Cultivatrón

Se hizo un análisis respecto a lo solicitado en el proyecto, se han creado 8 clases cumpliendo las características básicas solicitadas. Se han implementado el uso de herencias, clases abstractas, interfaces y en general todo lo visto hasta ahora en el curso.

El aplicativo creado simula el comportamiento de procesos agrícolas básicos como asignación de terrenos, creación de cultivos, recolección, sembrado, ataque de pestes y contratación de personal. Se permite al usuario tener un manejo y visión generales de los componentes básicos presentes para la administración adecuada de los mismos. El usuario tiene las opciones de registrar y a su vez el programa archiva la información relacionada con sus terrenos, los cultivos que siembre en ellos, la recolección que haga en dichos cultivos, su producción total (esta se hace según el tipo de cultivo), los empleados contratados, así como también las amenazas que se presentan en sus sembrados.

Al registrar terrenos el usuario tiene la libertad de sembrar en ellos cultivos de los siguientes tipos: banano, papa, sandía, mango y fresa, pero en cada terreno solo se permite un tipo, quiere decir que si en un terreno ya hay un cultivo de papa no puede haber otro de papa, pero sí puede haber de otros tipos. Dichos cultivos tienen ciertas especificaciones de elementos que requieren para poder cultivarse; niveles de nitrógeno, potasio, fósforo y además una adecuada irrigación, estas propiedades deben estar presentes en el terreno donde se desean plantar. El usuario podrá observar dicha información en el programa. En caso de que un terreno sea infértil para cultivar algún tipo de cosecha, se podrá fertilizar e irrigar el terreno para aumentar sus proteínas y así tener una posibilidad de sembrar algo más.

Los empleados que el usuario puede contratar se derivan en campesinos y agrónomos, los cuales a su vez también puede despedir, cabe mencionar que los empleados renuncian de forma aleatoria, haciendo un poco más real la situación que simula el programa. Los campesinos se asignan a un solo terreno y allí se encargan de labores como sembrado y recolección de cultivos. Además, cuando se requiera fertilizar un terreno para que este sea apto para la siembra de todos los tipos de cultivos el campesino será el encargado de llevar a cabo dicha tarea. Por otro lado, los agrónomos, de los cuales habrá uno por cada terreno, se encargan de orientar a los campesinos y de aplicar los pesticidas adecuados para cada amenaza cuando un cultivo esté bajo ataque de alguna, pues existen tres tipos de pestes que atacan a los cultivos: maleza, hongo y plagas, para las cuales se tiene sus respectivos pesticidas. Las plagas atacan cultivos de forma aleatoria y afectan una cantidad de este, si un cultivo se encuentra bajo amenaza será imposible cosecharlo, se debe exterminar primero la amenaza del cultivo para realizar esta acción.

A continuación, se mostrarán los diagramas de clases, de objetos, conceptos implementados en el aplicativo y los lugares donde estos están ubicados, también se tiene un manual de usuario con todas las especificaciones para el buen manejo de las funcionalidades del programa. (En este manual se han inscrito las imágenes con lo que imprime cada una de las funciones implementadas).

DIAGRAMA DE CLASES

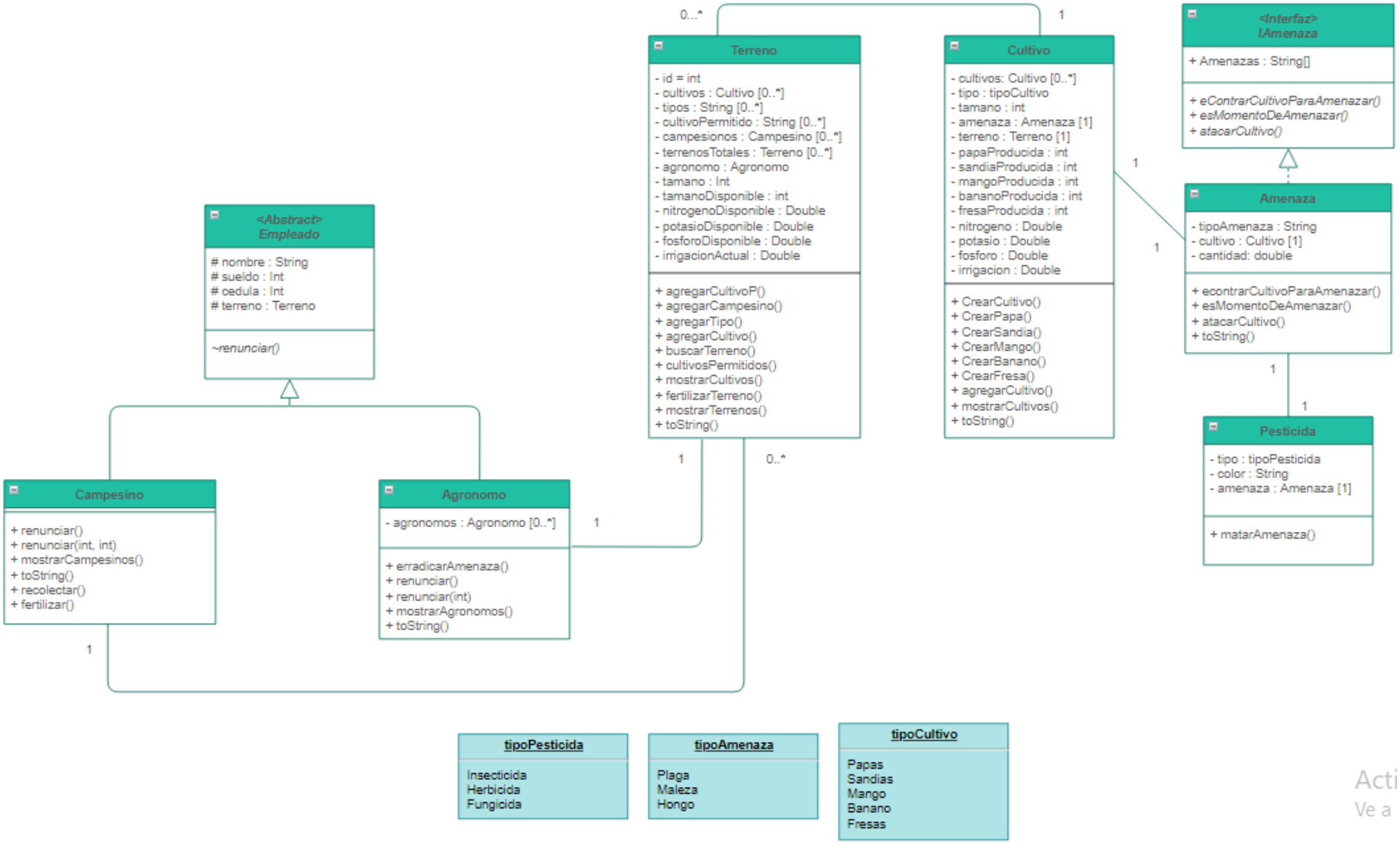
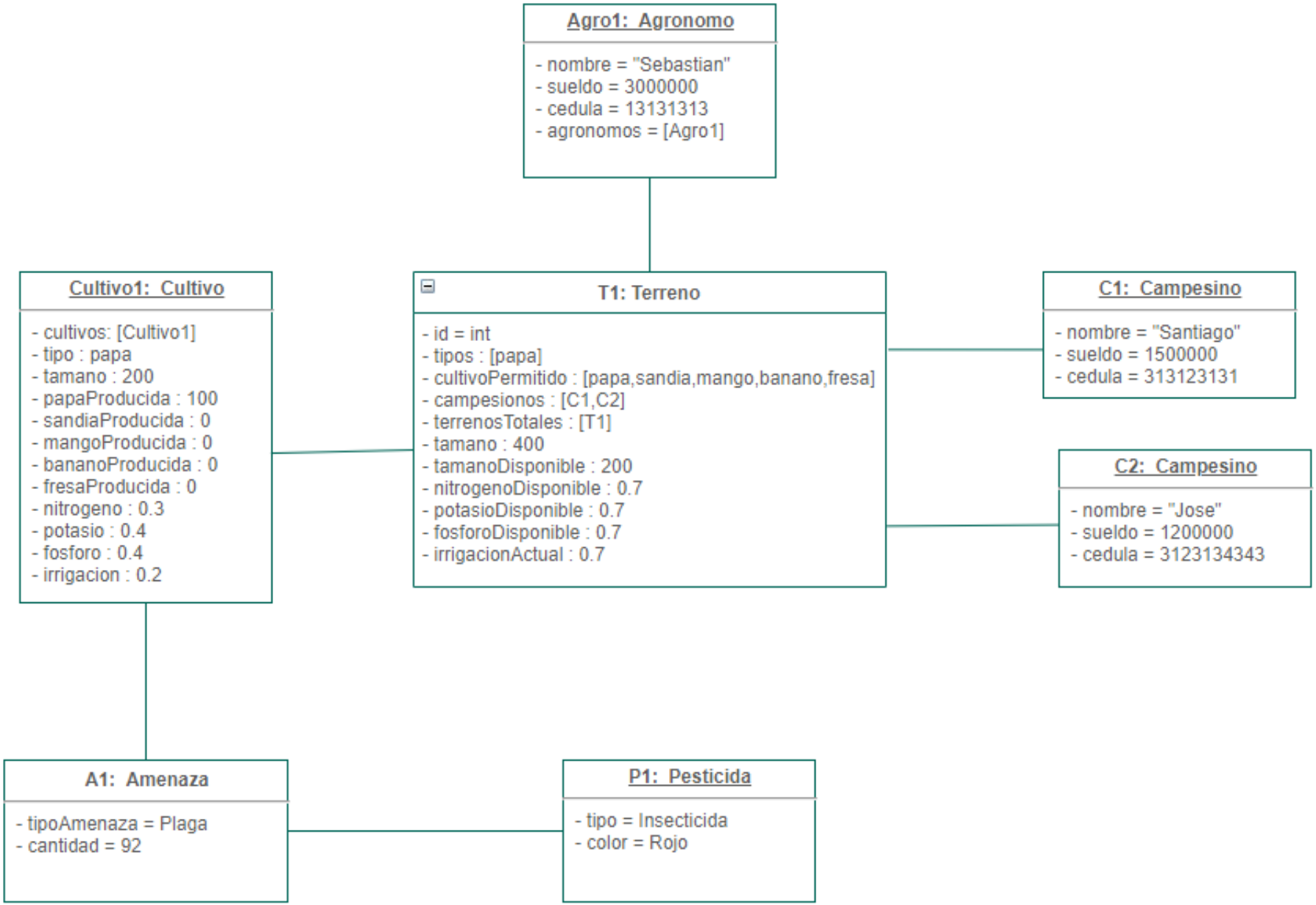


Diagrama De Objetos



Implementación de características de la programación orientada a objetos

1. **Clase abstracta y métodos abstractos**

Se implementó una clase llamada empleado, la cual implementa un método abstracto.

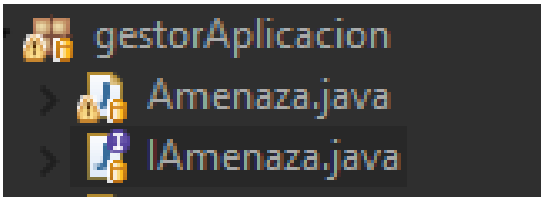
```
public abstract class Empleado implements Serializable{
    protected String nombre;
    protected int sueldo;
    protected int cedula;
    protected Terreno terreno;
    public Empleado() {
    }
    public Empleado(String nombre, int sueldo, int cedula, Terreno terreno) {
        this.nombre = nombre;
        this.sueldo = sueldo;
        this.cedula = cedula;
        this.terreno = terreno;
    }

    public Empleado(String nombre, int cedula, Terreno terreno) {
        this.nombre = nombre;
        this.sueldo = 1000000;
        this.cedula = cedula;
        this.terreno = terreno;
    }

    public abstract void renunciar();
```

2. **Interfaces**

Se ha implementado una interfaz llamada IAmenaza, esta es utilizada por la clase Amenaza.




```
public class Amenaza implements Serializable, IAmenaza {
```

3. **Herencia**

Se tiene una clase empleado, está hereda sus atributos, métodos y constructores a sus clases hijas: agrónomo y campesino.

```java
package gestorAplicacion.empleado;

import java.io.Serializable;

public abstract class Empleado implements Serializable{
    protected String nombre;
    protected int sueldo;
    protected int cedula;
    protected Terreno terreno;
    public Empleado() {
    }
    public Empleado(String nombre, int sueldo, int cedula, Terreno terreno) {
        this.nombre = nombre;
        this.sueldo = sueldo;
        this.cedula = cedula;
        this.terreno = terreno;
    }

    public Empleado(String nombre, int cedula, Terreno terreno) {
        this.nombre = nombre;
        this.sueldo = 1000000;
        this.cedula = cedula;
        this.terreno = terreno;
    }
```

4. Ligadura dinámica

En el diseño utilizado no se ha implementado ligadura dinámica.

5. Atributos de clase y métodos de clase

Se han implementado en varias clases este tipo de atributos y métodos, un ejemplo de este es agrónomo.

```java
package gestorAplicacion.empleado;
import java.io.Serializable;

public class Agronomo extends Empleado implements Serializable{
    private static LinkedList<Agronomo> agronomos = new LinkedList<Agronomo>();
    public Agronomo() {
    }
    public Agronomo(String nombre, int sueldo, int cedula, Terreno terreno) {
        super(nombre, sueldo, cedula, terreno);
        terreno.setAgronomo(this);
        agronomos.add(this);
    }
    public static void erradicarAmenaza(Amenaza amenaza, Cultivo cultivo) { //parte nueva
        if(cultivo.getTerreno().getAgronomo()!=null) {
            Pesticida pesticida = new Pesticida(amenaza);
            pesticida.matarAmenaza(cultivo);
        }
        else {
            System.out.println("No hay agronomo en el terreno con id "+cultivo.getTerreno().getId()+" por favor contrate uno
        }
    }
    public static String mostrarAgronomos() {
        String muestra = ""; int contador = 1;
        for (Integer i = 0; i < agronomos.size(); i++){
                muestra = muestra + (contador) + ". " + agronomos.get(i).getCedula()  + "\n";
                contador++;
        }
        return(muestra);
    }
```

6. Uso de constantes

Se implementó el uso de una constante en la clase terreno, para referirse a su ID, una vez colocado este, ya no se podrá cambiar.

```java
package gestorAplicacion.terreno;
import gestorAplicacion.empleado.*;

public class Terreno implements Serializable{
    private final String id; //Nuestra constante
    private LinkedList<Cultivo> cultivos = new LinkedList<Cultivo>();
    private LinkedList<String> tipos = new LinkedList<String>(); //Tipos de cultivo presentes en el terreno
    private LinkedList<String> cultivoPermitido = new LinkedList<String>();
    private LinkedList<Campesino> campesinos = new LinkedList<Campesino>(); //Campesinos que tabajan en el terreno
    private static LinkedList<Terreno> terrenosTotales = new LinkedList<Terreno>(); // Se usa para mostrar al usuario todos en lo
    private Agronomo agronomo; //Agrónomo que trabaja en el terreno
    private int tamano;
    private int tamanoDisponible;
    private double nitrogenoDisponible;
    private double potasioDisponible;
    private double fosforoDisponible;
    private double irrigacionActual;
```

7. Encapsulamiento

Se ha utilizado un encapsulamiento adecuado, colocando protected en las clases que heredan atributos, private en los atributos de clases que no heredan, y public en los métodos.

```java
public abstract class Empleado implements Serializable{
    protected String nombre;
    protected int sueldo;
    protected int cedula;
    protected Terreno terreno;
    public Empleado() {
    }
    public Empleado(String nombre, int sueldo, int cedula, Terreno terreno) {
        this.nombre = nombre;
        this.sueldo = sueldo;
        this.cedula = cedula;
        this.terreno = terreno;
    }

    public Empleado(String nombre, int cedula, Terreno terreno) {
        this.nombre = nombre;
        this.sueldo = 1000000;
        this.cedula = cedula;
        this.terreno = terreno;
    }

    public abstract void renunciar();

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }
```

```java
public class Cultivo implements Serializable{

    private static LinkedList<Cultivo> cultivos = new LinkedList<Cultivo>();//
    private String tipoCultivo;
    private int tamano;
    private Amenaza amenaza;
    private Terreno terreno;
    private static int papaProducida;
    private static int sandiaProducida;
    private static int mangoProducido;
    private static int bananoProducido;
    private static int fresaProducida;
    private double nitrogeno;
    private double potasio;
    private double fosforo;
    private double irrigacion;
    private int cantCampesino;
```

8. Sobrecarga de métodos y constructores

Se ha utilizado sobrecarga de constructores en la clase empleado.

```
public abstract class Empleado implements Serializable{
    protected String nombre;
    protected int sueldo;
    protected int cedula;
    protected Terreno terreno;
    public Empleado() {
    }
    public Empleado(String nombre, int sueldo, int cedula, Terreno terreno) {
        this.nombre = nombre;
        this.sueldo = sueldo;
        this.cedula = cedula;
        this.terreno = terreno;
    }

    public Empleado(String nombre, int cedula, Terreno terreno) {
        this.nombre = nombre;
        this.sueldo = 1000000;
        this.cedula = cedula;
        this.terreno = terreno;
    }
```

Se ha utilizado sobrecarga de métodos en la clase agrónomo y campesino.

Campesino:

```
 public Campesino(String nombre, int sueldo, int cedula, Terreno terreno) {
     super(nombre, sueldo, cedula, terreno);
     terreno.agregarCampesino(this);
 }
 public void renunciar() {
     double x = Math.random();
     if ((x < 0.15) && (Terreno.getTerreno().size() > 0)) {
         for(int i = 0; i < Terreno.getTerreno().size(); i++) {
             if(Terreno.getTerreno().get(i).getCampesinos().size() > 0) {
                 System.out.println(Terreno.getTerreno().get(i).getCampesinos().get(0));
                 System.out.println("Ha renunciado");
                 Terreno.getTerreno().get(i).getCampesinos().remove(0);
                 break;
             }
         }
     }
 }
 public void renunciar(int opcionElegida, int opcionElegida2) {
     Terreno.getTerreno().get(opcionElegida).getCampesinos().remove(opcionElegida2);
 }
```

Agrónomo:

```
33      public void renunciar() {
34          double x = Math.random();
35          int y = (int) Math.random()*agronomos.size();
36          if ((x < 0.15) && (agronomos.size() > 0)) {
37              System.out.println(agronomos.get(y));
38              System.out.println("Ha renunciado.");
39              agronomos.get(y).getTerreno().setAgronomo(null);
40              agronomos.remove(y);
41          }
42      }
43      public void renunciar(int opcionElegida) {
44          System.out.println(agronomos.get(opcionElegida));
45          System.out.println("Ha sido despedido.");
46          agronomos.get(opcionElegida).getTerreno().setAgronomo(null);
47          agronomos.remove(opcionElegida);
48      }
```

Manejo de referencias

Se ha utilizado el this en todos los constructores para desambiguar. Como también para invocar una sobrecarga de constructores y para referirse al mismo objeto en sí. Un ejemplo es la clase terreno.

```
public Terreno(String id, int tamano, double nitrogenoDisponible, double potasioDisponible, double fosforoDisponible, double
    this.id = id;
    this.tamano = tamano;
    this.nitrogenoDisponible = nitrogenoDisponible;
    this.potasioDisponible = potasioDisponible;
    this.fosforoDisponible = fosforoDisponible;
    this.irrigacionActual = irrigacionActual;
    this.tamanoDisponible = tamano;
    terrenosTotales.add(this);        //Agrego terreno creado
}

public Terreno(String id, int tamano) {
    this(id, tamano, Math.random(), Math.random(), Math.random(), Math.random());
    // ***this.getCultivoPermitido()***
}
```

El otro caso se puede ver en la clase de cultivo.

```
public Cultivo(String tipoCultivo, int tamano, double nitrogeno, double potasio, double fosforo, double irriga
    this.tipoCultivo = tipoCultivo;
    this.tamano = tamano;
    this.nitrogeno = nitrogeno;
    this.potasio = potasio;
    this.fosforo = fosforo;
    this.irrigacion = irrigacion;
    this.terreno = terreno;
    this.agregarCultivo(this);
}
```

Manual de usuario y funcionalidades

Al iniciar el programa se mostrará una consola CMD con el siguiente menú, en el cual tendrá que ingresar el número de la opción que desea elegir; vamos a definir cada una de sus opciones.

```
Selecciona una funcion
1. Contratar
2. Despedir
3. Produccion total de los cultivos
4. Examinar cultivos
5. Cultivar y cosechar
6. Agregar terreno
7. Fertilizar e irrigar terrenos
8. Terminar
```

1. Contratar

La funcionalidad contratar le dará la opción al usuario de contratar a empleados, ya sean campesinos o agrónomos; como también la opción de volver al menú de funcionalidades.

```
¿Qué trabajador desea contratar?
1. Campesino
2. Agronomo
3. Volver
```

Nota: Debe agregar primero un terreno para vincular empleados, de lo contrario no podrá vincularlos para trabajar en algún terreno.

```
1
Debe agregar primero un terreno para vincular trabajadores
```

1.1. Campesino

Al seleccionar campesino podremos contratar a un nuevo empleado.
Se debe ingresar los datos en el orden que se indica, el nombre (sin espacios), el salario es un entero positivo de máximo 9 cifras, asimismo lo es la cédula; también Se debe indicar el terreno al cual lo quiere vincular para laborar.

```
1
Ingrese el nombre:
Chishi
Ingrese el sueldo:
123456789
Ingrese el numero de cedula
987654321
Seleccione un terreno para vincular al trabajador
Opcion 1: 10

1
Se ha contratado un campesino.

El campesino con:
Nombre: Chishi
Cedula: 987654321
Sueldo: 123456789
Vinculado a terreno: 10
```

1.2. Agrónomo

Al seleccionar agrónomo podremos contratar a un nuevo empleado. El ingreso de datos se maneja de manera similar al caso anterior.

```
1
¿Qué trabajador desea contratar?
1. Campesino
2. Agronomo
3. Volver
2
Ingrese el nombre:
Cheneke
Ingrese el sueldo:
98989
Ingrese el numero de cedula
291892
Seleccione un terreno para vincular al trabajador
Opcion 1: 10
```

2. Despedir

La funcionalidad le dará la opción al usuario de despedir a empleados que ya hayan sido contratados, ya sean campesinos o agrónomos; como también la opción de volver al menú de funcionalidades.

```
2
¿Qué trabajador desea despedir:
1. Campesino
2. Agronomo
3. Volver
```

Nota: Debe agregar primero un terreno para despedir empleados, de lo contrario no podrá despedir.

```
2
Debe asignar primero terrenos, para así tener trabajadores y despedirlos.
```

Nota: En ocasiones los campesinos o los agrónomos van a renunciar aleatoriamente.

```
El campesino con:
Nombre: Chichi
Cedula: 12345
Sueldo: 10000
Vinculado a terreno: 10
Ha renunciado
```

```
El agronomo con:
Nombre: jaja
Cedula: 100
Sueldo: 10
Vinculado a terreno: 10
Ha renunciado.
```

2.1 Campesino

Debe seleccionar un terreno para mirar los campesinos vinculados a él, luego selecciona el número de cédula perteneciente al campesino que desea despedir.

```
1
Seleccione un terreno para mirar los campesinos que laboran en el
Opcion 1: 10

1
Seleccione un campesino para despedir
1. 12345
2. 987654321

2
Se ha despedio a:

El campesino con:
Nombre: Chishi
Cedula: 987654321
Sueldo: 123456789
Vinculado a terreno: 10
```

2.2 Agrónomo

Debe seleccionar el número de cédula correspondiente al agrónomo que desea despedir.

```
2
Seleccione un agronomo para despedir
1. 291892

1

El agronomo con:
Nombre: Cheneke
Cedula: 291892
Sueldo: 98989
Vinculado a terreno: 10
Ha sido despedido.
```

3. Producción total de cultivos

Esta opción mostrará la cantidad total de hectáreas que se han cosechado de cultivos.

```
3
Cantidad producida de cada tipo de cultivo
Cantidad de papas: 1.0 hectareas
Cantidad de sandias: 1.0 hectareas
Cantidad de bananos: 1.0 hectareas
Cantidad de mangos: 1.0 hectareas
Cantidad de fresas: 1.0 hectareas
```

4. Examinar Cultivos

Se desplegará un menú con el menú de cultivos y el terreno al que pertenecen, elige un cultivo para mirar las especificaciones y determinar si se encuentra bajo amenaza o no.

```
4
Seleccione un cultivo:
1. El cultivo de mango. En el terreno con id: 10
2. El cultivo de papa. En el terreno con id: 10
3. El cultivo de banano. En el terreno con id: 10
```

4.1 Cultivo sin amenaza

```
3
El cultivo de tipo banano ubicado en el terreno 10 no se encuentra bajo amenaza
```

4.2 Cultivo bajo amenaza

El usuario debe elegir si desea exterminar la amenaza con el pesticida.

```
1
El cultivo de tipo mango ubicado en el terreno 10 se encuentra bajo amenaza.
La amenaza que lo ataca es de tipo Maleza
¿Desea aplicar pesticida para eliminar la amenaza del cultivo?
1. Si
2. No
1
Se ha exterminado la amenaza!
```

Nota: Debe cultivar primero para examinarlos.

```
4
Debe cultivar primero
```

Nota: Si el terreno no tiene un agrónomo vinculado no podrá utilizar el pesticida respectivo para exterminar la amenaza que está atacando al cultivo.

```
1
El cultivo de tipo mango ubicado en el terreno 10 se encuentra bajo amenaza.
La amenaza que lo ataca es de tipo Plaga
¿Desea aplicar pesticida para eliminar la amenaza del cultivo?
1. Si
2. No
1
No hay agronomo en el terreno con id 10 por favor contrate uno para exterminar la amenaza
```

5. Cultivar y cosechar

Se desplegará el menú para elegir la elección correspondiente.

```
5
Escoja la funcion a realizar:
Para crear un cultivo ingrese 1
Para recolectar un cultivo ingrese 2
```

5.1. Crear cultivo

Debe de llenar las opciones como se lo indica la consola, el máximo de tamaño de un cultivo es de 9 dígitos, cabe mencionar que cada vez que se cree un cultivo este tomará una parte del terreno, o su totalidad dependiendo del tamaño.

```
1
Escoja la opcion del terreno en el que quiere sembrar:
Opcion 1: 10
Opcion 2: 200

1
Los cultivos que permite el terreno son:
papa
sandia
mango
banano
fresa

Escriba el tipo que quiere sembrar:
sandia
Escriba el tamaño que desea que tenga el cultivo:
10
Se ha creado exitosamente
```

5.2 Cosechar

La opción de cosechar le permitirá recolectar los cultivos del terreno, luego de tomarlos el terreno tendrá libre el espacio para un nuevo cultivo.

```
2
Escoja la opcion del terreno en el que quiere recolectar:
Opcion 1: 10
Opcion 2: 200

1
Ingrese la opcion que corresponde al tipo
Eliga una de las siguientes opciones
Opcion 1: sandia

1
Se ha recolectado del cultivo de tipo sandia 10 hectareas.
```

Nota: Antes de cultivar se necesita agregar un terreno.

```
5
Escoja la funcion a realizar:
Para crear un cultivo ingrese 1
Para recolectar un cultivo ingrese 2
1
No dispone de terrenos, por favor cree uno
```

Nota: Antes de cosechar se necesita crear un cultivo.

```
1
El terreno no tiene cultivos, por favor cree uno
```

Nota: Antes de cosechar o cultivar, el terreno debe tener asignado al menos un campesino.

```
Escoja la funcion a realizar:
Para crear un cultivo ingrese 1
Para recolectar un cultivo ingrese 2
2
Escoja el id del terreno en el que quiere recolectar:
Opcion 1: 1

1
El terreno no tiene campesinos que puedan recolectar, por favor asigne al menos 1
```

Nota: Puede que, al momento de crear el terreno, este no tenga los requerimientos mínimos para cultivar.

```
1
Escoja el id del terreno en el que quiere sembrar:
Opcion 1: 1

1
No dispone con los requerimientos suficientes para sembrar en este terreno, por favor irrigue o fertilice
```

Nota: Al momento de crear un cultivo, el tamaño de este no puede ser mayor a la del terreno disponible.

```
Escriba el tamaño que desea que tenga el cultivo:
1000
Se ha cancelado la operacion, el tamaño del cultivo no concuerda con el del terreno
```

Nota: Si ya se ha creado un cultivo de un tipo en ese terreno, o se crea un cultivo cuyo tipo no lo permite el terreno, no se creará el cultivo.

```
Escriba el tipo que quiere sembrar:
sandia
Escriba el tamaño que desea que tenga el cultivo:
1
Este cultivo ya estaba creado en este terreno o este tipo no es valido
```

Nota: Si el cultivo se encuentra bajo una amenaza, no se podrá recolectar; dicho cultivo debe estar en optimas condiciones para ser cosechado.

```
1
Ingrese la opcion que corresponde al tipo
Eliga una de las siguientes opciones
Opcion 1: mango

1
El cultivo se encuentra bajo una amenaza, por favor, exterminela para cosecharlo
```

6. Agregar terreno

Esta opción le dará la posibilidad al usuario de añadir terrenos al programa para implementar cultivos y empleados. El ID del terreno puede ser una combinación de letras y números, este no puede llevar espacios, el tamaño del terreno es un entero de máximo 9 dígitos. El terreno tendrá unos atributos aleatorios lo que definirá lo que se puede cultivar en él.

```
6
Indique el Id que le desea poner:
PruebaTerreno420
Ahora indique el tamano deseado:
4200
El terreno ha sido agregado exitosamente
Se ha creado un nuevo terreno con las siguientes propiedades:
Nitrogeno disponible: 0.29
Potasio disponible: 0.96
Fosforo disponible: 0.37
Nivel de irrigacion: 0.15
```

7. Fertilizar e irrigar

Esta opción permitirá al usuario mejorar los atributos del terreno para que la tierra sea más fértil y poder cultivar cualquier tipo de cultivo.

```
7
Escoja el terreno que desea fertilizar e irrigar:

Opcion 1: Prueba420
Opcion 2: 109DKEs092

1
Terreno fertilizado
Los cultivos que permite el terreno son:
papa
sandia
mango
banano
fresa
```

Nota: Si el terreno no tiene campesinos vinculados, no se podrá realizar esta acción.

```
7
Escoja el terreno que desea fertilizar e irrigar:

Opcion 1: Prueba420
Opcion 2: 109DKEs092

1
No ha contratado campesinos para este terreno.
Debe contratar al menos uno para realizar este trabajo
```