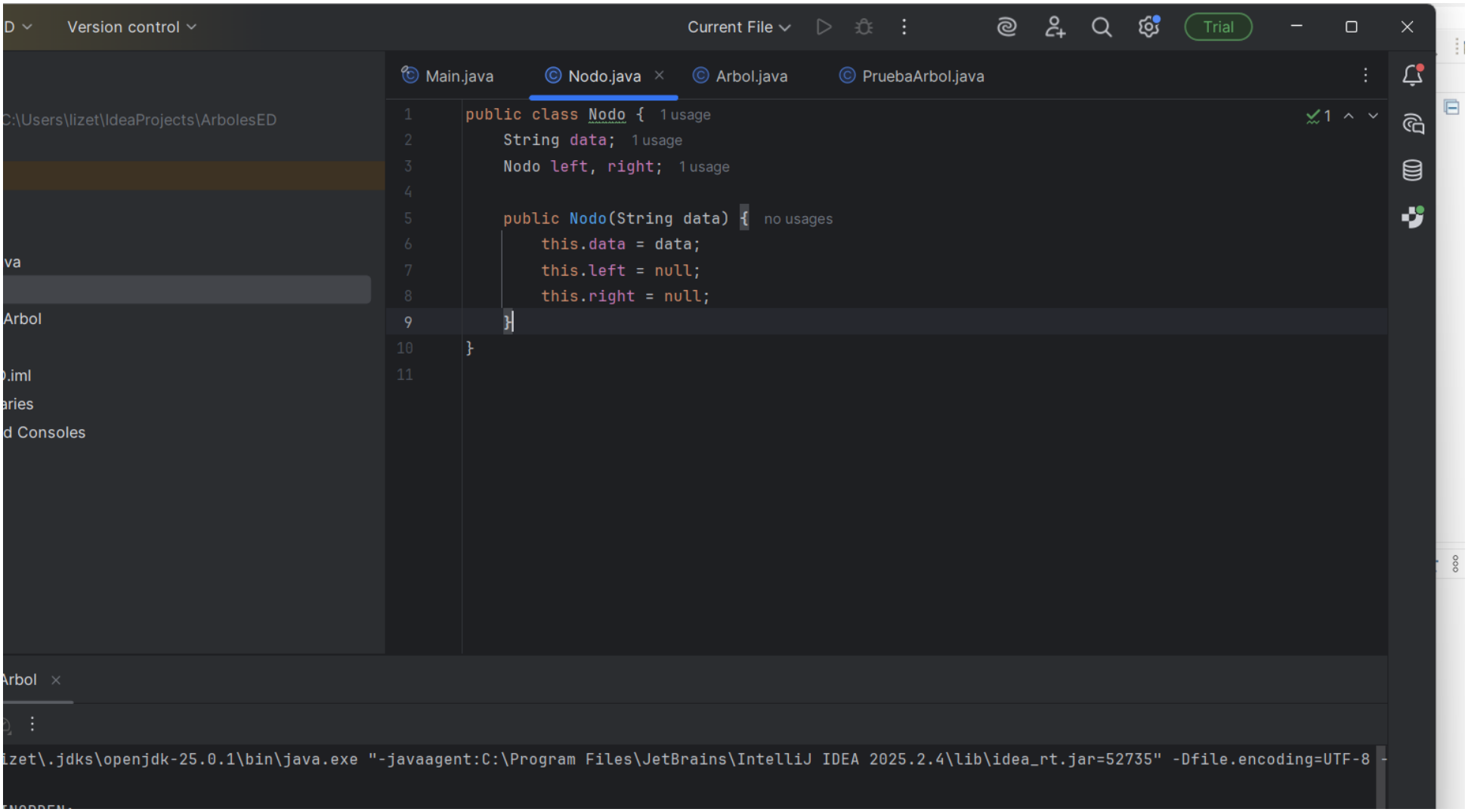
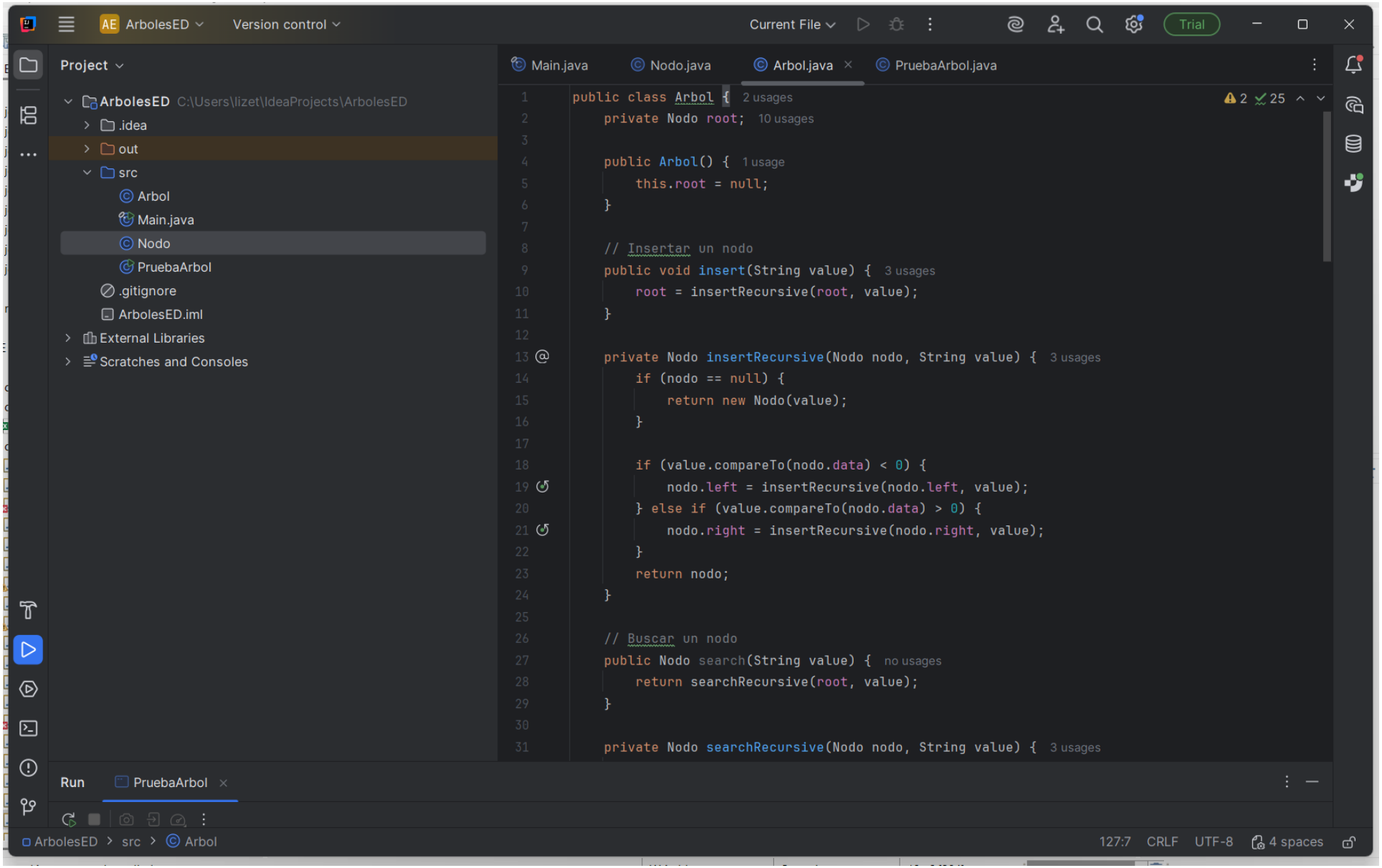
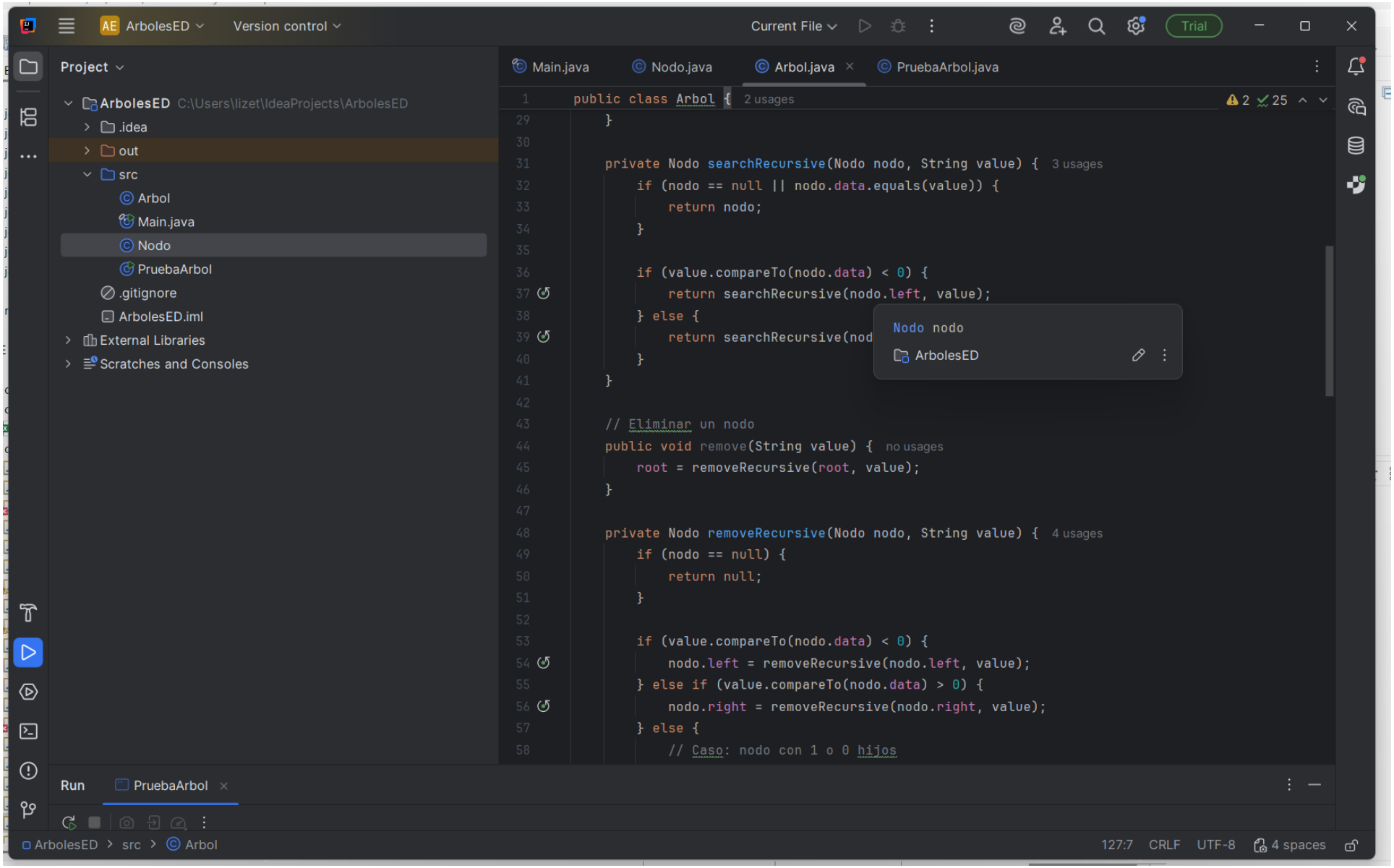
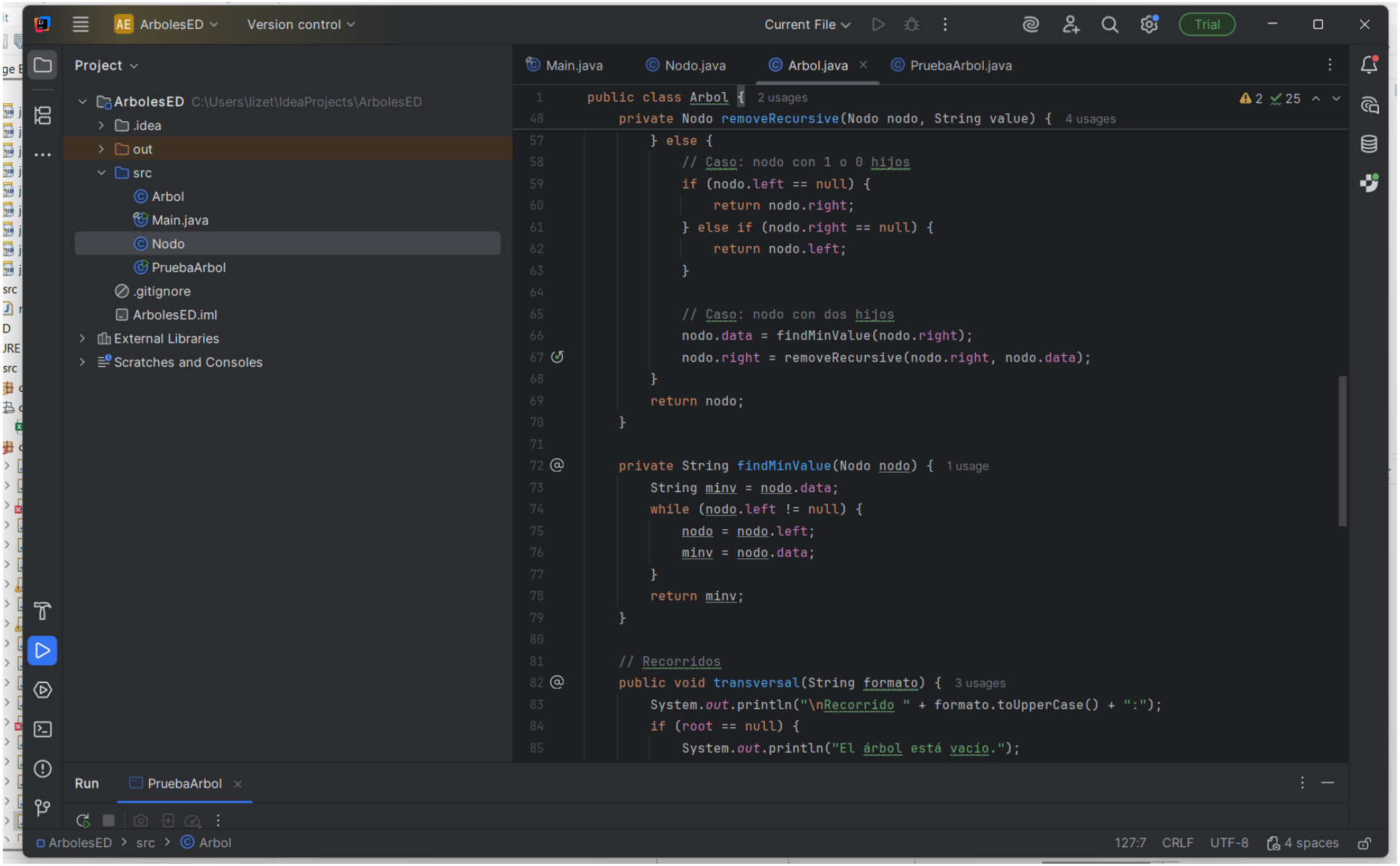
Clase Nodo

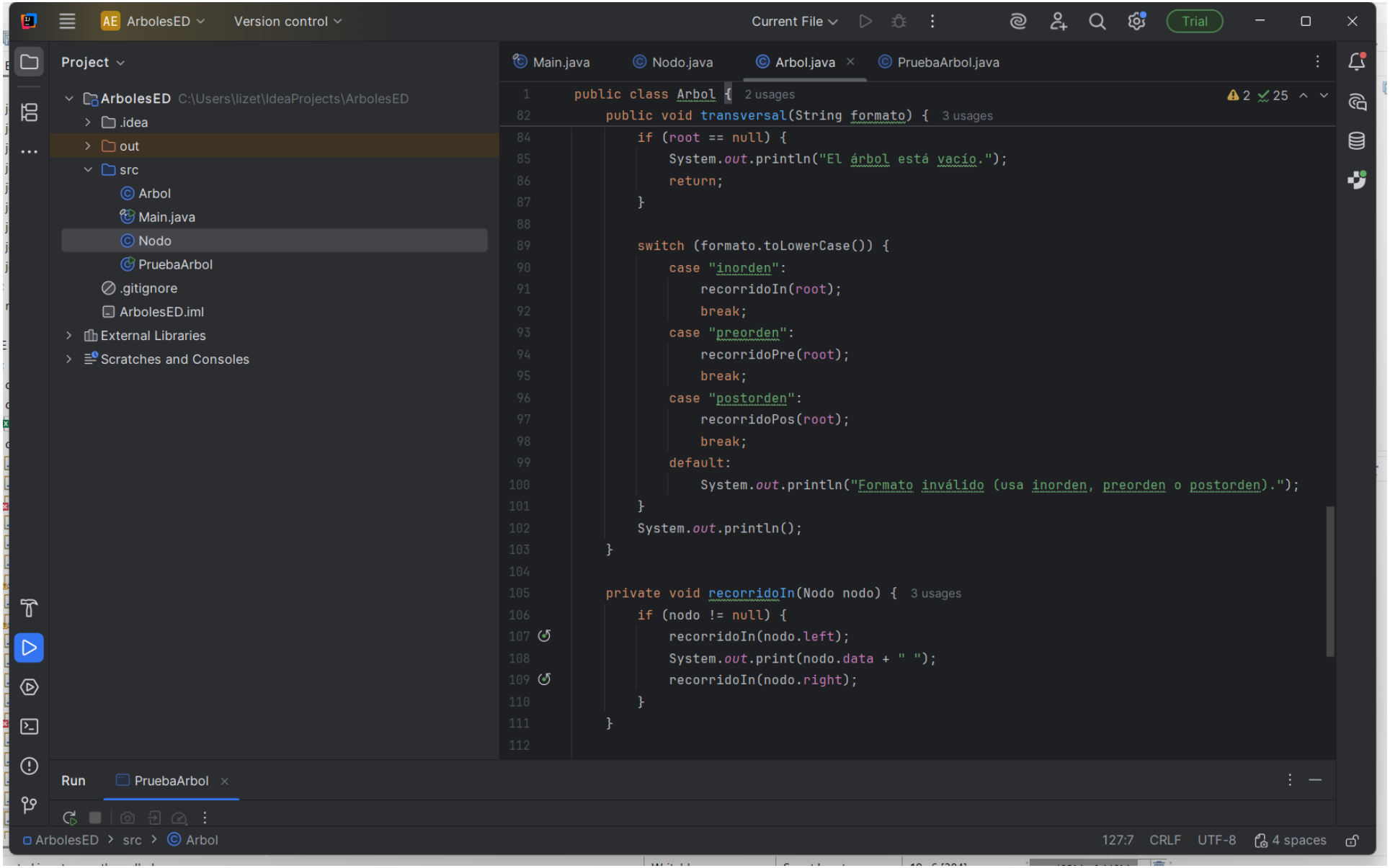

Version control
Current File
Trial
Main.java
Nodo.java
Arbol.java
PruebaArbol.java
C:\Users\lizet\IdeaProjects\ArbolesED
public class Nodo { 1 usage
String data; 1 usage
Nodo left, right; 1 usage
public Nodo(String data) { no usages
this.data = data;
this.left = null;
this.right = null;
}
}
Arbol
izet\.jdks\openjdk-25.0.1\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2025.2.4\lib\idea_rt.jar=52735" -Dfile.encoding=UTF-8

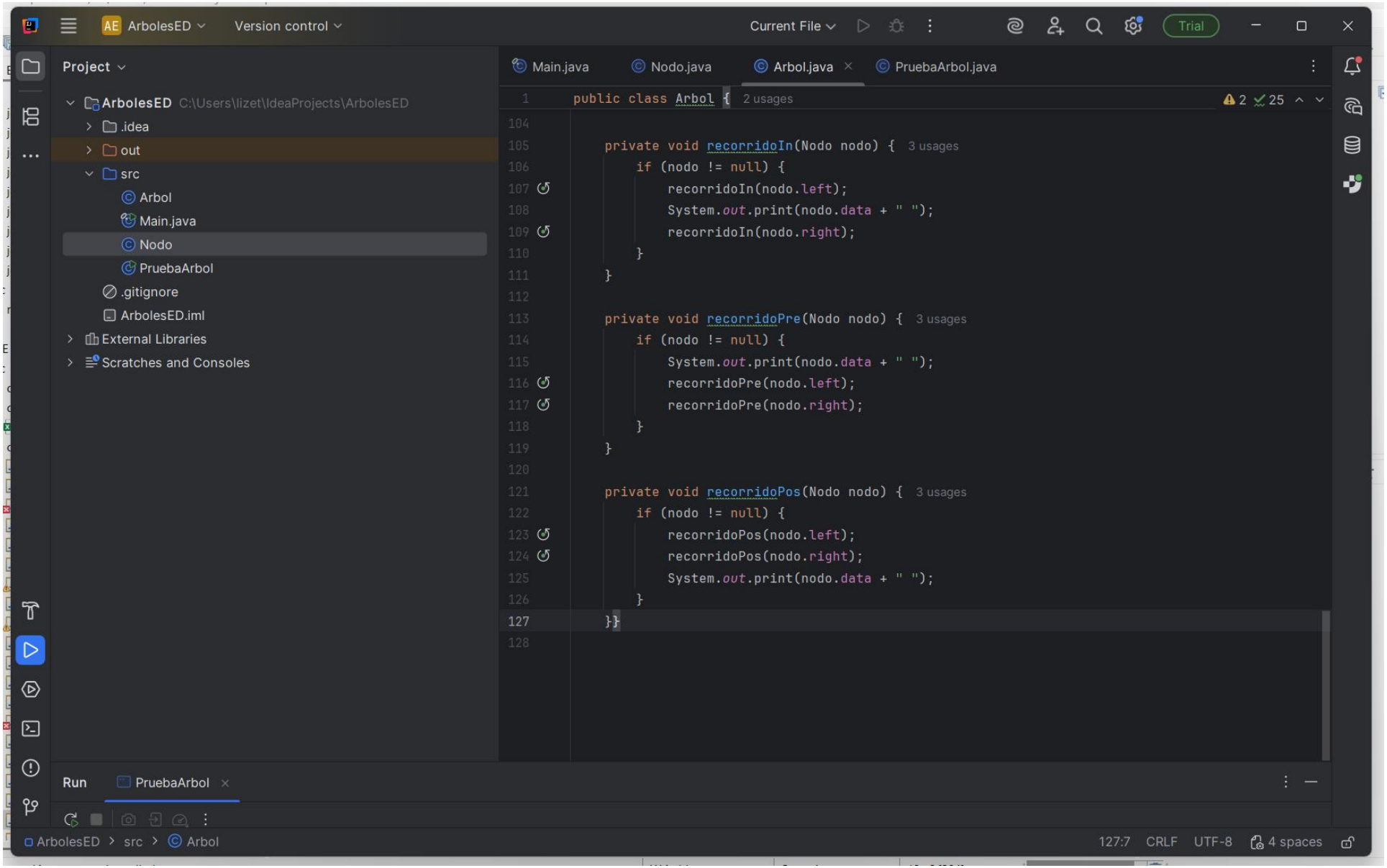
Clase Árbol


ArbolesED
Version control
Current File
Trial
Project
ArbolesED C:\Users\lizet\IdeaProjects\ArbolesED
.idea
out
src
Arbol
Main.java
Nodo
PruebaArbol
.gitignore
ArbolesED.iml
External Libraries
Scratches and Consoles
Main.java
Nodo.java
Arbol.java
PruebaArbol.java
public class Arbol { 2 usages
private Nodo root; 10 usages
public Arbol() { 1 usage
this.root = null;
}
// Insertar un nodo
public void insert(String value) { 3 usages
root = insertRecursive(root, value);
}
private Nodo insertRecursive(Nodo nodo, String value) { 3 usages
if (nodo == null) {
return new Nodo(value);
}
if (value.compareTo(nodo.data) < 0) {
nodo.left = insertRecursive(nodo.left, value);
} else if (value.compareTo(nodo.data) > 0) {
nodo.right = insertRecursive(nodo.right, value);
}
return nodo;
}
// Buscar un nodo
public Nodo search(String value) { no usages
return searchRecursive(root, value);
}
private Nodo searchRecursive(Nodo nodo, String value) { 3 usages
Run
PruebaArbol
ArbolesED > src > Arbol
127:7
CRLF
UTF-8
4 spaces

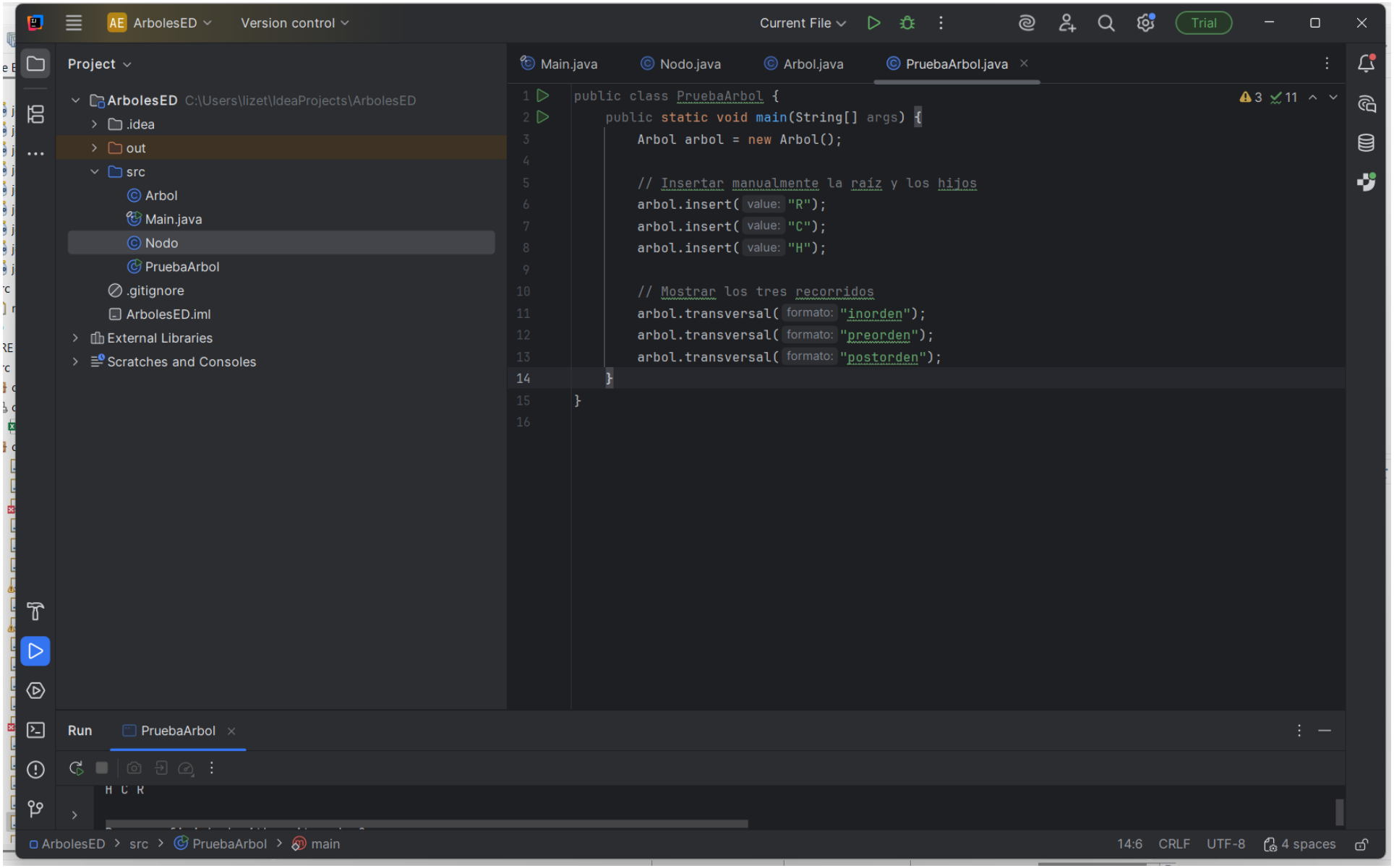

ArbolesED
Version control
Current File
Trial
Project
ArbolesED C:\Users\lizet\IdeaProjects\ArbolesED
.idea
out
src
Arbol
Main.java
Nodo
PruebaArbol
.gitignore
ArbolesED.iml
External Libraries
Scratches and Consoles
Main.java
Nodo.java
Arbol.java
PruebaArbol.java
public class Arbol { 2 usages
2 25
}
private Nodo searchRecursive(Nodo nodo, String value) { 3 usages
if (nodo == null || nodo.data.equals(value)) {
return nodo;
}
if (value.compareTo(nodo.data) < 0) {
return searchRecursive(nodo.left, value);
} else {
return searchRecursive(nod
}
}
Nodo nodo
ArbolesED
// Eliminar un nodo
public void remove(String value) { no usages
root = removeRecursive(root, value);
}
private Nodo removeRecursive(Nodo nodo, String value) { 4 usages
if (nodo == null) {
return null;
}
if (value.compareTo(nodo.data) < 0) {
nodo.left = removeRecursive(nodo.left, value);
} else if (value.compareTo(nodo.data) > 0) {
nodo.right = removeRecursive(nodo.right, value);
} else {
// Caso: nodo con 1 o 0 hijos
Run
PruebaArbol
ArbolesED > src > Arbol
127:7
CRLF
UTF-8
4 spaces


ArbolesED
Version control
Current File
Trial
Project
ArbolesED C:\Users\lizet\IdeaProjects\ArbolesED
.idea
out
src
Arbol
Main.java
Nodo
PruebaArbol
.gitignore
ArbolesED.iml
External Libraries
Scratches and Consoles
Main.java
Nodo.java
Arbol.java
PruebaArbol.java
public class Arbol { 2 usages
2 25
private Nodo removeRecursive(Nodo nodo, String value) { 4 usages
} else {
// Caso: nodo con 1 o 0 hijos
if (nodo.left == null) {
return nodo.right;
} else if (nodo.right == null) {
return nodo.left;
}
// Caso: nodo con dos hijos
nodo.data = findMinValue(nodo.right);
nodo.right = removeRecursive(nodo.right, nodo.data);
}
return nodo;
}
private String findMinValue(Nodo nodo) { 1 usage
String minv = nodo.data;
while (nodo.left != null) {
nodo = nodo.left;
minv = nodo.data;
}
return minv;
}
// Recorridos
public void transversal(String formato) { 3 usages
System.out.println("\nRecorrido " + formato.toUpperCase() + ":");
if (root == null) {
System.out.println("El árbol está vacío.");
Run
PruebaArbol
ArbolesED > src > Arbol
127:7
CRLF
UTF-8
4 spaces

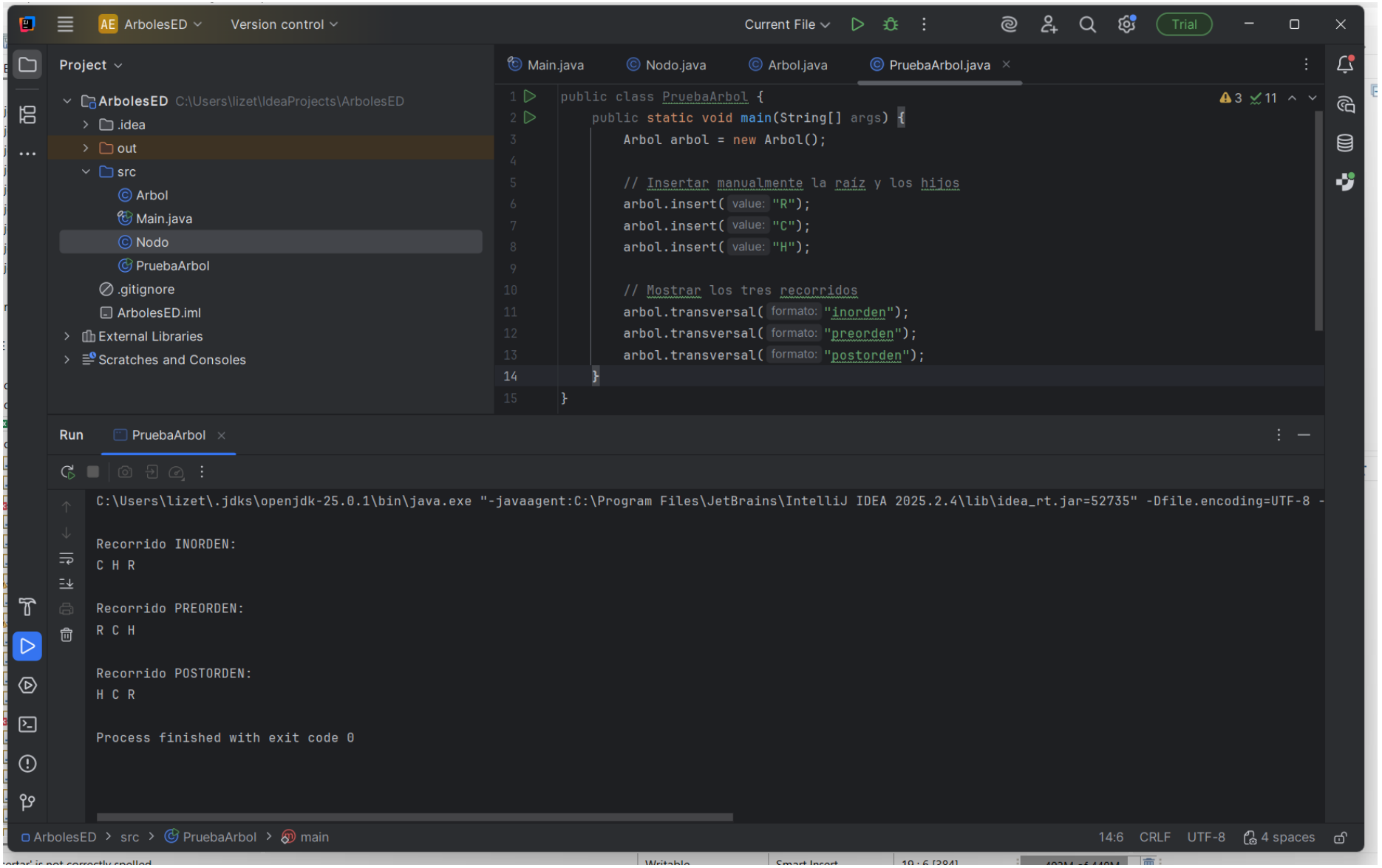

ArbolesED
Version control
Current File
Trial
Project
ArbolesED C:\Users\lizet\IdeaProjects\ArbolesED
.idea
out
src
Arbol
Main.java
Nodo
PruebaArbol
.gitignore
ArbolesED.iml
External Libraries
Scratches and Consoles
Main.java
Nodo.java
Arbol.java
PruebaArbol.java
public class Arbol { 2 usages
public void transversal(String formato) { 3 usages
if (root == null) {
System.out.println("El árbol está vacío.");
return;
}
switch (formato.toLowerCase()) {
case "inorden":
recorridoIn(root);
break;
case "preorden":
recorridoPre(root);
break;
case "postorden":
recorridoPos(root);
break;
default:
System.out.println("Formato inválido (usa inorden, preorden o postorden).");
}
System.out.println();
}
private void recorridoIn(Nodo nodo) { 3 usages
if (nodo != null) {
recorridoIn(nodo.left);
System.out.print(nodo.data + " ");
recorridoIn(nodo.right);
}
}
Run
PruebaArbol
ArbolesED > src > Arbol
127:7
CRLF
UTF-8
4 spaces


ArbolesED
Version control
Current File
Trial
Project
ArbolesED C:\Users\lizet\IdeaProjects\ArbolesED
.idea
out
src
Arbol
Main.java
Nodo
PruebaArbol
.gitignore
ArbolesED.iml
External Libraries
Scratches and Consoles
Main.java
Nodo.java
Arbol.java
PruebaArbol.java
public class Arbol { 2 usages
2 25
private void recorridoIn(Nodo nodo) { 3 usages
if (nodo != null) {
recorridoIn(nodo.left);
System.out.print(nodo.data + " ");
recorridoIn(nodo.right);
}
}
private void recorridoPre(Nodo nodo) { 3 usages
if (nodo != null) {
System.out.print(nodo.data + " ");
recorridoPre(nodo.left);
recorridoPre(nodo.right);
}
}
private void recorridoPos(Nodo nodo) { 3 usages
if (nodo != null) {
recorridoPos(nodo.left);
recorridoPos(nodo.right);
System.out.print(nodo.data + " ");
}
}}
Run
PruebaArbol
ArbolesED > src > Arbol
127:7
CRLF
UTF-8
4 spaces

Clase Prueba



Consola



GitHub: https://github.com/lizz-mond35/Estructura-de-Datos.git