

A dark blue vertical bar on the left side of the page. A blue arrow points to the right from the bar, containing the date.

7/2/2022

Manual Tecnico

EDD Proyecto Fase 2

Several thin, curved lines in dark blue and light grey originate from the bottom left and curve upwards and to the right.

Alberto Gabriel Reyes Ning, 201612174
ESTRUCTURA DE DATOS A

Table of Contents

Objetivos del Sistema	2
Especificación Técnica	2
Requisitos de Hardware	2
Requisitos de Software	2
Sistema Operativo	2
Lenguajes de Programación e IDE	2
Librerías Usados	2
Lógica del Programa	3
Clases Utilizadas	3
ABB.js.....	3
Constructor.....	3
Funciones	3
Variables.....	4
AVL_Pelicula.js	5
Constructor.....	5
Funciones	5
Variables.....	6
List_Users.js.....	7
Constructor.....	7
Funciones	7
Variables.....	8
TH_Categorias.js.....	9
Constructor.....	9
Funciones	9
Variables.....	11

Objetivos del Sistema

Desarrollar una aplicación utilizando las estructuras de datos y sus algoritmos explicados en clase, de tal forma que pueda simular los diferentes procesos que se dan en la empresa MovieCats. Dicha aplicación deberá ser capaz de representar las estructuras de forma visual, mediante la utilización de bibliotecas soportadas.

Desea realizar una simulación de todo el proceso que con lleva alquilar películas en una tienda virtual, es decir, desde que los clientes realizan la solicitud de alquilar y guardar sus comentarios y mostrar las películas y actores.

Especificación Técnica

Requisitos de Hardware

- Memoria RAM: 4 GBs o Superior
- Procesador: Intel Celeron o Superior

Requisitos de Software

Sistema Operativo

- Windows 8 o Superior

Lenguajes de Programación e IDE

- Java Versión 8 Update 271
- NetBeans 8.2

Librerías Usados

- JsonSimple 1.1
- GraphViz

Lógica del Programa

Clases Utilizadas

ABB.js

Constructor

Node_ABB()

```
class Node_ABB {
  constructor(dni, nombre_Actor, correo, descripcion, left = null, right = null) {
    this.dni = dni;
    this.nombre_Actor = nombre_Actor;
    this.correo = correo;
    this.descripcion = descripcion;
    this.left = left;
    this.right = right;
  }
}
```

Funciones

Insert()

```
insert(dni, nombre_Actor, correo, descripcion) {
  let node = new Node_ABB(dni, nombre_Actor, correo, descripcion);
  if (this.raiz === null) {
    this.raiz = node;
  } else {
    let temp = this.raiz;
    let status = false;
    while (temp && status === false) {
      if (temp.dni === dni) {
        status = true;
        break;
      }
      if (dni > temp.dni) {
        if (temp.right === null) {
          temp.right = node;
          status = true;
        } else {
          temp = temp.right;
        }
      } else if (dni < temp.dni) {
        if (temp.left === null) {
          temp.left = node;
          status = true;
        } else {
          temp = temp.left;
        }
      }
    }
  }
  this.size++;
}
```

DisplayIO()

```
displayIO(temp = this.raiz, go) {
  if (temp === null) {
    return;
  }
  this.displayIO(temp.left, go);
  console.log(temp.nombre_Actor + " ");
  this.displayIO(temp.right, go);
}
```

DisplayPreO()

```
displayPreO(temp = this.raiz, go) {  
  if (temp === null) {  
    return;  
  }  
  console.log(temp.nombre_Actor + " ");  
  this.displayPreO(temp.left, go);  
  this.displayPreO(temp.right, go);  
}
```

DisplayPostO()

```
displayPostO(temp = this.raiz, go) {  
  if (temp === null) {  
    return;  
  }  
  this.displayPostO(temp.left, go);  
  this.displayPostO(temp.right, go);  
  console.log(temp.nombre_Actor + " ");  
}
```

graficadora()

```
graficadora(temp) { // modify temp.data to temp.dni and adjust other values  
  let cadena = "";  
  if (temp === null) {  
    return cadena;  
  }  
  cadena = "nodo" + temp.dni + " [label = \"" + temp.nombre_Actor + "\"]; \n"; // dni: " + temp.dni + "\"\n  
  
  if (temp.left != null) { //:C0  
    cadena = cadena + this.graficadora(temp.left) + "nodo" + temp.dni + " -> nodo" + temp.left.dni + "\n";  
  }  
  if (temp.right != null) { //:C1  
    cadena = cadena + this.graficadora(temp.right) + "nodo" + temp.dni + " -> nodo" + temp.right.dni + "\n";  
  }  
  return cadena;  
}
```

Graph()

```
graph(path) {  
  let str = "";  
  str = "digraph G{\nlabel=\" Actores \";\ngraph[size=\"10,8\"]; \n";  
  if (this.raiz) {  
    let current = this.raiz;  
    let counter = 0;  
    str += this.graficadora(this.raiz);  
  }  
  str += '}'  
  //console.log(str);  
  d3.select(path).graphviz().width(1000).height(1000).renderDot(str);  
}
```

Variables

Dni, nombre_Actor, correo, descripción, left, right

AVL_Pelicula.js

Constructor

Node_AVL()

```
class Node_AVL {  
  constructor(id_Pelicula = null, nombre_Pelicula, descripcion, puntuacion_Star, precio_Q) {  
    this.id_Pelicula = id_Pelicula;  
    this.nombre_Pelicula = nombre_Pelicula;  
    this.descripcion = descripcion;  
    this.puntuacion_Star = puntuacion_Star;  
    this.precio_Q = precio_Q;  
    this.height = 1;  
    this.left = null;  
    this.right = null;  
  }  
}
```

Funciones

Height(N)

```
height(N) {  
  if (N === null) {  
    return 0;  
  }  
  return N.height;  
}
```

RightRotate()

```
rightRotate(y) {  
  let x = y.left;  
  let T2 = x.right;  
  x.right = y;  
  y.left = T2;  
  y.height = Math.max(this.height(y.left), this.height(y.right)) + 1;  
  x.height = Math.max(this.height(x.left), this.height(x.right)) + 1;  
  return x;  
}
```

LeftRotate()

```
leftRotate(x) {  
  let y = x.right;  
  let T2 = y.left;  
  y.left = x;  
  x.right = T2;  
  x.height = Math.max(this.height(x.left), this.height(x.right)) + 1;  
  y.height = Math.max(this.height(y.left), this.height(y.right)) + 1;  
  return y;  
}
```

getBalanceFactor()

```
getBalanceFactor(N) {  
  if (N == null) {  
    return 0;  
  }  
  
  return this.height(N.left) - this.height(N.right);  
}
```

InsertNodeHelper()

```

insertNodeHelper(node, id_Pelicula, nombre_Pelicula, descripcion, puntuacion_Star, precio_Q) {
  if (node == null) {
    return (new Node_AVL(id_Pelicula, nombre_Pelicula, descripcion, puntuacion_Star, precio_Q));
  }

  if (id_Pelicula < node.id_Pelicula) {
    node.left = this.insertNodeHelper(node.left, id_Pelicula, nombre_Pelicula, descripcion, puntuacion_Star, precio_Q);
  } else if (id_Pelicula > node.id_Pelicula) {
    node.right = this.insertNodeHelper(node.right, id_Pelicula, nombre_Pelicula, descripcion, puntuacion_Star, precio_Q);
  } else {
    return node;
  }

  node.height = 1 + Math.max(this.height(node.left), this.height(node.right));
  let balanceFactor = this.getBalanceFactor(node);

  if (balanceFactor > 1) {
    if (id_Pelicula < node.left.id_Pelicula) {
      return this.rightRotate(node);
    } else if (id_Pelicula > node.left.id_Pelicula) {
      node.left = this.leftRotate(node.left);
      return this.rightRotate(node);
    }
  }

  if (balanceFactor < -1) {
    if (id_Pelicula > node.right.id_Pelicula) {
      return this.leftRotate(node);
    } else if (id_Pelicula < node.right.id_Pelicula) {
      node.right = this.rightRotate(node.right);
      return this.leftRotate(node);
    }
  }

  return node;
}

```

InsertNode()

```

insertNode(id_Pelicula, nombre_Pelicula, descripcion, puntuacion_Star, precio_Q) {
  this.root = this.insertNodeHelper(this.root, id_Pelicula, nombre_Pelicula, descripcion, puntuacion_Star, precio_Q);
}

```

NodeWithMinimumValue()

```

nodeWithMimumValue(node) {
  let current = node;
  while (current.left !== null) {
    current = current.left;
  }
  return current;
}

```

Graph()

```

graficadora(temp) {
  let cadena = "";
  if (temp === null) {
    return cadena;
  }
  cadena = "nodo" + temp.id_Pelicula + " [label = \"" + temp.id_Pelicula + "\"]; \n"; // DPI: " + temp.dpi + " \n\n";

  if (temp.left != null) { //C0
    cadena = cadena + this.graficadora(temp.left) + "nodo" + temp.id_Pelicula + " -> nodo" + temp.left.id_Pelicula + "\n";
  }
  if (temp.right != null) { //C1
    cadena = cadena + this.graficadora(temp.right) + "nodo" + temp.id_Pelicula + " -> nodo" + temp.right.id_Pelicula + "\n";
  }
  return cadena;
}

graph(path) {
  let str = "";
  str = "digraph G[\nlabel=\"" + Peliculas + "\";\ngraph[\"size=\"10,0\"; \n";
  if (this.root) {
    let current = this.root;
    let counter = 0;
    str += this.graficadora(this.root);
  }
  str += "']";
  //console.log(str);
  d3.select(path).graphviz().width(1000).height(1000).renderDot(str);
}

```

Variables

Id_Pelicula, nombre_Pelicula, descripción, puntuacion_Star, precio_Q

Constructor

Node

```
class Node {
    constructor(data, nombre_Completo, nombre_Usuario, correo, contrasena, telefono, next = null, prev = null) {
        this.next = next;
        this.prev = prev;
        this.purchase = null;
        this.data = data;
        this.nombre_Completo = nombre_Completo;
        this.nombre_Usuario = nombre_Usuario;
        this.contrasena = contrasena;
        this.telefono = telefono;
        this.correo = correo;
    }
}
```

Funciones

Insert()

```
insert(data, nombre_Completo, nombre_Usuario, correo, contrasena, telefono) {
  if (!this.head) {
    this.insertFirst(data, nombre_Completo, nombre_Usuario, correo, contrasena, telefono);
  } else {
    this.insertLast(data, nombre_Completo, nombre_Usuario, correo, contrasena, telefono);
  }
}
```

InsertFirst()

```
}
    fpi2:2j36++!
    fpi2:pe9q = uem moqe(q9f9' uompre~cwmbyfco' uompre~he9nyto' colleo' couf9azew9' f6t6foou' fpi2:pe9q):
    ju264f9l2f(q9f9' uompre~cwmbyfco' uompre~he9nyto' colleo' couf9azew9' f6t6foou) {
    u1 tu264f9 } e2zf uome
```

InsertLast()

```

insertLast(data, nombre_Completo, nombre_Usuario, correo, contrasena, telefono) {
    let node = new Node(data, nombre_Completo, nombre_Usuario, correo, contrasena, telefono);
    // If empty, make head
    if (!this.head) {
        this.head = node;
    } else if (!this.fin) {
        this.fin = node;
        this.head.next = this.fin;
        this.head.prev = this.fin;
        this.fin.prev = this.head;
        this.fin.next = this.head;
    } else {
        let current = this.fin;
        current.next = node;
        this.fin = node;
        this.fin.next = this.head;
        this.fin.prev = current;
        this.head.prev = this.fin;
    }
    this.size++;
}

```

Graph()


```

graph(path) {
  let str = "";
  str = "digraph G{\nlabel=\n Usuarios \n;\ngraph[size=\n10,0\"]; \nnode [shape=circle];\n rankdir=LR\n";
  if (this.head) {
    let current = this.head;
    let counter = 0;
    while (current) {
      if (current == this.head && counter != 0) {
        break;
      }
      str += "node" + counter;
      str += "[label=" + current.data + "\n" + current.nombre_Completo + "\n" + current.nombre_Usuario + "\n" + current.contrasena + "\n" + c
      if (current == this.fin) {
        str += "node" + counter + ";\n";
      } else if (current.next) {
        str += "node" + counter + " -> node" + (counter + 1) + ";\n";
      }
      current = current.next;
      counter++;
    }
  }
  str += "};\n";
  d3.select(path).graphviz().width(1000).height(650).renderDot(str);
}

```

Login()

```

login(user, pass) {
  if (this.head != null) {
    let current = this.head;
    stop = false;
    while (current != null) {
      //console.log(current.nombre_Usuario + " " + current.contrasena);
      if (current == this.head && stop === true) {
        console.log("Incorrect information");
        return false;
      }
      if (current.nombre_Usuario == user && current.contrasena == pass) {
        console.log("True");
        return current;
      }
      stop = true;
      current = current.next;
    }
  }
  console.log("User Not Found");
  return false;
}

```

Variables

Next, prev, data, nombre_Completo, nombre_Usuario, contraseña, teléfono, correo

TH_Categorias.js

Constructor

Hash

```
class Hash { // Lista
  constructor(id_Categoria, company) {
    this.id_Categoria = null;
    this.company = company;
    this.key = id_Categoria % 20;
    this.next = null;
    this.list = null;
  }
}
```

Funciones

getHash()

```
getHash(id_Categoria) {
  let key = id_Categoria % 20;
  return key;
}
```

createList()

```
create_List() {
  for (let i = 0; i < 20; i++) {
    this.insert(i, null);
  }
}
```

Insert()

```
insert(id_Categoria, company) {
  let node = new Hash(id_Categoria, company);

  if (this.head === null) {
    this.head = node;
  } else {
    let temp = this.head;
    while (temp.next !== null) {
      temp = temp.next;
    }
    temp.next = node;
  }
}
```

Insert2()

```

insert2(id_Categoria, company) {
  let node = new Hash(id_Categoria, company);
  let key = this.getHash(id_Categoria, company);
  let temp = this.head;
  node.id_Categoria = id_Categoria;

  while (temp.key != key) {
    temp = temp.next;
  }
  if (temp.company === null) {
    temp.id_Categoria = id_Categoria;
    temp.company = company;
    //console.log(id_Categoria + " inputted into " + temp.key);
  } else {
    if (temp.list === null) {
      if (temp.id_Categoria == id_Categoria) {
        return;
      }
      temp.list = node;
      //console.log(id_Categoria + " inputted into " + temp.key);
    } else {
      let temp_ = temp.list;
      if (temp_.id_Categoria == id_Categoria) {
        return;
      }
      while (temp_.list != null) {
        if (temp_.id_Categoria == id_Categoria) {
          return;
        }
        temp_ = temp_.list;
      }
      temp_.list = node;
      //console.log(id_Categoria + " inputted into " + temp.key);
    }
  }
}

```

Buscar()

```

buscar(id_Categoria) {
  let key = this.getHash(id_Categoria % 20);
  let temp = this.head;

  while (temp.key != key) {
    temp = temp.next;
  }

  if (temp.id_Categoria == id_Categoria) {
    return temp;
  } else {
    let temp_ = temp.list;
    while (temp_.id_Categoria != id_Categoria) {
      temp_ = temp_.list;
    }
    if (temp_.id_Categoria == id_Categoria) {
      return temp_;
    }
  }
}

```

Graph()

```

graficadora() {
  let str = "";
  let temp = this.head;
  let count = 0;
  let rand = 0;
  let rowInfo = "{rank=same;";
  while (temp) {
    str += "Head" + count + " [label=\\"Head: " + temp.key + "\\\"];\\n";
    rowInfo += "Head" + count + ";";
    temp = temp.next;
    count++;
  }
  temp = this.head;
  str += "Head" + 0;
  count = 1;
  temp = temp.next;
  while (temp) {
    str += " -> Head" + count;
    temp = temp.next;
    count++;
  }
  str += "[style=invis];\\n" + rowInfo + ";\\n";

  temp = this.head;
  count = 0;
  rand = 0;
  while (temp) {
    let temp_ = temp.list;
    if (temp.id_Categoria !== null) {
      str += "Head" + count + " -> List" + rand + ";\\n";
      str += "List" + rand + " [label=\\" " + temp.company + "\\\"];\\n";
      rand++;
      while (temp_) {
        str += "List" + rand + " [label=\\" " + temp_.company + "\\\"];\\n";
      }
    }
    temp = temp.next;
    count++;
  }
  return str;
}

graph(path) {
  let str = "digraph structs\\n\\nrankdir=\\LR\\n\\nlabel=\\Categorias\\n\\nnode [shape=box];\\n";
  str += this.graficadora();
  str += ";";
  //console.log(str);
  d3.select(path).graphviz().width(1000).height(1000).renderDot(str);
}

```

Variables

Id_Categoria, company