Informe de Documentación del

Trabajo Práctico N°3: El equipo ideal

Comisión: 02 (Mañana)

Alumnos:

-Trejo Fernando (43.986.607)

-García Matías (41.385.966)

-Gordillo Fabricio (42.948.240)

Objetivos:

Se nos encargó realizar una aplicación con la cual el usuario pueda armar un equipo de trabajo compuesto por personas con diferentes roles y encontrar un conjunto lo más calificado posible de personas que cumpla estos requerimientos, y que no contenga ningún par de personas incompatibles.

Dificultades:

No tuvimos grandes dificultades a la hora de escribir la interfaz o la clase de manejo de equipo, sin embargo tuvimos problemas a la hora de lograr que el Solver nos de el mejor equipo, nos dice que no encuentra una solución adecuada, al mismo tiempo también tuvimos problemas con la clase SwingWorker ya que al no devolver un resultado el Solver tampoco nos devuelve algo algo el Thread.

De acuerdo a lo que pudimos ir debuggeando creemos que el problema esta en la solucion que no esta agregando como corresponde a las personas a la lista.

Implementación:

Clase Main: Clase de la interfaz gráfica de la aplicación. Atributos:

- JFrame frame: Objeto de la interfaz.
- JTextField txtNombre: Objeto de la interfaz
- JTextField txtRating: Objeto de la interfaz.
- JComboBox<String> cbRoles :Objeto de la interfaz
- JButton btnAgregarPersona :Objeto de la interfaz
- JButton btnAgregarRol :Objeto de la interfaz
- JTextField txtNuevoRol:Objeto de la interfaz
- ManejoEquipo manejoEquipo: Determina un manejoEquipo.
- Font fuente : Determina una fuente.
- Color colorFondo: Determina el color del fondo.
- Color colorBotones: Determina el color del objeto boton.
- Color colorBlanco: Deja guardado una variante de color blanco.

- JTextField textFieldCantidad: Objeto de la interfaz
- Simulacion simulación: Determina una Simulacion.
- JTextField textFieldResultado: Objeto de la interfaz
- JProgressBar progressBar: Objeto de la interfaz

Métodos:

void inicializarPanelAgregarRol() : Crea el panel de la primera pantalla para agregar los roles que se necesitan

void inicializarPanelMostrarSolucion(): Crea el panel con la solución

void inicializarPanelAgregarPersona(): Crea el panel para agregar las personas

void agregarPersona(): Agrega persona a arraylist del manejo de equipo y limpia los txt

void agregarRol():Agrega rol a la lista roles y limpia los txt

Clase Solver: Clase con el algoritmo que resuelve

Atributos:

Map<String, Integer> contadorPersonasEnRol(): Objeto Map que contiene personas

List<Persona> solución(): Lista que contiene las personas agregadas a la solución

Metodos:

boolean solve(int indicePersona): Metodo recursivo que hace la funcion de backtracking

boolean esCompatible(Persona persona): Verifica que una persona sea compatible

boolean todosLosRolesEstanCompletos(): Verifica que los roles esten completos para la solucion

boolean seNecesitaPersonaParaRol(String rol): Verifica que se sigan necesitando personas para el rol

Persona encontrarPersonaMenosCalificadaEnRol(String rol): Recorre la lista de la solucion y verifica la persona con menos rating

boolean personaActualEsCompatibleYMasCalificada(Persona personaActual, Persona personaMenosCalificada): Verifica que la persona actual que es mas

void agregarPersonaAEquipo(Persona personaActual, Persona personaMenosCalificada): Elimina la personas menos calificada y agrega a la persona actual

removerPersonaDeEquipo(Persona personaActual): Elimina persona del equipo

boolean sonCompatibles(Persona persona1, Persona persona2): Verifica que dos personas no tengan incompatibilidades entre sí

List<Persona> getSolucion():Contiene la solucion con las personas ideales

Simulacion: Clase que tiene el swingWorker

Atributos:

int _n: Variable int que inicia la barra de progreso

JTextField resultJTextField: Objeto de la interfaz

JProgressBar barraProgreso: Objeto de la interfaz

ManejoEquipo manejoEquipo: Determina un manejoEquipo.

Metodos:

void done(): Verifica si se finalizo o cancelo

Clase ManejoEquipo: Clase que contiene el manejo del equipo

Atributos:

List<Persona> personas: Lista que contiene personas

List<Rol> roles: Lista que contiene roles

List<Incompatibilidad> incompatibilities: Lista que contiene incopatibilidades

Metodos:

void addPersona(Persona person): Agrega personas a la lista personas

void addRol(Rol role) Agrega roles a la lista roles

void addIncompatibilidad(Incompatibilidad incompatibility): Agrega incompatibilidades al array Incompatibilities

void mostrarPersonas() Ejecuta un print de las personas con sus roles y sus rating

void mostrarRoles() Ejecuta un print con los roles y la cantidad que se necesitan

void mostrarIncompatibilidades()Ejecuta un print con las incopatibilidades

List<Persona> getPersonas() Get de la lista personas

List<Rol> getRoles() Get de la lista roles

List<Incompatibilidad> getIncompatibilidades() Get de la lista incopatibilidades