TECMILENIO

Actividad3 estructura de datos

alumna: Citlaly Guadalupe Zeferino Sierra
Profesor:Adalberto Emmanuel Rojas Perea

Primer problema: Serie de Fibonacci recursiva.

Para este problema utilice el Scanner para que pudiera leer los datos,la serie de fibonacci recursiva es una sucesión de números donde cada número es la suma de los dos anteriores.

asi creando primero la public class Fibonaccirecursiva

```java
import java.util.Scanner;

public class Fibonaccirecursiva{

static Scanner entrada = new Scanner(System.in);

static int serieFibonacciRecursivo(int n){
    if( n == 1 || n == 0 )
    return n;
    else
    return serieFibonacciRecursivo(n-1) + serieFibonacciRecursivo(n-2);
    }
```

Haciendo el reporte me percate que puse mal los valores de N,poniendo el uno de primera (Error ya corregido)If nos representa el caso base y en return nos representa la serie recursiva

```java
import java.util.Scanner;

public class Fibonaccirecursiva{

static Scanner entrada = new Scanner(System.in);

static int serieFibonacciRecursivo(int n){
    if( n == 0 || n == 1 )return n;
    else
    return serieFibonacciRecursivo(n-1) + serieFibonacciRecursivo(n-2);
    }
```

agregue para poner la cantidad de términos en este caso hay que poner el 7 para que nos de el enésimo número

```java
public static void main(String[] args) {
    int n;
    System.out.println("Favor ingresar la cantidad de terminos: ");
    n = entrada.nextInt();
    for(int i = 1; i <= n-0; i++)
    System.out.println("Fibonacci: " + (i+1) + " es: " + serieFibonacciRecursivo(i) + "\n");
    }
}
```

EL RESULTADO

```
PS C:\Users\citla\OneDrive\Actividad3> java Fibonaccirecursiva.java
Favor ingresar la cantidad de terminos:
7
Fibonacci: 2 es: 1

Fibonacci: 3 es: 1

Fibonacci: 4 es: 2

Fibonacci: 5 es: 3

Fibonacci: 6 es: 5

Fibonacci: 7 es: 8

Fibonacci: 8 es: 13
```

Para la Suma de subconjuntos tuve que investigar mas de que trataba para poder hacer el código es un conjunto de números enteros y un número objetivo, que determinar si existe un subconjunto cuyos elementos sumen exactamente
primero empecé haciendo un arreglo para los números, puse N para representar número de elementos a considerar y target la suma que

```
public class SubsetSum{
public static boolean subsetSum(int[] arr, int n, int target) {
    if (target == 0) {
        return true;
        }
        if (n == 0) {
            return false;
        }
        if (arr[n - 1] > target) {
            return subsetSum(arr, n - 1, target);
        }
        return subsetSum(arr, n - 1, target) ||
        subsetSum(arr, n - 1, target - arr[n - 1]);
    }
```

en el caso base es representado por el if Si target == 0 ya hemos alcanzado la suma deseada con algún subconjunto, devolvemos true.
Si n == 0 y target != 0 no quedan elementos por considerar, no se puede formar la suma, devolvemos false.

en el arreglo if (arr[n - 1] > target) es para que si el número que es mayor que el valor de target no lo incluirá

```java
Run main | Debug main
public static void main(String[] args) {
    int[] conjunto = {3, 34, 4, 12, 5, 2};
    int objetivo = 6;

    if (subsetSum(conjunto, conjunto.length, objetivo)) {
        System.out.println("Si existe un subconjunto que suma " + objetivo);
    } else {
        System.out.println("No existe subconjunto que sume " + objetivo);
    }
```

Defini un conjunto de números y el objetivo. el resultado en el caso de 1 no tiene quien lo sume

```
PS C:\Users\citla\OneDrive\Actividad3> java SubsetSum.java
Si existe un subconjunto que suma 6
PS C:\Users\citla\OneDrive\Actividad3> java SubsetSum.java
No existe subconjunto que sume 1
```

En el problema del sudoku empecé indicando el tamaño de de N a 9,use printBoard para que pudiera imprimir el tablero en la terminal con las filas y columnas y muestra cada número

```java
public class sudoku {
    static int N = 9;

    private static void printBoard(int[][] board) {
        for (int row = 0; row < N; row++) {
            for (int col = 0; col < N; col++) {
                System.out.print(board[row][col] + " ");
            }
            System.out.println();
        }
    }
```

El método isSafe es para verificar si un número específico se puede colocar en una casilla,lo que hace es revisar la fila y la columna de la casilla no tenga números repetidos

```java
private static boolean isSafe(int[][] board, int row, int col, int num) {
    for (int i = 0; i < N; i++) {
        if (board[row][i] == num) return false;
        if (board[i][col] == num) return false;
    }
    int startRow = row - row % 3;
    int startCol = col - col % 3;
    for (int r = startRow; r < startRow + 3; r++) {
        for (int c = startCol; c < startCol + 3; c++) {
            if (board[r][c] == num) return false;
        }
    }
    return true;
}
```

El propósito del Método findEmpty localizar la primera casilla vacía del tablero, aquella que aún no tiene un número asignado y está marcada con un cero

```java
private static int[] findEmpty(int[][] board) {
    for (int row = 0; row < N; row++) {
        for (int col = 0; col < N; col++) {
            if (board[row][col] == 0) {
                return new int[]{row, col};
            }
        }
    }
    return null;
}
```

Método solveSudoku

Este método implementa el algoritmo de backtracking, que es el corazón de la resolución del Sudoku. Primero llama a findEmpty para encontrar la siguiente casilla vacía. Si no hay ninguna, quiere decir que el tablero está completo, por lo que retorna true para indicar que se ha encontrado una solución. Si encuentra una casilla vacía, intenta colocar en ella los números del 1 al 9. Para cada número, verifica con isSafe si es seguro colocarlo. Si el número es válido, lo asigna temporalmente a la casilla y llama a solveSudoku de manera recursiva para continuar resolviendo el resto del tablero. Si la recursión logra completar el tablero, retorna true y termina. Si no, se da cuenta de que ese número no funciona, lo borra y prueba con el siguiente. Este proceso de probar números y retroceder cuando algo falla es lo que se conoce como backtracking, y se repite hasta encontrar la solución o determinar que el tablero no tiene solución.

```java
private static boolean solveSudoku(int[][] board) {
    int[] emptyPos = findEmpty(board);
    if (emptyPos == null) return true; // tablero resuelto

    int row = emptyPos[0];
    int col = emptyPos[1];

    for (int num = 1; num <= 9; num++) {
        if (isSafe(board, row, col, num)) {
            board[row][col] = num;

            if (solveSudoku(board)) return true;

            board[row][col] = 0; // backtracking
        }
    }
    return false;
}
```

```java
    public static void main(String[] args) {
        int[][] board = {
            {5, 3, 0, 0, 7, 0, 0, 0, 0},
            {6, 0, 0, 1, 9, 5, 0, 0, 0},
            {0, 9, 8, 0, 0, 0, 0, 6, 0},
            {8, 0, 0, 0, 6, 0, 0, 0, 3},
            {4, 0, 0, 8, 0, 3, 0, 0, 1},
            {7, 0, 0, 0, 2, 0, 0, 0, 6},
            {0, 6, 0, 0, 0, 0, 2, 8, 0},
            {0, 0, 0, 4, 1, 9, 0, 0, 5},
            {0, 0, 0, 0, 8, 0, 0, 7, 9}
        };

        if (solveSudoku(board)) {
            System.out.println("Solucion encontrada:");
            printBoard(board);
        } else {
            System.out.println("No tiene solucion el tablero");
        }
    }
}
```

(En esta primera imagen tuve error en el algoritmo y solo me ponia que el tablero no tenia solucion)

```
PS C:\Users\citla\OneDrive\Actividad3> java sudoku.java
No tiene solucion el tablero
PS C:\Users\citla\OneDrive\Actividad3> javac sudoku.java
PS C:\Users\citla\OneDrive\Actividad3> java sudoku.java
No tiene solucion el tablero
PS C:\Users\citla\OneDrive\Actividad3> javac sudoku.java
PS C:\Users\citla\OneDrive\Actividad3> java sudoku.java
No tiene solucion el tablero
```

en esta otra imagen ya me salio bien en la termin

```
PS C:\Users\citla\OneDrive\Actividad3> javac sudoku.java
PS C:\Users\citla\OneDrive\Actividad3> java sudoku.java
Solucion encontrada:
5 3 4 6 7 8 9 1 2
6 7 2 1 9 5 3 4 8
1 9 8 3 4 2 5 6 7
8 5 9 7 6 1 4 2 3
4 2 6 8 5 3 7 9 1
7 1 3 9 2 4 8 5 6
9 6 1 5 3 7 2 8 4
2 8 7 4 1 9 6 3 5
3 4 5 2 8 6 1 7 9
PS C:\Users\citla\OneDrive\Actividad3>
```

al