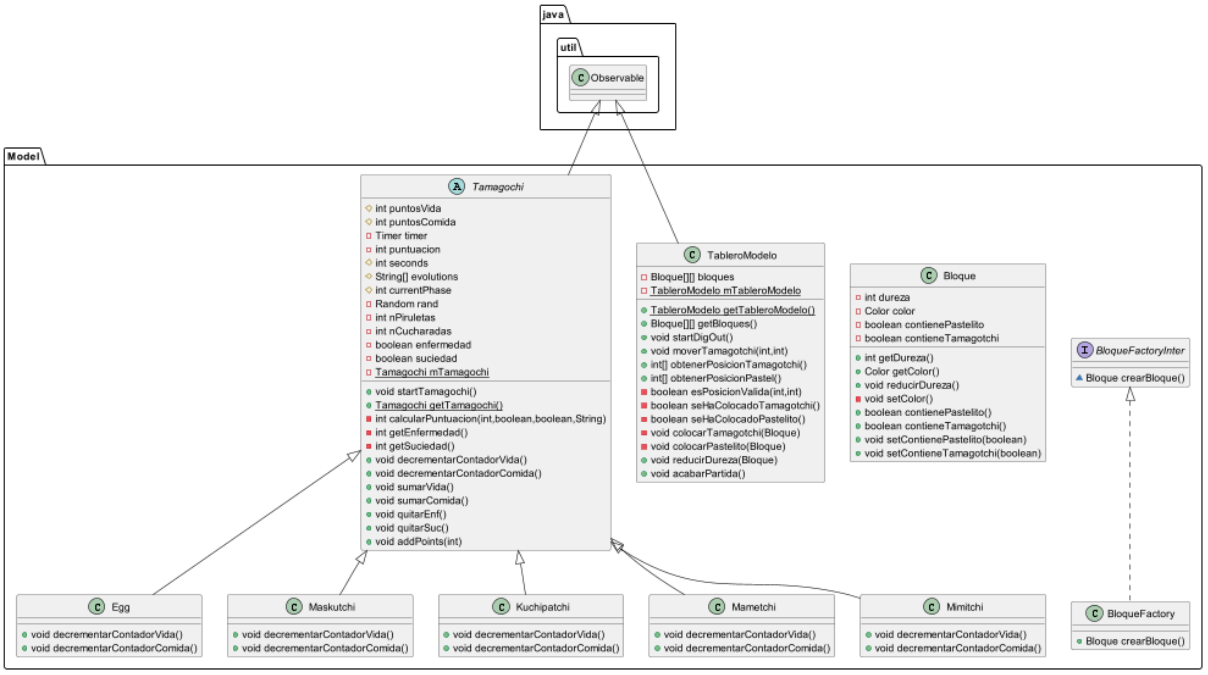
Tamagotchi Grupo Fracaso

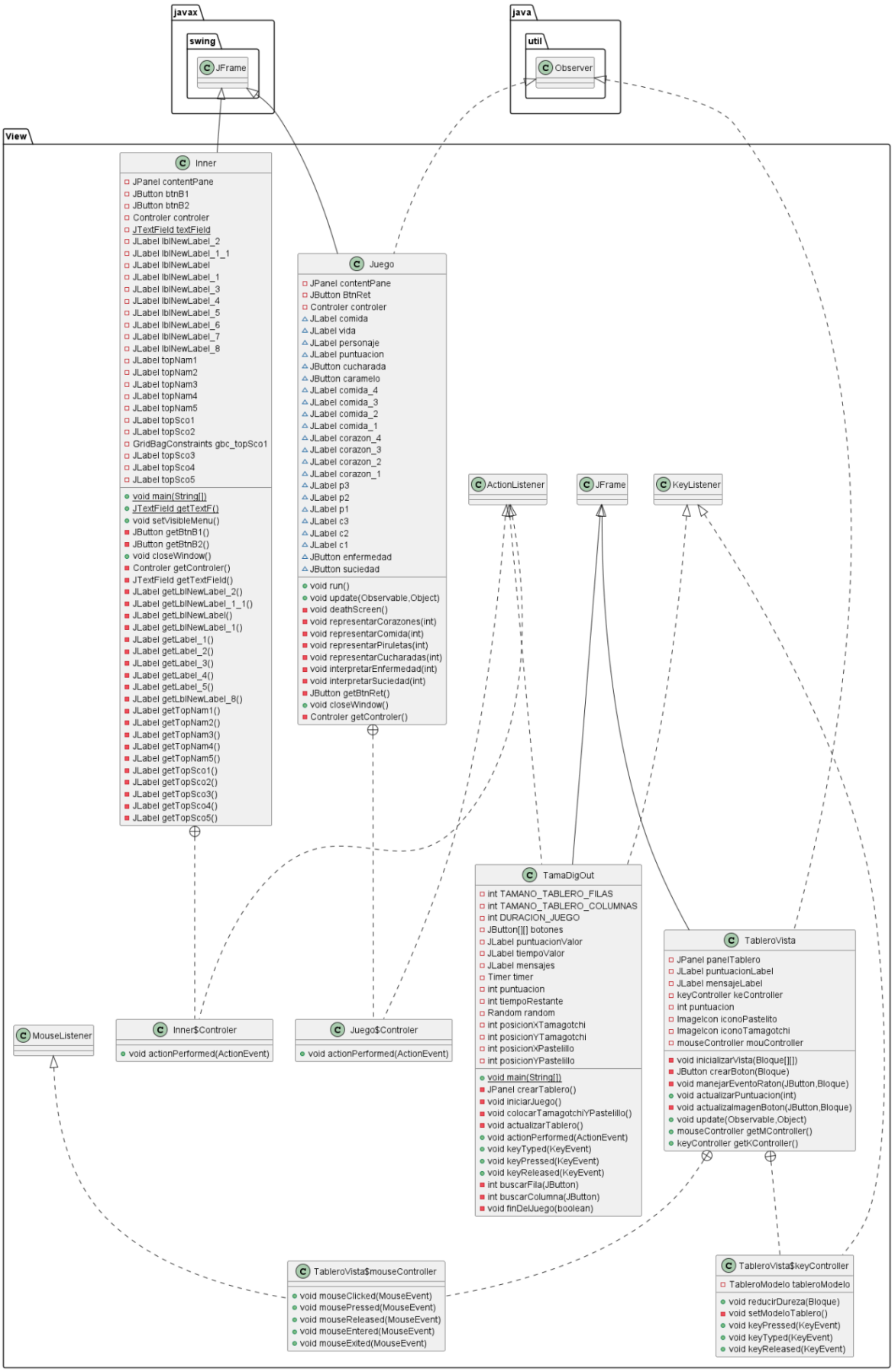
Este dossier es la presentación y explicación del desarrollo de nuestro proyecto Tamagotchi a lo largo del primero de los tres Sprints que tienen como objetivo una aplicación funcional que imita al mítico juego. Hemos entendido que la aplicación tiene tres partes esenciales: el menú, el juego y el minijuego. Por ello hemos decidido asignar dos personas a cada sección …

Diagrama de clases

Modelo:



Vista:


javax
swing
JFrame
java
util
Observer
View
Inner
JPanel contentPane
JButton btnB1
JButton btnB2
Controler controler
JTextField textField
JLabel lblNewLabel_2
JLabel lblNewLabel_1_1
JLabel lblNewLabel
JLabel lblNewLabel_1
JLabel lblNewLabel_3
JLabel lblNewLabel_4
JLabel lblNewLabel_5
JLabel lblNewLabel_6
JLabel lblNewLabel_7
JLabel lblNewLabel_8
JLabel topNam1
JLabel topNam2
JLabel topNam3
JLabel topNam4
JLabel topNam5
JLabel topSco1
JLabel topSco2
GridBagConstraints gbc_topSco1
JLabel topSco3
JLabel topSco4
JLabel topSco5
void main(String[])
JTextField getTextF()
void setVisibleMenu()
JButton getBtnB1()
JButton getBtnB2()
void closeWindow()
Controler getControler()
JTextField getTextField()
JLabel getLblNewLabel_2()
JLabel getLblNewLabel_1_1()
JLabel getLblNewLabel()
JLabel getLblNewLabel_1()
JLabel getLabel_1()
JLabel getLabel_2()
JLabel getLabel_3()
JLabel getLabel_4()
JLabel getLabel_5()
JLabel getLblNewLabel_8()
JLabel getTopNam1()
JLabel getTopNam2()
JLabel getTopNam3()
JLabel getTopNam4()
JLabel getTopNam5()
JLabel getTopSco1()
JLabel getTopSco2()
JLabel getTopSco3()
JLabel getTopSco4()
JLabel getTopSco5()
Juego
JPanel contentPane
JButton BtnRet
Controler controler
JLabel comida
JLabel vida
JLabel personaje
JLabel puntuacion
JButton cucharada
JButton caramelo
JLabel comida_4
JLabel comida_3
JLabel comida_2
JLabel comida_1
JLabel corazon_4
JLabel corazon_3
JLabel corazon_2
JLabel corazon_1
JLabel p3
JLabel p2
JLabel p1
JLabel c3
JLabel c2
JLabel c1
JButton enfermedad
JButton suciedad
void run()
void update(Observable,Object)
void deathScreen()
void representarCorazones(int)
void representarComida(int)
void representarPiruletas(int)
void representarCucharadas(int)
void interpretarEnfermedad(int)
void interpretarSuciedad(int)
JButton getBtnRet()
void closeWindow()
Controler getControler()
ActionListener
JFrame
KeyListener
TamaDigOut
int TAMANO_TABLERO_FILAS
int TAMANO_TABLERO_COLUMNAS
int DURACION_JUEGO
JButton[][] botones
JLabel puntuacionValor
JLabel tiempoValor
JLabel mensajes
Timer timer
int puntuacion
int tiempoRestante
Random random
int posicionXTamagotchi
int posicionYTamagotchi
int posicionXPastelillo
int posicionYPastelillo
void main(String[])
JPanel crearTablero()
void iniciarJuego()
void colocarTamagotchiYPastelillo()
void actualizarTablero()
void actionPerformed(ActionEvent)
void keyTyped(KeyEvent)
void keyPressed(KeyEvent)
void keyReleased(KeyEvent)
int buscarFila(JButton)
int buscarColumna(JButton)
void finDelJuego(boolean)
TableroVista
JPanel panelTablero
JLabel puntuacionLabel
JLabel mensajeLabel
keyController keController
int puntuacion
ImageIcon iconoPastelito
ImageIcon iconoTamagotchi
mouseController mouController
void inicializarVista(Bloque[][])
JButton crearBoton(Bloque)
void manejarEventoRaton(JButton,Bloque)
void actualizarPuntuacion(int)
void actualizaImagenBoton(JButton,Bloque)
void update(Observable,Object)
mouseController getMController()
keyController getKController()
MouseListener
Inner$Controler
void actionPerformed(ActionEvent)
Juego$Controler
void actionPerformed(ActionEvent)
TableroVista$mouseController
void mouseClicked(MouseEvent)
void mousePressed(MouseEvent)
void mouseReleased(MouseEvent)
void mouseEntered(MouseEvent)
void mouseExited(MouseEvent)
TableroVista$keyController
TableroModelo tableroModelo
void reducirDureza(Bloque)
void setModeloTablero()
void keyPressed(KeyEvent)
void keyTyped(KeyEvent)
void keyReleased(KeyEvent)

Controladores

En esta aplicación para comunicar la vista y el modelo usamos una clase privada en la propia vista que realiza este proceso, a esto lo llamamos controlador. Hemos detectado 2 controladores, el primero ha sido en la parte visual del juego Tamagotchi y la otra en la parte visual del juego TamaDigOut.

En la parte del Tamagotchi se usa total y solamente para la interacción con los botones. Son 5: BOTÓN VIDA, BOTÓN COMIDA, BOTÓN ENFERMEDAD, BOTÓN SUCIEDAD Y BOTÓN RETURN.
El controlador en este caso se añade como actionListener a los botones y cuando recibe la interacción realiza la llamada al modelo correspondiente.

Por otra parte, En el TamaDigOut hemos separado las responsabilidades de cada controlador, es decir Un controlador -Controlador- que implementa un KeyListener y otro -BotonMouseListener- que implementa un MouseListener.

Responsabilidades:
La clase Controlador implementa la interfaz KeyListener.
El método keyPressed() recibe las pulsaciones de teclas del usuario.
Se utiliza un switch-case para determinar la acción a realizar según la tecla presionada que se traduce en llamada al método moverTamagotchi() del modelo.

La clase BotonMouseListener implementa la interfaz MouseListener.
Y se encarga de Detectar cuando el usuario pasa el raton por un bloque mediante el metodo mouseEntered() y reducir la dureza del bloque en el modelo.

Observer

Observer(TableroVista) : Es la Vista que observa al Observable y se actualiza cuando cambia su estado. Tiene el método ***update(Observable o, Object arg)*** para recibir la notificación y actualizar el estado interno.

Observable(TableroModelo) : Es el Modelo que mantiene el estado y puede ser observado.
Tiene dos métodos:
addObserver(Observer observer) : Registra un observador.

notifyObservers():Notifica a todos los observadores sobre un cambio en el estado.

Notify/Update

Minijuego:
Notify : El método notifyObservers() es parte de la clase Observable que es en nuestro caso -TableroModelo-. Este método notifica a todos los observadores registrados que son en nuestro caso solo una clase -TableroVista- que ha habido un cambio en el estado del modelo.

Update : El método update() se implementa en la clase Observer -TableroVista-. Este método define como el observador responde ante un cambio en el modelo. Cuando se llama a update(), la vista se actualiza para reflejar los cambios en el modelo (esto incluye acciones de mover el tamagotchi, reducir la duresa de un bloque en el minijuego).
El update() recibe 2 argumentos :

- o : El observable TableroModelo que ha enviado la notificacion
- arg : Un objeto que puede contener informacion adicional sobre el cambio de estado. El observador TableroVista utiliza esta informacion para actualizar su estado interno.
  El metodo update() de TableroVista recibe el TableroModelo com arg. TableroVista utiliza el TableroModelo para actualizar la intefaz grafica.

Tamagotchi:
Dentro de la parte del juego principal el observable es la mae Tamagotchi.java esta es observada por juego.java, la parte visual. Dentro del metodo que ejecuta el reloj interno del juego se realizan todos los notifyObservers(), y existe un notify generico que envía el estado general del tamagotchi (vida, comida, evolucion…) y luego hay otros notifyObservers() más especiales como el que envía un String[] = {“TamaDigOut”} el cual sirve para notificar del comienzo del minijuego. Por otro lado tenemos la conexión entre el minijuego y el tamagotchi la cual utiliza también el patrón observer pero se aprovecha de un método que es llamado por la lógica del minijuego, la MAE TableroModelo.

Reparto trabajo

A la hora de repartir el proyecto vimos claras 3 partes a diseñar: el menú de inicio, el propio juego Tamagotchi y el minijuego TamaDigOut. Ya que un compañero fue inexistente, Ivan Andres e Iker López hicieron el menú principal. Jon Gete se encargó de crear el timer de la MAE Tamagotchi, Iker Lopez desarrolló todo el resto de la lógica del Tamagotchi al igual que la parte visual y todas las interacciones. Por otro lado Abdessamad realizó el código del minijuego TamaDigOut, aunque otra gran parte del minijuego era la implementación en el resto del proyecto este arduo y extenso proceso que llevo unas cuantas horas fue realizado por Ivan Andrés e Iker López. Los diagramas fueron realizados por Ivan Andrés. La persona que organizó los subgrupos y suministró ayuda al resto del equipo en cuanto a decisiones que le planteaban fue Iker López.

	Tareas	MENU	JUEGO TAMA	JUEGO DIGOUT
	Iker López	30%	90%	(implementacion)10%
	Iván Andrés	70%		(implementacion)10%
	Nicolas Pasucal			5%
	Abdessamad			75%
	Jon Gete		10%	
	Diego Pomares	x	x	x
HORAS TOTALES POR TAREA		3 HORAS	12 HORAS	12 HORAS (DEV + IMPLEMENT)