I. Class Main:

1.Requiere importación java.util.Scanner.

2. No posee atributos

3. Metodos:

3.1 public static void main(String[] args){

ValidadorMenu();

}

Este método se ejecuta automaticamente al ejecutar el programa, llama al método ValidadorMenu(). No cuenta con entradas o salidas. Es de tipo public.

3.2 private static void Texto(){

……..

}

Este metodo tiene como salida por pantalla las opciones del menú de Main. No posee entradas. Es de tipo private.

3.3 private static void ValidadorMenu(){

……..

}

Este metodo tiene como entradas la opcion seleccionada posteriormente a que se llame a Texto(). El metodo cuenta con una validación de la opcion seleccionada. No posee salida. Es de tipo private.

3.4 private static void OpcionElegida(int opc){

……..

}

Metodo de tipo private, requiere el parametro opc de tipo int, este parametro proviene de ValidadorMenu(). OpcionElegida(int opc) es llamado dentro de ValidadorMenu()

Se crea un objeto tipo Recetario, a fin de poder llamar a metodos de esa clase.

Este metodo se preocupa de evaluar el valor de opc y dependiendo de eso llamar al repectivo metodo de clase Recetario. AgregarReceta() , MostrarAllRecetas(), VerCantidadRecetas(), QuitarReceta(), FindRecetaIngredientes(), salir.

Este metodo cuenta con una validación de opc.

II. Class Receta

1. Requiere importación de java.util.ArrayList y java.util.Scanner.
2. Posee como atributos a ingredientes, un ArrayList<Ingrediente>, a instrucciones, un ArrayList<Instruccion> y a nombre, un String.
3. Metodos:
   1. Receta(){

….

}

El constructor no necesita parametros, sirve para crear objetos de tipo Receta.

* 1. public void CrearIngredientes(){

…...

}

Este metodo pide al usuario si desea ingresar mas Ingredientes a una receta. Cuenta con las respectivas validaciones. No posee salida. Es public.

* 1. private void IngresarIngredientes(){

……

}

Este metodo pide al usuario ingresar un Ingrediente a la Receta y lo agrega a ingredientes. No posee salida. Es private

* 1. public void MostrarIngredientes(){

…….

}

Este metodo recorre ingredientes y muestra los Ingredientes de la Receta. No posee entradas y es un metodo void de tipo public.

* 1. private void CambiarIngredientes(){

…..

}

Este metodo el pide al usuario que ingrese el nombre de un ingrediente a fin de cambiarlo. Posee las respectivas validaciones. Llama a MostrarIngredientes(). Es de tipo private. No posee entradas ni salidas.

* 1. private int BuscarIngrediente(String ingrediente){

……

}

Este metodo tiene como parametro el String ingrediente, que es necesario para buscar el Ingrediente en ingredientes. El metodo retorna pos, que es la posicion dentro de ingredientes.

* 1. public void CrearInstruccion(){

………

}

Este metodo no tiene parametros, llama a IngresarPaso() y pregunta si el usuario desea ingresar otro paso. Posee las respectivas validaciones. No tiene salidas. Es public.

* 1. private void IngresarPaso(){

…….

}

Este metodo no tiene parametros, pide al usuario un paso (Instruccion) de la Receta y lo añade a instrucciones. No posee salida. Es private.

* 1. public void MostrarInstrucciones(){

……

}

Este metodo recorre instrucciones y muestra las Instrucciones de la Receta. No posee entradas y es un metodo void de tipo public.

* 1. Los atributos cuentan con metodos get y set.

//CONSIDERACIONES: ES NECESARIO LOS GET/SET de los ARRAYLIST

REVISAR LOS ENCAPSULAMIENTOS

REVISAR METODOS BUSCAR, CAMBIARINGREDIENTE

III. Class Recetario

1. Requiere importación de java.util.ArrayList y java.util.Scanner.
2. Posee como atributo a recetas, un ArrayList<Receta>.
3. Metodos:
   1. public void AgregarReceta(){

…….

}

Este metodo no tiene parametros, pide al usuario que ingrese una Receta, con su respectivo nombre, CrearIngredientes() y CrearInstruccion(), posteriormente se añade a recetas. No tiene salida. Es public.

* 1. public void QuitarReceta(){

…….

}

Este metodo no posee parametros, llama a Preguntar() y teniendo esa información eliminar al respectivo elemento en recetas. No tiene salidas. Es publico

* 1. private static int Preguntar(){

…….

}

Este metodo no posee parametros, llama a MostrarAllRecetas() y pide al usuario que ingrediente que quiere eliminar y retorna la posicion ingresada por el usuario. Es static int.

* 1. public static void MostrarAllRecetas(){

……

}

Este metodo recorre recetas y muestra las Recetas. No posee entradas y es un metodo void de tipo public static.

* 1. public void FindRecetaIngredientes(){

…….

}

Este metodo pide al usuario que ingrese un Ingrediente y llama a Buscar() usando a la entrada como parametro de Buscar(). Es public. No posee salida.

* 1. private static void Buscar(String ing){ //REVISAR FUNCIONALIDAD

……

}

NO ENTENDI LO QUE HACE

* 1. public void VerCantidadRecetas(){

int cantidad=recetas.size();

System.out.println("La cantidad de recetas disponibles es: "+ cantidad);

System.out.println("");

}

Este metodo muestra al usuario la cantidad de recetas disponibles en recetas. Es public y no posee entradas ni salidas.

* 1. public void RankearRecetas(){

……

}

EXPLICAR AQUÍ

IV. Class Instruccion

1. No requiere importaciones.
2. Posee como atributo a paso, un String.
3. Metodos:
   1. public Instruccion(){

…..

}

Este metodo se crea objetos de tipo Instrucción e inicia a paso.

* 1. Posee metodos get y set del atributo paso.

V. Class Ingrediente

1. No requiere importaciones.
2. Posee como atributo a nombre, un String.
3. Metodos:
   1. public Ingrediente(){

…..

}

Este metodo se crea objetos de tipo Ingrediente e inicia a nombre.

* 1. Posee metodos get y set del atributo nombre.