INFORME DE IMPLEMENTACION: SISTEMA DE GESTIÓN DE TAREAS HASH

💻 ALGORITMOS II

🖐🏻 **ALUMNO**: Torres Juan

📍 INTRODUCCIÓN

En este informe se podrá ver la implementación de tablas de dispersión (hash) para un sistema de gestión de tareas en Java con su respectivo método de resolución de colisiones. El programa permite realizar inserción, eliminación lógica, modificación, listar las tareas según su estado (pendiente, en progreso, finalizado), y dar de alta una tarea eliminada.

🧰 ESTRUCTURA DEL PROGRAMA

CLASES:

- Main: contiene el menu principal y la entrada de usuario.
- Tarea: describe al objeto con sus atributos (nombre, descripción, estado, ID, y si esta habilitada) y contiene constructores, setters y getters.
- TablaDispersión: describe la tabla hash (tamaño de la tabla, número de elementos y factor de carga), posee inserción, eliminación lógica,

modificación, listar las tareas según su estado (pendiente, en progreso, finalizado), y dar de alta una tarea eliminada.

VALIDACIONES IMPLEMENTADAS:

- Verificaciones y validaciones de ID.
- Verificaciones de entradas numéricas y de opciones.
- Prevención de operaciones si no hay tareas activas o eliminadas.

📌 METODOS PRINCIPALES

```
public boolean insertar(Tarea t)
    {
        int posicionInicial = calcularPosicion(t.getId());
        int posicion = posicionInicial;
        int i=0;

        while(tabla[posicion] != null)
        {

            i++;

            //RESUELVE LA COLISION
            posicion = resolverColision(posicionInicial, i);

            //COMPRUEBA SI TABLA HASH ESTA LLENA
            if(i>=m)
            {
                return false;
            }
        }

        //ASIGNA EL OBJETO A INSERTAR EN UNA NUEVA POSICION VACIA
        tabla[posicion] = t;
        numElementos++;
```

```
        factorCarga = (double) numElementos / m;
        if(factorCarga > 0.5)
        {
            System.out.println("FACTOR DE CARGA SUPERA EL 50%. CONVIENE A
        }

        return true;
    }
```

💡 En este método se inserta el objeto con sus atributos a la posición calculada mediante el método de la multiplicación, según el ID ingresado por el usuario.

En el mismo, tambien se invoca el método para resolver una colisión si se produce y comprueba si la tabla hash esta llena, si es así devuelve FALSE.

Tambien, calcula el factor de carga y lanza una advertencia si este supera el 50% y, por ultimo retorna TRUE si se inserto correctamente.

```
public long transformaCadena(String id)
    {
        long d=0;
        int i=0;

        for(i=0;i< Math.min(5, id.length());i++)
        {
            d = d *27 + (int)id.charAt(i);

        }
        if(d<0) d = -d;
        return d;
    }
```

💡 Este método recibe como parámetro de tipo String el ID de la tarea y a través de la forma ASCII transforma la cadena de caracteres en un número, y retorna ese mismo número.

```
public int calcularPosicion(String id)
  {

      double R=0.6180339;
      double valor = transformaCadena(id);

      double total = R*valor;
      double decimal = total - Math.floor(total);

      return (int) (decimal * 101);
  }
```

💡 Este método recibe como parámetro de tipo String el ID de la tarea, y a través del método de la multiplicación calcula la posición donde se va a insertar o donde está el objeto recibido.

```
public int resolverColision(int posicionInicial, int i)
  {
      return(posicionInicial + i*i) % 101;
  }
```

💡 Este método recibe como parámetro la posición del objeto y el índice iterado para aplicar exploración cuadrática.

🤓 CONCLUSIÓN

El sistema permite administrar tareas de manera eficiente, a través del hashing. Se aplica eliminación lógica para no romper la estructura de la tabla hash. Por consiguiente, realizar este ejercicio permitió un buen entendimiento de una tabla hash, manejo de direcciones con el método de la multiplicación, y resolución de colisiones, a través de la exploración cuadrática.