TANQUE DE AGUA

Se quiere crear una aplicación para simular proceso de llenar y vaciar un tanque de agua empleado para riego de campos, que está situado cerca de un pantano.

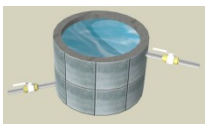
El llenado y vaciado de agua del tanque depende de dos factores principales:

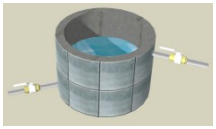
- Es estado de sequía de la región en la que está situado el pantano, y
- El grado de humedad del suelo del campo a regar.

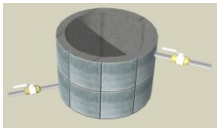
Por estas razones se deberán tener en cuenta una serie de reglas que deberán cumplirse a la hora de llenar y vaciar el tanque de agua y que la aplicación deberá validar si se cumplen o no.

La pantalla principal de la aplicación (entregada) es la siguiente:

La clase java “Tanque” (entregada) tiene 3 JLabel con las imágenes del taque “VACIO”, “MEDIO” y ”LLENO”


jlabelLleno


jLabelMedio


jLabelVacio

1) Crea una nueva clase de tipo enumerado (**enum**) llamado **EstadoTanque** en el paquete **net.ausiasmarch.tanque.modelo** que tenga los siguientes valores:

   VACIO, MEDIO y LLENO

1) En la clase Tanque, añade los siguientes métodos y propiedades:

Método y propiedades	**Descripción**
EstadoTanque estado	Propiedad de tipo EstadoTanque con el estado en el que se encuentra el Tanque.
Tanque() (YA ESTA HECHO)	Añade el código necesario para poner el estado a "VACIO" y hacer visible la etiqueta correspondiente e invisible las otras dos. Por ejemplo, para hacer visible la etiqueta `jLabelVacio`, haremos `jLabelVacio.SetVisible(true);`
void llenarMedio()	Llena el tanque, dependiendo de cuál sea el `estado` actual. Si esta VACIO lo pone MEDIO y si está MEDIO lo pone LLENO. En estos casos, hace visible e invisible las etiquetas correspondientes. Si está LLENO o no está en ninguno de los estados no hace nada.
void vaciarMedio()	Vacía el tanque, dependiendo de cuál sea el `estado` actual. Si está LLENO lo pone MEDIO y si está MEDIO lo deja VACIO. En estos casos hace visible e invisible las etiquetas correspondientes. Si está VACIO o no está en ninguno de los estados anteriores no hace nada.
EstadoTanque getEstado()	Obtiene el estado el Tanque.

3) Añade una nueva clase llamada `ControlRiego` en el paquete net.ausiasmarch.tanque.modelo

Esta nueva clase, que validará el estado del tanque, añade los siguientes métodos y propiedades:

Método y propiedades	**Descripción**
String mensaje	Contendrá el último mensaje de error
int sequia	Contendrá el nivel de sequía
int humedad	Contendrá el nivel de humedad del suelo
EstadoTanque estado	Estado del tanque de tipo EstadoTanque (el enumerado)
ControlRiego(EstadoTanque estado)	Constructor que asigna a la propiedad `estado` el parámetro objeto `estado` de tipo `EstadoTanque`, del tanque que se quiere validar
ControlRiego()	Constructor sin parámetros
int `permitidoLLenar`()	Realiza la validación de las reglas de llenado del tanque. Retornará 0 si es posible llenar una nueva mitad del tanque en función de su estado y el nivel de sequía. Sino retornará un número mayor que 0 y pondrá el mensaje de error correspondiente en la propiedad mensaje. Reglas del **llenado** desde el pantano. Los mensajes de error para cada valor 1 a 4 son: 1: El valor de la sequía no puede ser mayor que 10 o menor que 0 2: No se puede llenar el tanque si está lleno 3: No se puede llenar si la sequía es 10. 4: No se puede llenar más de la mitad si la sequía es mayor o igual a 7.

int permitidoVaciar()	Realiza la validación de las reglas de vaciado del tanque. Retornará 0 si es posible llenar una nueva mitad del tanque en función de su estado y el nivel de humedad. Sino retornará un número mayor que 0 y pondrá el mensaje de error correspondiente en la propiedad mensaje Reglas de **vaciado** para riego. Los mensajes de erros para cada valor 1 a 4 son: 1: El valor de la humedad del suelo no puede ser mayor que 10 o menor que 0 2: No se puede vaciar NADA para regar si el tanque ya está vacío. 3: No se puede vaciar nada para regar si la humedad del suelo es mayor o igual a 7 4: No se puede vaciar más allá de la mitad para regar si la humedad del suelo es mayor o igual a 3 y menor que 7.
String getMansaje()	Retorna el mensaje de la propiedad mensaje

Añade también los métodos get y set de las propiedades sequia, humedad y estado.

4) En la clase Main:

4.1) Declara una propiedad privada llamada controlRiego de la clase ControlRiego

4.2) En el constructor de Main, después de la línea initComponents()

Crea un objeto de la clase ControlRiego, pasando al constructor ControlRiego(EstadoTanque estado)el estado actual del tanque, y asígnalo a la variable controlRiego. El estado actual del tanque se obtiene con el método getEstado() del tanque.

4.3) Modifica los manejadores de eventos de los botones jButtonLLenarTanque y jButtonVaciarTanque con las siguientes acciones:

<u>En el botón “Llenar Medio”</u>

a) Declara una variable llamada sequia de tipo int

b) Valida el dato entrada del campo de texto correspondiente al nivel de sequía para que sea un número entero, usando el método apropiado de la clase Convert. Si el valor no es válido mostrar un mensaje usando el método mensaje de Main y salir haciendo return.

c) Convierte el campo de texto de entrada de nivel de sequía a entero y asígnalo a la variable sequia

d) Asigna sequia a la propiedad respectiva del objeto controlRiego con el método set.

e) Llama al método de validación de llenado permitidoLLenar() del objeto controlRiego. Si devuelve un valor distinto de cero, llama al método getMensaje() de controlRiego para mostrar el mensaje del error ocurrido usando el método mensaje de Main y salir haciendo return.

Si todo es correcto llama al método llenarMedio() del objeto tanque y modifica el estado del objeto controlRiego pasandole, a su metodo setEstado el estado actual del tanque, que podemos obtener con su método getEstado()

.

En el botón “Vaciar Medio”

a) Declara una variable llamada humedad de tipo int

b) Valida el dato entrada del campo de texto correspondiente al nivel de humedad para que sea un entero, usando el método apropiado de la clase `Convert`. Si el valor no es válido mostrar un mensaje del error usando el método `mensaje` de `Main` y salir con `return`.

c) Convierte el campo de texto de entrada de nivel de humedad a entero y asígnalo a la variable humedad

d) Asigna humedad a la propiedad respectiva del objeto `controlRiego` con el método set.

e) Llama al método de validación de vaciado `permitidoVaciar()` del objeto `controlRiego`. Si devuelve un valor distinto de cero, llama al método `getMensaje()` de `controlRiego` para mostrar el mensaje del error ocurrido usando el método `mensaje` de `Main` y luego sal haciendo `return`.

Si todo es correcto llama al método **`vaciarMedio()`** del objeto `tanque` y modifica el estado del objeto `controlRiego` , pasándole a su método `setEstado` el estado actual º del tanque, que podemos obtener con su método **`getEstado()`**