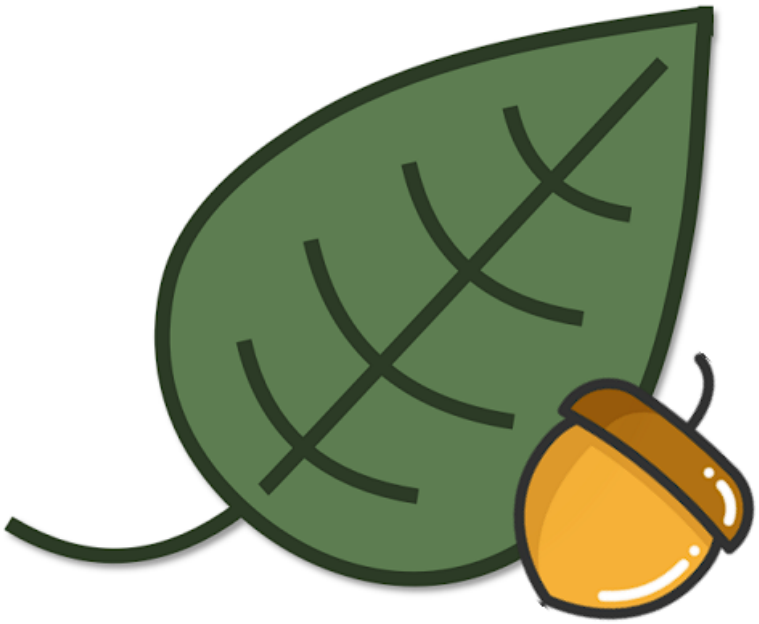
# Sistema invernadero

## Manual técnico

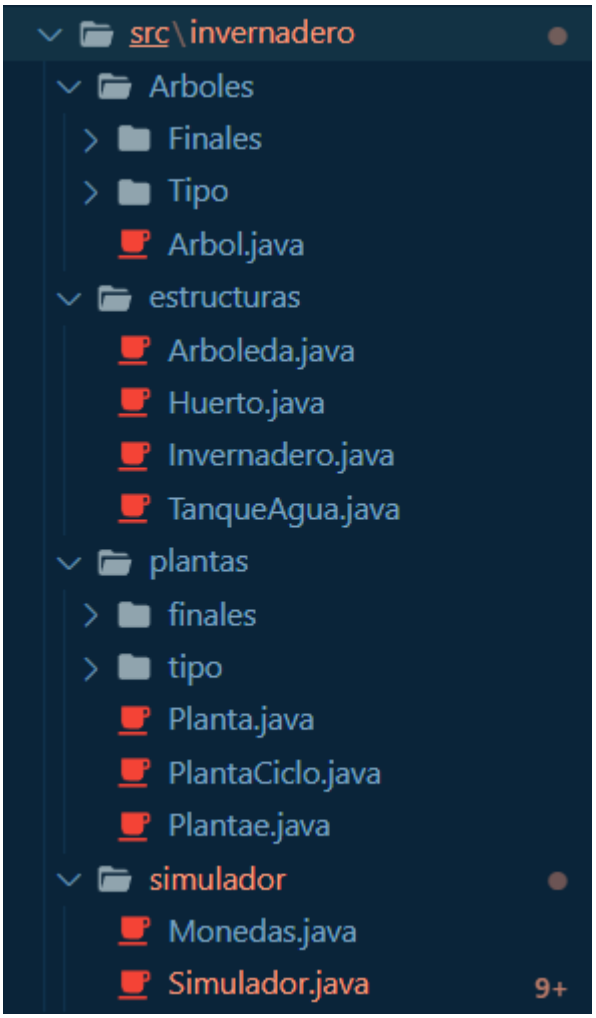


TRABAJO INVERNADERO

Entrega II. Herencia

*Mauro Cordal Casaldarnos 1º DAW*

# Organización de paquetes



# Paquete Simulador

## Clase Simulador

### Atributos

nombres: Array de todos los nombres de las plantas y árboles disponibles

```
private String[] nombres = {AlmacenPropiedades.ARROZ.getNombre(),AlmacenPropiedades.GARBANZOS.getNombre(),AlmacenPropiedades.LECHUGA.getNomb
AlmacenPropiedades.PATATA.getNombre(),AlmacenPropiedades.PIMIENTO.getNombre(),AlmacenPropiedades.TRIGO.getNombre(),
AlmacenPropiedades.KIWI.getNombre(), AlmacenPropiedades.MADRONHO.getNombre(),AlmacenPropiedades.MANZANO.getNombre(),
AlmacenPropiedades.MELOCOTONERO.getNombre(),AlmacenPropiedades.NARANJO.getNombre()};
```

```java
/**
 * si tiene la mejora aspersores
 */
private boolean aspersores = false;

/**
 * si tiene la mejora molino
 */
private boolean molino = false;

/**
 * instancia de la clase Estadisticas para registrar la cosecha
 */
private estadisticas.Estadisticas stats;

/**
 * instancia de la clase monedas
 */
private Monedas monedas = new Monedas();

/**
 * contador de los dias que pasaron desde que empezó el juego
 */
public int numDias = 0;

/**
 * los invernaderos que tiene comprados.
 * Empieza con uno
 */
private Invernadero[] invernaderos;

/**
 * las arboledas compradas
 */
private Arboleda[] arboledas;
```

```java
/**
 * El tanque que tiene el agua para regar.
 * Solo hay uno por partida
 */
private TanqueAgua tanqueDeAgua = new TanqueAgua();

/**
 * El nombre de la partida / empresa del jugador
 */
private String nombreEmpresa;

/**
 * contador de los productos cosechados en todos los invervaderos
 */
private int productosCosechados = 0;
```

Constructor

```java
public static int porcentaje(int num1, int num2){
    float dec1 = num1;
    float dec2 = num2;
    return (int) ((dec1/dec2)*100);
}
```

## Métodos

**porcentaje**
método auxiliar y estático que se usa para calcular un x por ciento

```java
public static int porcentaje(int num1, int num2){
    float dec1 = num1;
    float dec2 = num2;
    return (int) ((dec1/dec2)*100);
}
```

**init**
inicializa stats, arboledas, invernadero y nombreEmpresa

```java
public void init() {
    this.stats = new Estadisticas(this.nombres);
    this.arboledas = new Arboleda[0];
    Scanner sc = new Scanner (System.in);
    System.out.println(x:"Nombre de la empresa: ");
    try {
        this.nombreEmpresa = sc.nextLine();
    } catch (Exception e) {
    }
    this.invernaderos = new Invernadero[1];
    System.out.println(x:"Nombre del invernadero:");
    try {
        this.invernaderos[0] = new Invernadero(sc.nextLine());
    } catch (Exception e) {
    }
}
```

**menu**
muestra la estructura del menú principal

```java
public void menu() {
    System.out.println();
    System.out.println(x:"_____________________");
    System.out.println(x:"| 1. Estado general    |");
    System.out.println(x:"| 2. Estado cultivos   |");
    System.out.println(x:"| 3. Informes          |");
    System.out.println(x:"| 4. Naturapedia       |");
    System.out.println(x:"| 5. Pasar día         |");
    System.out.println(x:"| 6. Recolectar agua   |");
    System.out.println(x:"| 7. Regar             |");
    System.out.println(x:"| 8. Plantar           |");
    System.out.println(x:"| 9. Cosechar          |");
    System.out.println(x:"| 10. Desbrozar        |");
    System.out.println(x:"| 11. Arrancar         |");
    System.out.println(x:"| 12. Mejorar          |");
    System.out.println(x:"| 13. Pasar varios días|");
    System.out.println(x:"| 14. Salir            |");
    System.out.println(x:"_____________________");
}
```

**menuInv**

muestra un menú para seleccionar un invernadero

```
public void menuInv() {
    System.out.println(x:"Seleccione una opcion");
    System.out.println(x:"-------------------------- Invernaderos --------------------------");
    System.out.println(x:"[Plantas vivas / Plantas plantadas / Terreno total]");
    int i = 1;
    for (Invernadero invernadero : invernaderos) {
        if (invernadero != null){
            System.out.println(i + " -" + invernadero.getName() + " " + invernadero.getAlive() + "/" +
            invernadero.getNum() + "/" + invernadero.getTiles());
            i++;
        }
    }
    System.out.println(x:"0 salir");
}
```

**menuArb**

muestra un menú para seleccionar una arboleda

```
public void menuArb() {
    System.out.println();
    System.out.println();
    System.out.println(x:"Seleccione una opcion");
    System.out.println(x:"-------------------------- Arboledas --------------------------");
    System.out.println(x:"[Arboles vivos / Arboles plantados / Terreno total]");
    int i = 1;
    for (Arboleda arboleda : arboledas) {
        if (arboleda != null){
            System.out.println(i + " -" + arboleda.getName() + " " + arboleda.getAlive() + "/" + arboleda.getNum() + "/" +
            arboleda.getTiles());
            i++;
        }
    }
    System.out.println(x:"0 salir");
}
```

**selectInv**

permite seleccionar un invernadero del array invernaderos

```
public int selectInv() {
    this.menuInv();
    Scanner sc = new Scanner (System.in);
    boolean flag = true;
    while (flag){
        int selec = 0;
        try {
            selec = sc.nextInt();

        } catch (Exception e) {
            System.err.println(x:"Pon un numero valido!!!");
        }
        if (selec != 0) {
            if (selec -1 <= this.invernaderos.length) {
                return selec - 1;
            }else{
                System.out.println(x:"Pon un numero valido!!!");
            }
        }
        if (selec == 0){
            flag = false;
        }

    }
    return -1;
}
```

**selectArb**
permite seleccionar una arboleda del array arboledas

```java
public int selectArb() {
    this.menuArb();
    Scanner sc = new Scanner (System.in);
    boolean flag = true;
    while (flag){
        int selec = 0;
        try {
            selec = sc.nextInt();

        } catch (Exception e) {
            System.err.println(x:"Pon un numero valido!!!");
        }
        if (selec != 0) {
            if (selec -1 <= this.arboledas.length) {
                return selec - 1;
            }else{
                System.out.println(x:"Pon un numero valido!!!");
            }
        }else{
            System.out.println();
            flag = false;
        }

    }
    return -1;
}
```

**showGeneralStatus**:
muestra el status de cada estructura, el numero de días, si tiene molino y las monedas

```java
public void showGeneralStatus() {
    for (Invernadero invernadero : invernaderos) {
        if (invernadero != null) {
            invernadero.showStatus();
        }
    }

    for (Arboleda arboleda : arboledas) {
        if (arboleda != null) {
            arboleda.showStatus();
        }
    }
    System.out.println();
    tanqueDeAgua.showStatus();
    System.out.println("Dia:" + numDias);

    if (molino) {
        System.out.println(x:"El molino añade un 20% de ganancia a la hora de recolectar los cultivos triturables!!!");
    }else{
        System.out.println(x:"Aun no tienes molino");
    }
    this.monedas.showMonedas();
}
```

**showSpecificStatus**

muestra un menú para seleccionar un invernadero o una arboleda en concreto y de esta muestra los stats y los de sus plantas o arboles

```java
public void showSpecificStatus() {
    Scanner sc = new Scanner (System.in);
    Boolean flag = true;
    do{
        int indice = -1;
        indice = -1;
        int cont = 1;
        System.out.println();
        System.out.println();
        System.out.println(x:"De que quieres ver mas informacion???");

        for (Invernadero invernadero : invernaderos) {
            if (invernadero != null) {
                System.out.println(cont +  " [I] " + invernadero.getName());
                cont++;
            }
        }
        for (Arboleda arboleda : arboledas) {
            if (arboleda != null) {
                System.out.println(cont + " [A] " + arboleda.getName());
                cont++;
            }
        }
        System.out.println(cont + " Salir");
        try {
            indice = sc.nextInt();
        } catch (Exception e) {

        }
        if (indice == -1 || indice == cont) {
            flag = false;
        }else{
            if (indice >= 1 && indice < cont) {
                if (indice > this.invernaderos.length) {
                    indice = indice - this.invernaderos.length;
                    if (arboledas[indice-1] != null) {
                        System.out.println(arboledas[indice-1].getName() + " es nivel " + arboledas[indice-1].getNivel());
                        if (arboledas[indice-1].isGoteo()) {
                            System.out.println(arboledas[indice-1].getName() + " tiene siestema de riego por goteo instalado");
                        }
                        System.out.println();
                        arboledas[indice-1].showTileStatus();
                    }
                    }
                }else{
                    if (invernaderos[indice-1] != null) {
                        System.out.println(invernaderos[indice-1].getName() + " es nivel " + invernaderos[indice-1].getNivel());
                        invernaderos[indice-1].showTileStatus();
                    }
                }
            }else{
            System.out.println(x:"pon un numero valido");
            }
        }
    }while(flag);
}
```

**nextDay**
llama al método growCrops de cada estructura y si tiene aspersores primero al método waterCrops

```java
public void nextDay() {
    System.out.println();
    System.out.println();
    if (aspersores) {
        this.waterCrops();
    }
    for (Invernadero invernadero : invernaderos) {
        if (invernadero != null) {

            invernadero.growCrops();
        }
    }
    for (Arboleda arboleda : arboledas) {
        if (arboleda != null) {
            arboleda.growCrops();
        }
    }

    this.numDias++;
}
```

**addWater**

pide un número que puede ser 1, 5, 10, 33 o 0 para añadir al tanque (33 es para llenar y 0 cancelar) se comprueba cuánta agua falta en el tanque y si se puede pagar el precio

```
public void addWater() {
    Scanner sc = new Scanner (System.in);
    boolean flag = true;
    while(flag){
        System.out.println();
        System.out.println();
        System.out.println(x:"--------------------------------");
        System.out.println(x:"AÑADIR AGUA");
        System.out.println(x:"elige la cantidad a añadir");
        System.out.println(x:"1 / 5 / 10");
        System.out.println(x:"33 para llenar");
        System.out.println(x:"0 para salir");
        System.out.println(x:"--------------------------------");
        int agua = 0;
        try {
            agua = sc.nextInt();
        } catch (Exception e) {
        }
        if (agua == 0 || agua == 1 || agua == 5 || agua == 10 || agua == 33) {
            if (agua == 33) {
                if (monedas.gastar((tanqueDeAgua.getMax() - tanqueDeAgua.getWater()) - (tanqueDeAgua.getMax() - tanqueDeAgua.getWater())/10)) {
                    tanqueDeAgua.addWater(tanqueDeAgua.getMax() - tanqueDeAgua.getWater());
                    System.out.println(x:"Tanque de agua lleno");
                }
            }else{
                if (agua > tanqueDeAgua.getMax() - tanqueDeAgua.getWater()) {
                    agua = tanqueDeAgua.getMax() - tanqueDeAgua.getWater();
                }
                if (monedas.gastar(agua)) {
                    tanqueDeAgua.addWater(agua);
                    System.out.println("añadidos " + agua + " de agua");
                    tanqueDeAgua.showStatus();
                    monedas.showMonedas();
                }
            }
            flag = false;
        }else{
            System.out.println(x:"introduce un numero valido!!!");
        }
    }
}
```

**waterCrops**

riega todas las plantas y árboles y usa el agua necesaria

```
public void waterCrops() {
    int contRegadas = 0;
    int contVivas = 0;
    for (Invernadero invernadero : invernaderos) {
        if (invernadero != null) {
            int[] cont = invernadero.waterCrops(tanqueDeAgua.getWater());
            contRegadas = contRegadas + cont[0];
            contVivas = contVivas + invernadero.getAlive();
            this.tanqueDeAgua.usar(cont[1]);
        }
    }
    for (Arboleda arboleda : arboledas) {
        if (arboleda != null) {
            Random ran = new Random();
            int[] cont = arboleda.waterCrops(tanqueDeAgua.getWater());
            contRegadas = contRegadas + cont[0];
            contVivas = contVivas + arboleda.getAlive();
            if (arboleda.isGoteo()) {
                if (ran.nextBoolean()) {
                    this.tanqueDeAgua.usar(cont[1]);
                }
            }else{
                this.tanqueDeAgua.usar(cont[1]);
            }
        }
    }
    System.out.println("se han regado " + contRegadas + "/" + contVivas + " plantas y arboles de los invernaderos y arboledas");
}
```

**plant**

el usuario selecciona una de las plantas o árboles finales y se crea un objeto de cada una para mostrar el precio y se añade a la estructura seleccionada el tipo de planta seleccionada si hay monedas suficientes

```
public void plant() {
    int prec = 0;
    Scanner sc = new Scanner (System.in);
    System.out.println();
    System.out.println();
    int selec = invOrArb(this.arboledas.length > 0);
    if (selec == 1) {
        boolean flag = true;
        while (flag) {
            Trigo tri = new Trigo();
            Arroz arr = new Arroz();
            Garbanzos gar = new Garbanzos();
            Lechuga lec = new Lechuga();
            Patata pat = new Patata();
            Pimientos pim = new Pimientos();
            System.out.println();
            System.out.println();
            System.out.println("1. Trigo       " + tri.getPrice() + " monedas");
            System.out.println("2. Arroz       " + arr.getPrice() + " monedas");
            System.out.println("3. Garbanzos   " + gar.getPrice() + " monedas");
            System.out.println("4. Lechuga     " + lec.getPrice() + " monedas");
            System.out.println("5. Patata      " + pat.getPrice() + " monedas");
            System.out.println("6. Pimientos   " + pim.getPrice() + " monedas" );
            System.out.println(x:"7. Salir");
            int plt = 7;
            try {
                plt = sc.nextInt();
            } catch (Exception e) {
                System.err.println(x:"Selecciona un numero valido!!!");
            }
            if (plt == 7) {
                return;
            }
```

```
switch (plt) {
    case 1:
        prec = tri.getPrice();
        break;
    case 2:
        prec = arr.getPrice();
        break;
    case 3:
        prec = gar.getPrice();
        break;
    case 4:
        prec = lec.getPrice();
        break;
    case 5:
        prec = pat.getPrice();
        break;
    case 6:
        prec = pim.getPrice();
        break;
}
if (plt >= 1 && plt <= 6) {
    if (monedas.gastable(prec)) {
    int indice = selectInv();
        if (indice == -1) {
            return;
        }
        if (indice >= 0 && indice < invernaderos.length) {
            if (invernaderos[indice] != null) {
                if (!invernaderos[indice].isLleno()) {
                    invernaderos[indice].plants(plt);
                    invernaderos[indice].showCapacity();
                    monedas.gastar(prec);
                    flag = false;
```

```
                    }else{
                        System.out.println(x:"Este invernadero esta lleno");
                    }
                }
            }
        }
    }else{
        System.out.println(x:"Selecciona un numero valido!!!");
    }

}
}
```

```
if (selec == 2 && this.arboledas.length > 0) {
    boolean flag = true;
    while (flag) {
        Kiwi ki = new Kiwi();
        Madronho mad = new Madronho();
        Manzano man = new Manzano();
        Melocotonero mel = new Melocotonero();
        Naranjo nar = new Naranjo();
        System.out.println();
        System.out.println();
        System.out.println("1. Kiwi          " + ki.getPrice() + " monedas")
        System.out.println("2. Madroño       " + mad.getPrice() + " monedas"
        System.out.println("3. Manzano       " + man.getPrice() + " monedas"
        System.out.println("4. Melocotonero  " + mel.getPrice() + " monedas"
        System.out.println("5. Naranjo       " + nar.getPrice() + " monedas"
        System.out.println(x:"6. Salir");
        int arb = 6;
        try {
            arb = sc.nextInt();
        } catch (Exception e) {
            System.err.println(x:"Selecciona un numero valido!!!");
        }
        if (arb == 6) {
            return;
        }
        switch (arb) {
            case 1:
                prec = ki.getPrice();
                break;
            case 2:
                prec = mad.getPrice();
                break;
            case 3:
                prec = man.getPrice();
                break;
            case 4:
```

```
                case 4:
                    prec = mel.getPrice();
                    break;
                case 5:
                    prec = nar.getPrice();
                    break;
            }
            if (arb >= 1 && arb <= 5) {
                if (monedas.gastable(prec)){
                    int indice = selectArb();
                    if (indice == -1) {
                        return;
                    }
                    if (indice >= 0 && indice < arboledas.length) {
                        if (arboledas[indice] != null) {
                            if (!arboledas[indice].isLleno()) {
                                arboledas[indice].plants(arb);
                                arboledas[indice].showCapacity();
                                monedas.gastar(prec);
                                flag = false;
                            }else{
                                System.out.println(x:"Esta arboleda esta llena");
                            }
                        }
                    }
                }

            }else{
                System.out.println(x:"Selecciona un numero valido!!!");
            }
        }
    }

}
```

**harvest**

cosecha o todos los invernaderos o todas las arboledas, crea contadores de recolectado, dinero ganado, plantas muertas en el proceso y ganado con plantas triturables y va contando lo que consigue con cada planta o árbol. Si tiene molino registra la harina

```
public void harvest() {
    int ganadoTrit = 0;
    int recolectado = 0;
    int cont = 0;
    int ganado = 0;
    int muertas = 0;
    int opcion = invOrArb(this.arboledas.length > 0);
    if (opcion == 1) {
        for (Invernadero invernadero : invernaderos) {
            if (invernadero != null) {
                int[] harvs = invernadero.harvest(stats);
                recolectado = recolectado + harvs[0];
                muertas = muertas + harvs[1];
                cont = cont + harvs[2];
                ganado = ganado + harvs[3];
                ganadoTrit = ganadoTrit + harvs[4];
                if (this.molino) {
                    ganadoTrit = (ganadoTrit * 120)/100;
                    stats.registrarHarina(ganadoTrit);
                }
            }
        }
    }
    if (opcion == 2 && this.arboledas.length > 0) {
        for (Arboleda arboleda : arboledas) {
            if (arboleda != null) {
                int[] harvs = arboleda.harvest(stats);
                recolectado = recolectado + harvs[0];
                cont = cont + harvs[2];
                ganado = ganado + harvs[3];
            }
        }
    }
```

```
    if (opcion == 3) {
        return;
    }
    monedas.addMonedas(ganado + ganadoTrit);
    System.out.println(cont + " plantas recolectadas han producido " + recolectado + " productos");
    System.out.println("Han muerto " + muertas + " plantas en el proceso");
    System.out.println("Has ganado " + (ganado + ganadoTrit) + " monedas");
}
```

**plow**
llama al método plow de todos los invernaderos o de todas las arboledas y lo ganado con el método plow de arboledas lo añade a monedas

```
public void plow() {
    int opcion = invOrArb(this.arboledas.length > 0);
    if (opcion == 1) {
        for (Invernadero invernadero : invernaderos) {
            if (invernadero != null) {
                invernadero.plow();
            }
        }
    }
    if (opcion == 2 && this.arboledas.length > 0) {
        int ganado = 0;
        for (Arboleda arboleda : arboledas) {
            if (arboleda != null) {
                ganado = ganado + arboleda.plow();
            }
        }
        monedas.addMonedas(ganado);
    }

}
```

**unroot**

llama al método unroot de un edificio en concreto. Comprueba que

```
public void unroot() {
    int opcion = invOrArb(this.invernaderos.length > 0);
    if (opcion == 1) {
        boolean flag = true;
        while (flag){
            int indice = selectInv();
            if (indice >= 0 && indice < invernaderos.length) {
                invernaderos[indice].unroot();
                flag = false;
            }
            if (opcion == 0) {
                return;
            }
        }
    }
    if (opcion == 2 && this.arboledas.length > 0) {
        boolean flag = true;
        while (flag) {
            int indice= selectArb();
            int coste = arboledas[indice].getPrice()/2;

            if (opcion == 0 || !monedas.gastable(coste)) {
                return;
            }
            if (indice >= 0 && indice < this.arboledas.length) {
                arboledas[indice].unroot();
                monedas.gastar(coste);
                flag = false;
            }

        }
    }

}
```

**upgrade**
con una estructura de cases que se resume en esta estructura. También se controla que se pueda comprar la mejora y de gastar las monedas al comprarla, que no puedes mejorar un edificio a nivel máximo ni volver a comprar las mejoras de compra única como el molino, riego por goteo o aspersores

1. Comprar edificio
    a. Invernadero
    b. Arboleda
    c. Molino
2. Mejorar edificio
    a. Invernadero
        i. Aumentar tamaño
    b. Arboleda
        i. Aumentar tamaño
        ii. Añadir sistema de riego por goteo
3. Mejorar tanque de agua
    a. Aumentar tamaño
    b. Añadir aspersores
4. Cancelar

```java
public void upgrade() {
    boolean flag = true;
    while (flag) {
        System.out.println();
        System.out.println();
        System.out.println(x:" Elige que mejorar");
        System.out.println();
        System.out.println(x:"1 Comprar edificio");
        System.out.println(x:"2 Mejorar edificio");
        System.out.println(x:"3 Mejorar tanque de agua");
        System.out.println(x:"4 cancelar");
        Scanner sc = new Scanner (System.in);
        int opcion = 0;
        int precio;
        try {
            opcion = sc.nextInt();
        } catch (Exception e) {
        }
        switch (opcion) {
            case 1:
                System.out.println();
                System.out.println();
                System.out.println(x:"1. Invernadero");
                System.out.println(x:"2. Arboleda");
                if (!this.molino) {
                    System.out.println(x:"3. Molino");
                }
                try {
                    opcion = sc.nextInt();
                } catch (Exception e) {
                }
                sc.nextLine();
                int indice;
                Boolean x;
                String nombre = "";
                switch (opcion) {
```

```java
case 1:
    System.out.println();
    System.out.println();
    System.out.println(x:"1. Invernadero");
    System.out.println(x:"2. Arboleda");
    if (!this.molino) {
        System.out.println(x:"3. Molino");
    }
    try {
        opcion = sc.nextInt();
    } catch (Exception e) {
    }
    sc.nextLine();
    int indice;
    Boolean x;
    String nombre = "";
    switch (opcion) {
        case 1:
            x = monedas.gastar(this.invernaderos.length*500);
            if (!x) {
                break;
            }
            System.out.println(x:"Nombre del invernadero nuevo: ");
            try {
                nombre = sc.nextLine();
            } catch (Exception e) {
            }
            Invernadero[] inv = new Invernadero[this.invernaderos.length + 1];
            System.arraycopy(this.invernaderos, srcPos:0, inv, destPos:0, this.invernaderos.length);
            inv[inv.length - 1] = new Invernadero(nombre);
            this.invernaderos = inv;
            break;
        case 2:
            x = monedas.gastar(this.arboledas.length*500);
            if (!x) {
                break;
            }
            System.out.println(x:"Nombre de la arboleda nueva: ");
            try {
                nombre = sc.nextLine();
            } catch (Exception e) {
            }
            Arboleda[] arb = new Arboleda[this.arboledas.length + 1];
            System.arraycopy(this.arboledas, srcPos:0, arb, destPos:0, this.arboledas.length);
            arb[arb.length - 1] = new Arboleda(nombre);
            this.arboledas = arb;
            break;
```

```
        case 3:
            x = monedas.gastar(price:3000);
            if (!this.molino && x) {
                this.molino = true;
            }
            break;
    }
    break;
case 2:
    System.out.println();
    System.out.println();
    System.out.println(x:"1. Invernadero");
    if (this.arboledas.length > 0) {
        System.out.println(x:"2. Arboleda");
    }
    try {
        opcion = sc.nextInt();
    } catch (Exception e) {
    }
    switch (opcion) {
        case 1:
            System.out.println();
            System.out.println();
            indice = selectInv();
                if (indice >= 0 && indice < invernaderos.length) {
                    if (invernaderos[indice].getNivel() == 10) {
                        System.out.println();
                        System.out.println();
                        System.out.println(x:"No hay mejoras disponibles");
                    }else{
                        System.out.println();
                        System.out.println();
                        System.out.println(invernaderos[indice].getName() + " es nivel " + invernaderos[indice].getNivel());
                        precio = 150*invernaderos[indice].getNivel();
                        System.out.println("La siguiente mejora cuesta " + precio + " monedas");
                        System.out.println(x:"1. Aumentar tamaño");
                        try {
                            opcion = sc.nextInt();
                        } catch (Exception e) {
                        }
                        if (opcion == 1) {
                            x = monedas.gastar(precio);
                            if(x){
                                System.out.println();
                                System.out.println();
                                invernaderos[indice].upgrade();
                            }
                        }
                    }
```

```java
case 2:
    System.out.println();
    System.out.println();
    indice = selectArb();
        if (indice >= 0 && indice < arboledas.length) {
            if (arboledas[indice].getNivel() == 10 && !arboledas[indice].isGoteo()) {
                System.out.println();
                System.out.println();
                System.out.println(x:"No hay mejoras disponibles");
            }else{
                System.out.println();
                System.out.println();
                System.out.println(arboledas[indice].getName() + " es nivel " + arboledas[indice].getNivel());
                precio = 150*arboledas[indice].getNivel();
                System.out.println("1. Aumentar tamaño                    "+ precio + " monedas");
                if (!arboledas[indice].isGoteo()) {
                    System.out.println(x:"2. Añadir sistema de riego por goteo   500 monedas");
                }
                try {
                    opcion = sc.nextInt();
                } catch (Exception e) {
                }
                if (opcion == 1) {
                    x = monedas.gastar(precio);
                    if(x){
                        System.out.println();
                        System.out.println();
                        arboledas[indice].upgrade();
                    }
                }
                if (opcion == 2 && !arboledas[indice].isGoteo()) {
                    if (indice >= 0 && indice < arboledas.length) {
                        if (!this.arboledas[indice].isGoteo()) {
                            x = monedas.gastar(price:500);
                            if (x) {
                                System.out.println();
                                System.out.println();
                                this.arboledas[indice].gotear();
                            }

                        }
                    }
                }
            }
        }
        break;
}
break;
```

```
            case 3:
                System.out.println();
                System.out.println();
                System.out.println(x:"1. Aumentar tamaño     500 monedas");
                if (!this.aspersores) {
                    System.out.println(x:"2. Añadir aspersores  1000 monedas");
                }
                try {
                    opcion = sc.nextInt();
                } catch (Exception e) {
                }
                if (opcion == 1) {
                    x = monedas.gastar(price:500);
                    if (x) {
                        tanqueDeAgua.upgrade();
                    }
                }
                if (opcion == 2 && !aspersores) {
                    x = monedas.gastar(price:1000);
                    if (x) {
                        this.aspersores = true;
                    }
                }
                break;
            case 4:
                System.out.println(x:"saliendo...");
                flag = false;
                break;
            default:
                System.out.println(x:"Pon un numero valido");
                break;
        }
        opcion = 0;
    }
}
```

**forwardDays**
llama al método nextday del propio simulador tantas veces como se lo indiques

```java
public void forwardDays() {
    Scanner sc = new Scanner (System.in);
    System.out.println();
    System.out.println();
    System.out.println(x:"Cuantos dias quieres pasar?");
    int x = 0;
    try {
        x = sc.nextInt();
    } catch (Exception e) {
    }
    for (int i = 0; i < x; i++) {
        this.nextDay();
    }
}
```

**invOrArb**
método auxiliar para seleccionar entre invernadero o arboleda. devuelve 1 si es invernadero o 2 si es arboleda

```java
private int invOrArb(boolean arbo){
    int elec = 1;
    boolean flag = true;
    while (flag){
        System.out.println(x:"1. Invernadero");
        if (arbo) {
            System.out.println(x:"2. Arboleda");
        }
        System.out.println(x:"3. Salir");
        Scanner sc = new Scanner (System.in);
        try {
            elec = sc.nextInt();
        } catch (Exception e) {
        }
        if (elec != 1 && elec != 2 && elec != 3) {
            System.out.println(x:"Elige un numero valido");
        }else{
            flag = false;
        }
    }
    return elec;
}
```

**showStats**
llama al método mostrar del atributo stats

```
public void showStats() {
    this.stats.mostrar();
}
```

**showNatura**
muestra informacion de un tipo de planta o arboles concreto creando un objeto del tipo que el usuario quiere la informacion y llamando a su metodo showInfoNatura

```java
public void ShowNatura() {
    Scanner sc = new Scanner (System.in);
    int selec = 0;
    int opcion = 0;
    while(selec != 3){
        System.out.println();
        System.out.println();
        selec = invOrArb(arbo:true);
        if (selec == 1) {
            System.out.println();
            System.out.println();
            System.out.println(x:"1. Trigo");
            System.out.println(x:"2. Arroz");
            System.out.println(x:"3. Garbanzos");
            System.out.println(x:"4. Lechuga");
            System.out.println(x:"5. Patata");
            System.out.println(x:"6. Pimientos");
            try {
                opcion = sc.nextInt();
            } catch (Exception e) {}
            switch (opcion) {
                case 1:
                    Trigo trig = new Trigo();
                    trig.showInfoNatura();
                    break;
                case 2:
                    Arroz arr = new Arroz();
                    arr.showInfoNatura();
                    break;
                case 3:
                    Garbanzos gar = new Garbanzos();
                    gar.showInfoNatura();
                    break;
                case 4:
                    Lechuga lec = new Lechuga();
                    lec.showInfoNatura();
                    break;
                case 5:
                    Patata pat = new Patata();
                    pat.showInfoNatura();
                    break;
                case 6:
                    Pimientos pim = new Pimientos();
                    pim.showInfoNatura();
                    break;
                default:
                    System.out.println(x:"Escribe un numero valido!!!");
                    break;
```

```
            if(selec == 2){
                System.out.println();
                System.out.println();
                System.out.println(x:"1. Kiwi");
                System.out.println(x:"2. Madroño");
                System.out.println(x:"3. Manzano");
                System.out.println(x:"4. Melocotonero");
                System.out.println(x:"5. Naranjo");
                try {
                    opcion = sc.nextInt();
                } catch (Exception e) {}
                switch (opcion) {
                    case 1:
                        Kiwi ki = new Kiwi();
                        ki.showInfoNatura();
                        break;
                    case 2:
                        Madronho mad = new Madronho();
                        mad.showInfoNatura();
                        break;
                    case 3:
                        Manzano man = new Manzano();
                        man.showInfoNatura();
                        break;
                    case 4:
                        Melocotonero mel = new Melocotonero();
                        mel.showInfoNatura();
                        break;
                    case 5:
                        Naranjo nar = new Naranjo();
                        nar.showInfoNatura();
                        break;
                    default:
                        System.out.println(x:"Escribe un numero valido!!!");
                        break;
                }
            }
        }
    }
```

**main**

```
public static void main(String[] args) {
    Simulador sim = new Simulador();
    sim.init();
    boolean flag = true;
    while (flag) {
        Random ran = new Random();
        sim.menu();
        Scanner sc = new Scanner (System.in);
        int opcion = 0;
        try {
            opcion = sc.nextInt();
        } catch (Exception e) {
        System.err.println(x:"Escribe un numero valido!!!");
        }
        switch(opcion){
            case 1:
                sim.showGeneralStatus();
                break;
            case 2:
                sim.showSpecificStatus();
                break;
            case 3:
                sim.showStats();
                break;
            case 4:
                sim.ShowNatura();
                break;
            case 5:
                sim.nextDay();
                sim.showGeneralStatus();
                break;
            case 6:
                sim.addWater();
                break;
            case 7:
                sim.waterCrops();
                break;
            case 8:
                sim.plant();
                break;
            case 9:
                sim.harvest();
                break;
            case 10:
                sim.plow();
                break;
```

```
                case 11:
                    sim.unroot();
                    break;
                case 12:
                    sim.upgrade();
                    break;
                case 13:
                    sim.forwardDays();
                    sim.showGeneralStatus();
                    break;
                case 14:
                    System.out.println(x:"Saliendo...");
                    flag = false;
                    break;
```

```
                case 98:
                    int selec = sim.invOrArb(sim.arboledas.length > 0);
                    if (selec == 1) {
                        selec = sim.selectInv();
                        if (selec == -1) {
                            return;
                        }
                        for (int i = 0; i < 4; i++) {
                            if (sim.invernaderos[selec]!=null) {
                                if (!sim.invernaderos[selec].isLleno()) {
                                    int ind = ran.nextInt(bound:5)+1;
                                    sim.invernaderos[selec].plants(ind);
                                }
                            }
                        }
                    }
                    if (selec == 2 && sim.arboledas.length > 0) {
                        selec = sim.selectArb();
                        if (selec == -1) {
                            return;
                        }
                        for (int i = 0; i < 5; i++) {
                            if (sim.arboledas[selec]!=null) {
                                if (!sim.arboledas[selec].isLleno()) {
                                    int ind = ran.nextInt(bound:4)+1;
                                    sim.arboledas[selec].plants(ind);
                                }
                            }
                        }
                    }
                    break;
                case 99:
                    sim.monedas.addMonedas(gain:1000);
                    break;
                default:
                    System.out.println(x:"Escribe un numero valido!!!");
                    break;
            }
        }
    }
}
```

# Clase Monedas

## Atributos

```java
/**
 * monedas del jugador
 */
private int monedas = 100;
```

## Métodos

**showMonedas**
muestra por pantalla las monedas

```java
public void showMonedas() {
    System.out.println("Tienes " + this.monedas + " monedas!!!");
}
```

**addMonedas**
suma a monedas la cantidad pasada por parámetro

```java
public void addMonedas(int gain){
    this.monedas = this.monedas + gain;
}
```

**gastable**
devuelve true si monedas restado número pasado por parámetro es mayor que 0

```java
public boolean gastable(int price) {
    if ((monedas - price) < 0) {
        System.out.println(x:"Monedas insufiecientes!!!");
        return false;
    }
    return true;
}
```

**gastar**
resta de monedas el número pasado por parametro

```java
public boolean gastar(int price){
    if (this.monedas - price < 0) {
        System.out.println(x:"Monedas insufiecientes!!!");
        return false;
    }else{
        this.monedas = this.monedas - price;
        return true;
    }
}
```

# Paquete estructuras

## Clase abstracta Huerto

### Atributos

```java
/**
 * nivel de mejora. Por cada nivel aumenta 10 espacios. Empiexa en 1
 */
protected int nivel = 1;

/**
 *nombre del invernadero
 */
protected String name;

/**
 *las plantas que tiene el invernadero
 */
protected Plantae[] plantas;

/**
 * capacidad para 10 plantas por defecto
 * @param name nombre del invernadero.
 */
```

### Constructor

Inicializa name con la string que se pasa por parámetros y plantas con un array de Plantae

```java
protected Huerto(String name) {
    this.name = name;
    plantas = new Plantae[10];
}
```

## Métodos

**getters**

```java
public int getNivel(){
    return nivel;
}

/**
 * @return nombre del invernadero
 */
public String getName() {
    return name;
}

/**
 * @return cantidad plantas plantadas
 */
public int getNum(){
    int res = 0;
    for (Plantae planta : this.plantas) {
        if (planta != null) {
            res++;
        }
    }
    return res;
}

/**
 * @return cantidad espacio para plantas,
 */
public int getTiles(){
    return plantas.length;
}
```

```java
public int getAlive(){
    int res = 0;
    for (Plantae planta : plantas) {
        if (planta != null && !planta.isDead()) {
            res++;
        }
    }
    return res;
}

/**
 * @return cantidad plantas maduras
 */
public int getMature() {
    int res = 0;
    for (Plantae planta : plantas) {
        if (planta != null && planta.isMature()) {
            res++;
        }
    }
    return res;
}

/**
 * @return cantidad plantas regadas
 */
public int getWatered() {
    int res = 0;
    for (Plantae planta : plantas) {
        if (planta != null && planta.isWatered()) {
            res++;
        }
    }
    return res;
}
```

**isIleno**
devuelve si está lleno o no

```java
public boolean isLleno() {
    return plantas[plantas.length - 1] != null;
}
```

**showStatus**
devuelve por pantalla información sobre la estructura y las plantas que contiene

```java
public void showStatus() {
    System.out.println("=============== " + this.name + " ===============");
    System.out.println("Nivel: " + this.nivel + " / 10");
    System.out.println("Ocupacion: " + this.getNum() + "/" + this.getTiles() + " " + Simulador.porcentaje(getNum(), getTiles()) + "%");

    if (getAlive() != 0) {
        System.out.println("Plantas vivas: " + this.getAlive() + "/" + this.getNum() +" "+ Simulador.porcentaje(getAlive(), getNum()) + "%");
        System.out.println("Plantas regadas: " + this.getWatered() + "/" + this.getAlive()+ " " + Simulador.porcentaje(getWatered(), getAlive()) + "%");
        System.out.println("Plantas maduras: " + this.getMature() + "/" + this.getAlive() + " " + Simulador.porcentaje(getMature(), getAlive()) + "%");
    }else{
        System.out.println("Plantas vivas: " + 0 + "/" + this.getNum() +" 0%" );
        System.out.println("Plantas regadas: " + 0 + "/" + 0 + " 0%");
        System.out.println("Plantas maduras: " + 0 + "/" + 0 + " 0%");
    }

}
```

**showTileStatus**
llama a showStatus de cada planta o árbol

```java
public void showTileStatus() {
    boolean flag = true;
    for (Plantae plt : plantas) {
        if (plt != null) {
            flag = false;
            plt.showStatus();
        }
    }
    if (flag) {
        System.out.println();
        System.out.println(x:"Esto esta vacio");
    }
}
```

**waterCrops**

riega si hay agua suficiente y según el tipo de planta o árbol hace una lógica distinta. Devuelve el contador de plantas regadas y del agua gastada

```
public int[]  waterCrops(int agua) {
    int cont = 0;
    int gastado = 0;
    boolean flag = false;
    Random ran = new Random();
    for (Plantae plt : plantas) {
        flag = false;
        if (plt != null) {
            if (plt.isDead()) {
            }else{
                if (plt instanceof Autorregable) {
                    flag = plt.water();
                    if(flag){
                        gastado++;
                        cont++;
                        agua = agua-1;
                    }
                }

                if (plt instanceof Sumergida) {
                    if (agua > 4) {
                        flag = plt.water();
                        if(flag){
                            gastado = gastado + 5;
                            cont++;
                            agua = agua-5;
                        }
                    }
                }
```

```java
if (plt instanceof Secano) {
    if ( ran.nextInt(bound:4) == 0) {
        flag = plt.water();
        if (flag) {
            cont++;
        }
    }else{
        if (agua > 0) {
            flag = plt.water();
            if(flag){
                cont++;
                gastado++;
                agua = agua-1;
            }
        }
    }
}

if (plt instanceof Sedienta || plt instanceof SedientaCiclo) {
    if (agua > 1) {
        flag = plt.water();
        if(flag){
            cont++;
            gastado = gastado + 2;
            agua = agua-2;
        }
    }

}
```

```java
                if (agua > 0 && !flag) {
                    flag = plt.water();
                    if(flag){
                        cont++;
                        gastado++;
                        agua = agua-1;
                    }
                }else{
                    if (!flag) {
                        System.out.println(x:"no hay agua para regar");
                    }
                }
            }
        }
    }
    int[] res = {cont,gastado};
    return res;
}
```

**growCrops**
llama al método grow de cada planta o arbol

```java
public void growCrops() {
    for (Plantae plt : plantas) {
        if (plt != null) {
            plt.grow();
        }
    }
}
```

**upgrade**
crea un array de Plantae con 10 índices más que el actual para sustituirlo

```java
public void upgrade() {
    Plantae[] plantNuev = new Plantae[this.plantas.length + 10];
    System.arraycopy(this.plantas, srcPos:0, plantNuev, destPos:0, this.plantas.length);
    this.subirNivel();
    this.plantas = plantNuev;
}
```

**harvest**
llama al método harvest de cada planta. Cuenta las muertas en el proceso y las triturables

```java
public int[] harvest(Estadisticas estats) {
    int contReco = 0;
    int contMuer = 0;
    int contPlant = 0;
    int ganado = 0;
    int contTrit = 0;
    int prod = 0;
    int monedas = 0;
    for (Plantae planta : plantas) {
        if (planta != null) {
            prod = planta.harvest();
            if (planta.isDead()) {
                contMuer++;
            }
            if(prod != 0){
                estats.registrarCosecha(planta.getName(), prod);
                contReco = contReco + prod;
                contPlant++;
            }
            if (planta instanceof Triturable) {
                monedas = prod * planta.getGain();
                contTrit = contTrit + monedas;
            }else{
                ganado = ganado + prod * planta.getGain();
            }

        }
    }
    int res [] = {contReco , contMuer , contPlant , ganado , contTrit};
    return res;
}
```

**unroot**
convierte todas las plantas o arboles en null

```
public void unroot() {
    for (int i = 0; i < plantas.length; i++) {
        plantas[i] = null;
    }
}
```

**plow**
convierte las plantas o árboles muertas a null y suma la mitad del precio por cada una y al final devuelve el contador

```
public int plow() {
    int x= 0;
    for (int i = 0; i < plantas.length; i++) {
        if (plantas[i] != null) {
            if (plantas[i].isDead()) {
                x = x + (plantas[i].getPrice()/2);
                plantas[i] = null;
            }
        }
    }
    return x;
}
```

## Métodos Abstractos

```
public abstract void plants(int tipo);
```

```
public abstract void showCapacity();
```

```
public abstract void subirNivel();
```

# Clase Invernadero

## Métodos

**plants**
según el número que se le pase por parámetro crea un tipo de colanta y la añade en el primer hueco que encuentre

```java
@Override
public void plants(int tipo) {
        for (int i = 0; i < this.plantas.length; i++) {
            if (this.plantas[i] == null) {
                switch (tipo) {
                    case 1:
                        Trigo tri = new Trigo();
                        this.plantas[i] = tri;
                        break;
                    case 2:
                        Arroz arr = new Arroz();
                        this.plantas[i] = arr;
                        break;
                    case 3:
                        Garbanzos gg = new Garbanzos();
                        this.plantas[i] = gg;
                        break;
                    case 4:
                        Lechuga lec = new Lechuga();
                        this.plantas[i] = lec;
                        break;
                    case 5:
                        Patata pat = new Patata();
                        this.plantas[i] = pat;
                        break;
                    case 6:
                        Pimientos pim = new Pimientos();
                        this.plantas[i] = pim;
                        break;
                }
                break;
            }
        }
    }
```

**subirNivel**

si el nivel es 10 ya no sube mas, si no es diez suma uno

```java
@Override
public void subirNivel() {
    if (nivel == 10) {
        System.out.println(x:"Ya está al nivel maximo!!!");
    }else{
        nivel++;
        System.out.println("El invernadero " + name + " sube a nivel " + nivel + "!!!");
    }
}
```

**showCapacity**

muestra por pantalla sus estadísticas

```java
@Override
public void showCapacity() {
    System.out.println("invernadero " + this.name + " " + Simulador.porcentaje(getNum(), getTiles()) + "% de capacidad " + this.getNum() + "/" + this.getTiles());
}
```

# Clase Arboleda

## Atributos

```java
/**
 * si tiene la mejora de sistema de riego por goteo
 */
private boolean goteo = false;
```

## Constructor

El atributo plantas que hereda de huerto es ahora un array de arboles

```java
public Arboleda(String name) {
    super(name);
    this.plantas = new Arbol[10];
}
```

## Métodos

**getters**

```java
public boolean isGoteo(){
    return goteo;
}
```

**gotear**

convierte goteo a true

```java
public void gotear(){
    System.out.println("Añadido sistema de riego por goteo en " + super.getName());
    goteo = true;
}
```

**getPrice**
con un contador suma todos los precios de los árboles y lo devuelve al final

```
public int getPrice(){
    int res = 0;
    for (Plantae arb : plantas) {
        if (arb != null) {
            res = res + arb.getPrice();
        }
    }
    return res;
}
```

**subirNivel**
si el nivel es 10 ya no sube mas, si no es diez suma uno

```
@Override
public void subirNivel() {
    if (nivel == 10) {
        System.out.println(x:"Ya está al nivel maximo!!!");
    }else{
        nivel++;
        System.out.println("La arboleda " + name + " sube a nivel " + nivel + "!!!");
    }
}
```

**plants**
según el número que se le pase por parámetro crea un tipo de colanta y la añade en el primer hueco que encuentre

```java
@Override
public void plants(int tipo) {
    for (int i = 0; i < this.plantas.length; i++) {
        if (this.plantas[i] == null) {
            switch (tipo) {
                case 1:
                    Kiwi ki = new Kiwi();
                    this.plantas[i] = ki;
                    break;
                case 2:
                    Madronho mad = new Madronho();
                    this.plantas[i] = mad;
                    break;
                case 3:
                    Manzano man = new Manzano();
                    this.plantas[i] = man;
                    break;
                case 4:
                    Melocotonero mel = new Melocotonero();
                    this.plantas[i] = mel;
                    break;
                case 5:
                    Naranjo nar = new Naranjo();
                    this.plantas[i] = nar;
                    break;
            }
            break;
        }
    }
}
```

**showCapacity**
muestra por pantalla sus estadísticas

```java
@Override
public void showCapacity() {
    System.out.println("Arboleda " + this.name + " " + Simulador.porcentaje(getNum(), getTiles()) + "% de capacidad " + this.getNum() + "/" + this.getTiles());
}
```

# Clase TanqueAgua

## Atributos

```java
/**
 * cantidad disponible de agua
 */

private int water = 10;

/**
 * cantidad de agua maxima actual
 */
private int maxWater = 20;
```

## Métodos

**getters**

```java
/**
 * @return cantidad disponible de agua
 */
public int getWater() {
    return water;
}

/**
 * @return cantidad de agua maxima actual
 */
public int getMax() {
    return maxWater;
}
```

**toString**

devuelve información del tanque en string

```java
public String toString(){
   return "Tanque de agua al " + Simulador.porcentaje(this.water, this.maxWater) + "% de su capacidad " + this.water + "/" + this.maxWater;
}
```

**showStatus**

muestra el método toString

```java
public void showStatus() {
    System.out.println(toString());
}
```

**addWater**
sí water mas el numero pasado por parámetro es menor que maxWater entonces iguala watar a maxwater, si es menor a water le suma el número pasado por parámetros

```
public int addWater(int water) {
    if (this.water + water > maxWater) {
        int a = this.water;
        this.water = this.maxWater;
        return this.maxWater - a;
    }else{
        this.water = this.water + water;
        return water;
    }
}
```

**upgrade**
suma 5 a maxWater

```
public void upgrade() {
    this.maxWater = this.maxWater + 5;
}
```

**usar**
le resta a water el número pasado por parámetros

```
public void usar(int num){
    this.water = this.water - num;
}
```

# Paquete Plantas

## Clase Abstracta Plantae

### Atributos

```java
/**
 *  coste
 */
protected int price;

/**
 * ganancia por producto
 */
protected int gain;

/**
 * maximo de productos
 */
protected int prodMax;

/**
 * minimo de productos
 */
protected int prodMin;

/**
 * dias que tarda en madurar
 */
protected int diasToMature;

/**
 * dias que tarda en morir
 */
protected int diasToDead;

 /**
 *nombre coloquial
 */
protected String name;

/**
 *nombre cientifico
 */
protected String scientificName;
```

```java
/**
 *contador de dias que lleva viva la planta
 */
protected int numDias = 0;

/**
 *está regada o no
 */
protected boolean watered = false;

/**
 *está muerta o no
 */
protected boolean dead = false;

/**
 *está madura o no
 */
protected boolean mature = false;
```

Constructor

```java
public Plantae(CultivoDatos cult){
    this.name  = cult.getNombre();
    this.scientificName = cult.getCientifico();
    this.price = cult.getCoste();
    this.gain = cult.getMonedas();
    this.diasToMature = cult.getMadura();
    this.diasToDead = cult.getMuerte();
    this.prodMin = cult.getProdMin();
    this.prodMax = cult.getProdMax();
}
```

## Métodos

**getters**

```java
public String getName() {
    return name;
}

public String getScientificName() {
    return scientificName;
}

public boolean isWatered() {
    return watered;
}

public boolean isDead() {
    return dead;
}

public boolean isMature() {
    return mature;
}

public int getPrice() {
    return price;
}

public int getGain() {
    return gain;
}

public String getProduct() {
    return name;
}
```

**showInfoNatura**

muestra un resumen de los atributos del tipo de la planta

```java
public void showInfoNatura() {
    System.out.println("==========" + this.name + "==========");
    System.out.println("Precio: " + this.price);
    System.out.println("Producto: " + getProduct());
    System.out.println("Número de productos: " + this.prodMax + "/" + this.prodMin);
    System.out.println("Monedas por producto: " + this.gain + "/" + "producto");
    System.out.println("Maduración: " + this.diasToMature);
    System.out.println("Tiempo de vida: " + this.diasToDead);
}
```

**toString**
devuelve el estado de la planta

```
public String toString() {
    return this.name + "-" + "Viva:" + (this.dead?"no":"si") + " | " + "Madura " + (this.mature?"si":"no") + " | " + "Regada: " + (this.watered?"si":"no");
}
```

**showStatus**
muestra información del estado de la planta

```
public void showStatus() {
    System.out.println("---------------" + this.name +  "---------------");
    System.out.println("Edad: " + this.numDias + " dias");
    System.out.println("Viva: " + (this.dead?"no":"si"));
    System.out.println("Regada: " + (this.watered?"si":"no"));
    System.out.println("Madura " + (this.mature?"si":"no"));
}
```

**water**
convierte watered a true

```
public boolean water() {
    if (this.dead) {
        return false;
    }
    if (!this.watered){
        this.watered=true;
        System.out.println(this.name + " está regada");
        return true;
    }
    return false;
}
```

**kill**
método auxiliar que pone a watered, y mature a false y dead a true

```
protected void kill() {
    this.watered = false;
    this.dead = true;
    this.mature = false;
}
```

**harvest**
si la planta está madura llama a kill y devuelve un numero entre el minimo y el máximo de productos

```
public int harvest(){
    if (this.mature) {
        kill();
        Random ran = new Random();
        return ran.nextInt(this.prodMin,this.prodMax);
    } else {
        return 0;
    }
}
```

**replant**

```
public void replant() {
    this.numDias = 0;
    this.dead = false;
    this.watered = false;
    this.mature = false;
}
```

## Métodos abstractos

```
public abstract void grow();
```

# Clase Abstracta Planta

hereda de Plantae

## Métodos

**grow**
si la planta no está regada hay una probabilidad de que muera, si diasToDead i numDias son iguales muere directamente, si diasToMature es igual a numDias madura la planta. Convierte watered a false y suma un dia

```java
@Override
public void grow() {
    if (!dead){
        ++this.numDias;
        if (!this.watered || this.numDias == this.diasToDead) {
            Random ran = new Random();
            if ( ran.nextBoolean() || this.numDias == this.diasToDead){
                kill();
                System.out.println(name + " ha muerto");
            }
        }
        if (this.numDias == this.diasToMature && !dead) {
            this.mature = true;
            System.out.println( name + " ha madurado");
        }
        this.watered = false;
    }else{
        System.out.println("no puedes hacer esto porque " + name + " está muerta");
    }
}
```

# Clase Abstracta PlantaCiclo

hereda de Plantae

## Atributos

```java
/**
 * contador de los dias que lleva en el ciclo para replantarse
 */
protected int diasCiclo = 0;

/**
 * numero de dias que tarda en replantarse despues de morir
 */
protected int ciclo;
```

## Constructor

iguala ciclo a getCiclo del objeto CultivoDatos que se pasa por parametros

```java
public PlantaCiclo(CultivoDatos cult) {
    super(cult);
    this.ciclo = cult.getCiclo();
}
```

## Métodos

**showInfoNatura**
añade información de los ciclos

```java
@Override
public void showInfoNatura() {
    super.showInfoNatura();
    System.out.println("Ciclo: " + ciclo + " días");
}
```

**harvest**
si el harvest de plantae devuelve algo distinto de 0 convierte dead a false, watered a true y diasCiclo a 0

```java
@Override
public int harvest() {
    int x = super.harvest();
    if (x == 0) {
        return 0;
    }
    dead = false;
    watered = true;
    diasCiclo = 0;
    return x;

}
```

**grow**
es como grow de plantae pero si está muerta y diasCiclo es igual a Ciclo se replanta

```java
@Override
public void grow() {
    ++this.numDias;
    ++diasCiclo;
    if (!this.watered || this.numDias == this.diasToDead) {
        Random ran = new Random();
        if (ran.nextBoolean() || this.numDias == this.diasToDead){
            kill();
            diasCiclo = 0;
            System.out.println(name + " ha muerto");
        }

        if (diasCiclo == this.diasToMature && !dead) {
            this.mature = true;
            System.out.println( name + " ha madurado");
        }
        this.watered = false;
    }else{
        if (diasCiclo == ciclo) {
            replant();
            diasCiclo = 0;
            System.out.println(name + " se ha replantado");
        }else{
            System.out.println(name + " está muerta pero lleva " + diasCiclo + "/" + ciclo + " para replantarse!!!");
        }
    }
}
```

# Paquete Plantas.Tipo

## Clase Hoja

hereda de Planta

### métodos

**grow**
si no se riega muere directamente

```java
@Override
public void grow() {
    if (!dead){
        ++this.numDias;
        if (!this.watered || this.numDias == this.diasToDead) {
                kill();
                System.out.println(name + " ha muerto");
            }

        if (this.numDias == this.diasToMature && !dead) {
            this.mature = true;
            System.out.println( name + " ha madurado");
        }
        this.watered = false;
    }else{
        System.out.println("no puedes hacer esto porque " + name + " está muerta");
    }
}
```

# Clase Secano

hereda de Planta

Su implementación sirve para usar el instanceof en el método waterCrops de invernadero

# Clase Sedienta

hereda de Planta

## métodos

**grow**
la probabilidad de morir si no se riega es de 1/4

```java
@Override
public void grow() {
    if (!dead){
        ++this.numDias;
        if (!this.watered || this.numDias == this.diasToDead) {
            Random ran = new Random();
            if (ran.nextInt(bound:4) == 0 || this.numDias == this.diasToDead){
                kill();
                System.out.println(name + " ha muerto");
            }
        }
        if (this.numDias == this.diasToMature && !dead) {
            this.mature = true;
            System.out.println( name + " ha madurado");
        }
        this.watered = false;
    }else{
        System.out.println("no puedes hacer esto porque " + name + " está muerta");
    }
}
```

# Clase SedientaCiclo

hereda de PlantaCiclo

## métodos

**grow**
la probabilidad de morir si no se riega es de 1/4  y ademas implementa los ciclos

```java
@Override
public void grow() {
    ++this.numDias;
    ++diasCiclo;
    if (!this.watered || this.diasCiclo == this.diasToDead) {
        Random ran = new Random();
        if (ran.nextInt(bound:4) != 0 || this.numDias == this.diasToDead){
            kill();
            diasCiclo = 0;
            System.out.println(name + " ha muerto");
        }

        if (diasCiclo == this.diasToMature && !dead) {
            this.mature = true;
            System.out.println( name + " ha madurado");
        }
    }else{
        if (this.dead) {
            if (diasCiclo == ciclo ) {
                replant();
                diasCiclo = 0;
                System.out.println(name + " se ha replantado");
            }else{
                System.out.println(name + " está muerta pero lleva " + diasCiclo + "/" + ciclo + " para replantarse!!!");
            }
        }
    }
    this.watered = false;
```

## Clase Tuberculo

hereda de Planta

### métodos

**harvest**
tiene una probabilidad de 1/2  de replantarse después de ser cosechada

```java
@Override
public int harvest() {
    int res = super.harvest();
    Random ran = new Random();
    if (ran.nextBoolean()) {
        replant();
        System.out.println(name + " se ha replantado sola");
    }
    return res;
}
```

## Interfaces bandera

### Hidroponica

### Industrial

### Sumergida

### Triturable

# Paquete Plantas.Finales

## Clase Arroz

hereda de Planta implementa sumergida y triturable

### métodos

**grow**
probabilidad de 3/4  de morir si no se riega

```
    @Override
    public void grow() {
        if (!dead){
            ++this.numDias;
            if (!this.watered || this.numDias == this.diasToDead) {
                Random ran = new Random();
                if ( ran.nextInt(bound:4) != 0|| this.numDias == this.diasToDead){
                    kill();
                    System.out.println(name + " ha muerto");
                }
            }
            if (this.numDias == this.diasToMature && !dead) {
                this.mature = true;
                System.out.println( name + " ha madurado");
            }
            this.watered = false;
        }else{
            System.out.println("no puedes hacer esto porque " + name + " está muerta");
        }
    }
}
```

## Clase Garbanzos

hereda de Secano

## Clase Lechuga

hereda de hoja

## Clase Patata

hereda de Tubérculo implementa Industrial

## Clase Pimientos

hereda de SedientaCiclo

## Clase Trigo

hereda de Planta implementa triturable

# Paquete Árbol

## Clase Arbol

### Atributos

```
/**
 * contador de los dias que lleva en el ciclo para replantarse
 */
protected int diasCiclo = 0;

/**
 * numero de dias que tarda en replantarse despues de morir
 */
protected int ciclo;
```

```
/**
 * se pone true cuando madura por primera vez
 */
protected boolean growed = false;
```