|  |  |  |
| --- | --- | --- |
| A picture containing company name  Description automatically generated | Nombres: | |
| Fecha: 21 de octubre de 2021 | Ciclo: 4 |
| **Estructura de Datos** | |
| **Quiz 12** | |

1. (20%) Complete la clase **Votante**, que representa la información de una persona que votó en unas elecciones. Como puede notar, los objetos de esta clase poseen: nombre, cédula, si votó en blanco o no y el nombre del candidato por el que votó. Los objetos de esta clase son comparables, y el criterio de comparación que usaremos será la cédula del votante.
2. class Votante: Comparable<Votante> {  
    // Atributos  
    var cédula: Int = 0  
    var nombre: String = ""  
    var votoEnBlanco: Boolean = false  
    var votoCandidato: String = ""  
    // Constructores:  
    constructor()  
     
    constructor(cédula: Int, nombre: String, votoEnBlanco: Boolean, votoCandidato: String) {  
    this.cédula = cédula  
    this.nombre = nombre  
    this.votoEnBlanco = votoEnBlanco  
    this.votoCandidato = votoCandidato  
    }  
     
    override fun compareTo(other: Votante): Int {  
    return *compareValues*(this.cédula,other.cédula)  
    }  
     
     
   }
3. (50%) Escriba la clase **Votación**, que representa una elección en nuestro país. El único atributo de esta clase será un árbol binario de búsqueda (implementado como un árbol AVL) de votantes (la clase desarrollada en el punto anterior). Los cuatro únicos métodos a implementar son: un método que retorne el número de votantes que hay en el árbol, el número de personas que votaron en blanco y el número de personas o votantes que votaron por un candidato dado.

class Votación {  
 private val votantes: IBinarySearchTree<Votante> = *TAVL*()  
  
 constructor()  
  
 fun cantidadVotantes(): Int {  
 fun numVotantes(arbol: IBinarySearchTree<Votante>): Int {  
 var contador:Int = 0  
 if (arbol.isEmpty){  
 return contador  
 }  
 else{  
 contador= numVotantes(arbol.left) + numVotantes(arbol.right)  
 if (arbol.root != null){  
 contador++  
 }  
 return contador  
 }  
 }  
  
  
 // Usamos la función anterior  
 return numVotantes(votantes)  
 }  
 fun cantidadVotantesBlanco(): Int {  
 fun numVotantesBlanco(arbol: IBinarySearchTree<Votante>): Int {  
 var contador:Int = 0  
 if (arbol.isEmpty){  
 return contador  
 }  
 else{  
 contador= numVotantesBlanco(arbol.left) + numVotantesBlanco(arbol.right)  
 if (arbol.root != null && arbol.root.votoEnBlanco){  
 contador++  
 }  
 return contador  
 }  
 }  
 return numVotantesBlanco(votantes)  
  
 }  
  
 fun cantidadVotosCandidato (candidato:String):Int{  
 fun numVotantesCandidato(arbol: IBinarySearchTree<Votante>,nomCandidato:String):Int{  
 var contador:Int = 0  
 if (arbol.isEmpty){  
 return 0  
 }  
 else{  
 var contador = numVotantesCandidato(arbol.left,nomCandidato) + numVotantesCandidato(arbol.right,nomCandidato)  
 if (arbol.root.votoCandidato == candidato){  
 contador++  
  
 }  
 return contador  
 }  
 }  
 return numVotantesCandidato(votantes,candidato)  
 }  
  
}

1. (30%) En nuestro país, si el número de votos en blanco es más de la mitad de todos los votos que hubo en una elección, ésta deberá repetirse. Suponiendo que solo se presentaron dos candidatos: “ZULUAGA” y “SANTOS”, escriba una función externa, que reciba un objeto de la clase votación anteriormente desarrollada y que retorne: “ELECCIÓN INVÁLIDA” (si debe repetirse), “EMPATE” (si hubo un empate entre ambos candidatos), o el nombre de candidatos que más votos tuvo. **OJO: esta función no es recursiva**.

fun decidirVotacion(votacionPresidente:Votación):String{  
 var respuesta:String = ""  
 val candidato\_1= "ZULUAGA"  
 val candidato\_2= "SANTOS"  
 if ((votacionPresidente.cantidadVotantesBlanco()) > (votacionPresidente.cantidadVotantes().toDouble() / 2) ){  
  
 respuesta = "ELECCION INVALIDA"  
 }  
  
 else if (votacionPresidente.cantidadVotosCandidato(candidato\_1) == votacionPresidente.cantidadVotosCandidato(candidato\_2) ){  
 respuesta= "EMPATE"  
 }  
  
 else if (votacionPresidente.cantidadVotosCandidato(candidato\_1) > votacionPresidente.cantidadVotosCandidato(candidato\_2)){  
 respuesta = candidato\_1  
 }  
  
 else{  
 respuesta = candidato\_2  
 }  
  
 return respuesta  
}