Ejercicios Árboles Binarios

Nombres: Kevin Sierra Rendón, Juan David Quiroga Dimate

ID: 974216

ID: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Ejercicio 2

class nodo {  
 int valor; // Almacena el valor del nodo (un número)  
 nodo izquierdo, derecho; // Referencias a los subárboles izquierdo y derecho  
  
 public nodo(int valor) {  
 this.valor = valor; // Asigna el valor al nodo  
 }  
}  
  
class arbol {  
 nodo raiz; // Punto de partida del árbol  
  
 public void agregar(int valor) {  
 raiz = agregarrec(raiz, valor); // Llama al método recursivo para agregar  
 }  
  
 private nodo agregarrec(nodo actual, int valor) {  
 if (actual == null) return new nodo(valor); // Crea un nuevo nodo si no hay nada  
 if (valor < actual.valor) {  
 actual.izquierdo = agregarrec(actual.izquierdo, valor); // Agrega a la izquierda si es menor  
 } else if (valor > actual.valor) {  
 actual.derecho = agregarrec(actual.derecho, valor); // Agrega a la derecha si es mayor  
 }  
 return actual; // Devuelve el nodo actual si el valor ya existe  
 }  
}  
  
public class Main {  
 public static void main(String[] args) {  
 arbol ar = new arbol(); // Crea una nueva instancia del árbol  
 ar.agregar(50); // Agrega el primer valor  
 ar.agregar(30); // Agrega otro valor  
 ar.agregar(70); // Agrega un tercer valor  
  
 System.out.println("Árbol creado con los nodos: 50, 30, 70");  
 }  
}

insertar la captura de pantalla

# Ejercicio 3

class nodo {  
 int valor; // Almacena el valor del nodo (un número)  
 nodo izquierdo, derecho; // Referencias a los subárboles izquierdo y derecho  
  
 public nodo(int valor) {  
 this.valor = valor; // Asigna el valor al nodo  
 }  
}  
  
class arbol {  
 nodo raiz; // Punto de partida del árbol  
  
 public void agregar(int valor) {  
 raiz = agregarrec(raiz, valor); // Llama al método recursivo para agregar  
 }  
  
 private nodo agregarrec(nodo actual, int valor) {  
 if (actual == null) return new nodo(valor); // Crea un nuevo nodo si no hay nada  
 if (valor < actual.valor) {  
 actual.izquierdo = agregarrec(actual.izquierdo, valor); // Agrega a la izquierda si es menor  
 } else if (valor > actual.valor) {  
 actual.derecho = agregarrec(actual.derecho, valor); // Agrega a la derecha si es mayor  
 }  
 return actual; // Devuelve el nodo actual si el valor ya existe  
 }  
  
 public void mostrarinorden() {  
 inorden(raiz); // Inicia el recorrido inorden desde la raíz  
 }  
  
 private void inorden(nodo nodo) {  
 if (nodo != null) { // Verifica si el nodo existe  
 inorden(nodo.izquierdo); // Recorre el subárbol izquierdo primero  
 System.out.println(nodo.valor); // Imprime el valor del nodo actual  
 inorden(nodo.derecho); // Recorre el subárbol derecho después  
 }  
 }  
}  
  
public class Main {  
 public static void main(String[] args) {  
 arbol ar = new arbol(); // Crea una nueva instancia del árbol  
 ar.agregar(50); // Agrega el primer valor  
 ar.agregar(30); // Agrega otro valor  
 ar.agregar(70); // Agrega un tercer valor  
  
 System.out.println("Recorrido inorden del árbol:");  
 ar.mostrarinorden(); // Muestra los valores en orden  
 }  
}

insertar la captura de pantalla

# Ejercicio 4

class nodo {  
 int valor; // Almacena el valor del nodo (un número)  
 nodo izquierdo, derecho; // Referencias a los subárboles izquierdo y derecho  
  
 public nodo(int valor) {  
 this.valor = valor; // Asigna el valor al nodo  
 }  
}  
  
class arbol {  
 nodo raiz; // Punto de partida del árbol  
  
 public void agregar(int valor) {  
 raiz = agregarrec(raiz, valor); // Llama al método recursivo para agregar  
 }  
  
 private nodo agregarrec(nodo actual, int valor) {  
 if (actual == null) return new nodo(valor); // Crea un nuevo nodo si no hay nada  
 if (valor < actual.valor) {  
 actual.izquierdo = agregarrec(actual.izquierdo, valor); // Agrega a la izquierda si es menor  
 } else if (valor > actual.valor) {  
 actual.derecho = agregarrec(actual.derecho, valor); // Agrega a la derecha si es mayor  
 }  
 return actual; // Devuelve el nodo actual si el valor ya existe  
 }  
  
 public void agregarVarios(int[] valores) {  
 for (int valor : valores) { // Recorre cada valor en el arreglo  
 agregar(valor); // Agrega cada valor al árbol  
 }  
 }  
  
 public void mostrarinorden() {  
 inorden(raiz); // Inicia el recorrido inorden desde la raíz  
 }  
  
 private void inorden(nodo nodo) {  
 if (nodo != null) { // Verifica si el nodo existe  
 inorden(nodo.izquierdo); // Recorre el subárbol izquierdo primero  
 System.out.println(nodo.valor); // Imprime el valor del nodo actual  
 inorden(nodo.derecho); // Recorre el subárbol derecho después  
 }  
 }  
}  
  
public class Main {  
 public static void main(String[] args) {  
 arbol ar = new arbol(); // Crea una nueva instancia del árbol  
 int[] valores = {50, 30, 70, 20, 40, 60, 80}; // Arreglo con los valores a insertar  
 ar.agregarVarios(valores); // Inserta todos los valores  
  
 System.out.println("Recorrido inorden del árbol:");  
 ar.mostrarinorden(); // Muestra los valores en orden  
 }  
}

insertar la captura de pantalla

# Ejercicio 5

class nodo {  
 int valor; // Almacena el valor del nodo (un número)  
 nodo izquierdo, derecho; // Referencias a los subárboles izquierdo y derecho  
  
 public nodo(int valor) {  
 this.valor = valor; // Asigna el valor al nodo  
 }  
}  
  
class arbol {  
 nodo raiz; // Punto de partida del árbol  
  
 public void agregar(int valor) {  
 raiz = agregarrec(raiz, valor); // Llama al método recursivo para agregar  
 }  
  
 private nodo agregarrec(nodo actual, int valor) {  
 if (actual == null) return new nodo(valor); // Crea un nuevo nodo si no hay nada  
 if (valor < actual.valor) {  
 actual.izquierdo = agregarrec(actual.izquierdo, valor); // Agrega a la izquierda si es menor  
 } else if (valor > actual.valor) {  
 actual.derecho = agregarrec(actual.derecho, valor); // Agrega a la derecha si es mayor  
 }  
 return actual; // Devuelve el nodo actual si el valor ya existe  
 }  
  
 public void agregarVarios(int[] valores) {  
 for (int valor : valores) { // Recorre cada valor en el arreglo  
 agregar(valor); // Agrega cada valor al árbol  
 }  
 }  
  
 public void mostrarinorden() {  
 inorden(raiz); // Inicia el recorrido inorden desde la raíz  
 }  
  
 private void inorden(nodo nodo) {  
 if (nodo != null) { // Verifica si el nodo existe  
 inorden(nodo.izquierdo); // Recorre el subárbol izquierdo primero  
 System.out.println(nodo.valor); // Imprime el valor del nodo actual  
 inorden(nodo.derecho); // Recorre el subárbol derecho después  
 }  
 }  
  
 public void mostrarpreorden() {  
 preorden(raiz); // Inicia el recorrido preorden desde la raíz  
 }  
  
 private void preorden(nodo nodo) {  
 if (nodo != null) { // Verifica si el nodo existe  
 System.out.println(nodo.valor); // Imprime el valor del nodo actual primero  
 preorden(nodo.izquierdo); // Recorre el subárbol izquierdo después  
 preorden(nodo.derecho); // Recorre el subárbol derecho al final  
 }  
 }  
  
 public void mostrarpostorden() {  
 postorden(raiz); // Inicia el recorrido postorden desde la raíz  
 }  
  
 private void postorden(nodo nodo) {  
 if (nodo != null) { // Verifica si el nodo existe  
 postorden(nodo.izquierdo); // Recorre el subárbol izquierdo primero  
 postorden(nodo.derecho); // Recorre el subárbol derecho después  
 System.out.println(nodo.valor); // Imprime el valor del nodo actual al final  
 }  
 }  
}  
  
public class Main {  
 public static void main(String[] args) {  
 arbol ar = new arbol(); // Crea una nueva instancia del árbol  
 int[] valores = {50, 30, 70, 20, 40, 60, 80}; // Arreglo con los valores a insertar  
 ar.agregarVarios(valores); // Inserta todos los valores  
  
 System.out.println("Recorrido inorden:");  
 ar.mostrarinorden(); // Muestra los valores en inorden  
 System.out.println("Recorrido preorden:");  
 ar.mostrarpreorden(); // Muestra los valores en preorden  
 System.out.println("Recorrido postorden:");  
 ar.mostrarpostorden(); // Muestra los valores en postorden  
 }  
}

insertar la captura de pantalla

# Ejercicio 6

class nodo {  
 int valor; // Almacena el valor del nodo (un número)  
 nodo izquierdo, derecho; // Referencias a los subárboles izquierdo y derecho  
  
 public nodo(int valor) {  
 this.valor = valor; // Asigna el valor al nodo  
 }  
}  
  
class arbol {  
 nodo raiz; // Punto de partida del árbol  
  
 public void agregar(int valor) {  
 raiz = agregarrec(raiz, valor); // Llama al método recursivo para agregar  
 }  
  
 private nodo agregarrec(nodo actual, int valor) {  
 if (actual == null) return new nodo(valor); // Crea un nuevo nodo si no hay nada  
 if (valor < actual.valor) {  
 actual.izquierdo = agregarrec(actual.izquierdo, valor); // Agrega a la izquierda si es menor  
 } else if (valor > actual.valor) {  
 actual.derecho = agregarrec(actual.derecho, valor); // Agrega a la derecha si es mayor  
 }  
 return actual; // Devuelve el nodo actual si el valor ya existe  
 }  
  
 public void agregarVarios(int[] valores) {  
 for (int valor : valores) { // Recorre cada valor en el arreglo  
 agregar(valor); // Agrega cada valor al árbol  
 }  
 }  
  
 public boolean buscar(int valor) {  
 return buscarrec(raiz, valor); // Inicia la búsqueda desde la raíz  
 }  
  
 private boolean buscarrec(nodo actual, int valor) {  
 if (actual == null) return false; // Retorna false si no se encuentra  
 if (valor == actual.valor) return true; // Retorna true si se encuentra el valor  
 return valor < actual.valor ?  
 buscarrec(actual.izquierdo, valor) : // Busca a la izquierda si es menor  
 buscarrec(actual.derecho, valor); // Busca a la derecha si es mayor  
 }  
}  
  
public class Main {  
 public static void main(String[] args) {  
 arbol ar = new arbol(); // Crea una nueva instancia del árbol  
 int[] valores = {50, 30, 70, 20, 40, 60, 80}; // Arreglo con los valores a insertar  
 ar.agregarVarios(valores); // Inserta todos los valores  
  
 System.out.println("¿Existe 40? " + ar.buscar(40)); // Debería imprimir true  
 System.out.println("¿Existe 90? " + ar.buscar(90)); // Debería imprimir false  
 }  
}

insertar la captura de pantalla