EXAMEN 1º PRÁCTICAS PDOO **8/11/2021**

Nombre: _____ Grupo de prácticas: A3

Entrega: A través de PRADO, cuatro ficheros:

- Un archivo comprimido con el proyecto Java modificado.
- Una captura de pantalla con TODA la ventana de Netbeans una vez iniciado el juego.
- Un archivo comprimido con el proyecto Ruby modificado.
- Captura de pantalla de la consola una vez terminada una ejecución del juego del laberinto[1].

Duración: 90 minutos.

Descripción:

JAVA (Civitas): Reglas del juego adicionales:

- Las 14 calles están agrupadas en tríos o cuartetos (dos tríos y dos cuartetos) que forman barrios de distintos grados de sostenibilidad ecológica: baja, media y alta, y que determinan el porcentaje del impuesto ECO que debe añadirse al precio de la compra de las viviendas convencionales, usando la siguiente fórmula:

  precioCompraConImpuesto=precioCompra(1+porcentajeImpuestoECO/100)*

  siendo el *porcentajeImpuestoECO* el siguiente:

  - Barrios de sostenibilidad baja: 12%
  - Barrios de sostenibilidad media: 8%
  - Barrios de sostenibilidad alta: 0% (es decir, exentos del pago del impuesto)
- El precio del alquiler también está afectado de la misma forma por el impuesto ECO, aunque su precio aumenta en la mitad del *porcentajeImpuestoECO*.
- El impuesto ECO lo recauda la Hacienda pública.

Diagramas adicionales:

- Diagrama de clases *DCCivitasExamen1_A3*
- Diagrama de secuencias *Jugador.comprarExamen1_A3*

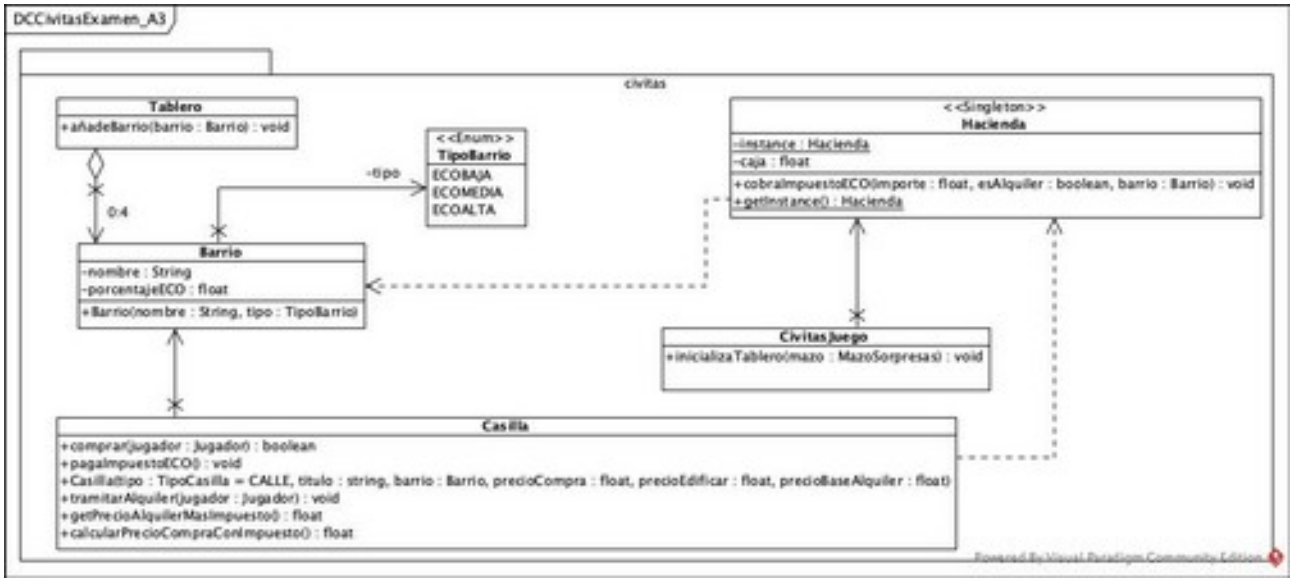
Tarea a realizar:

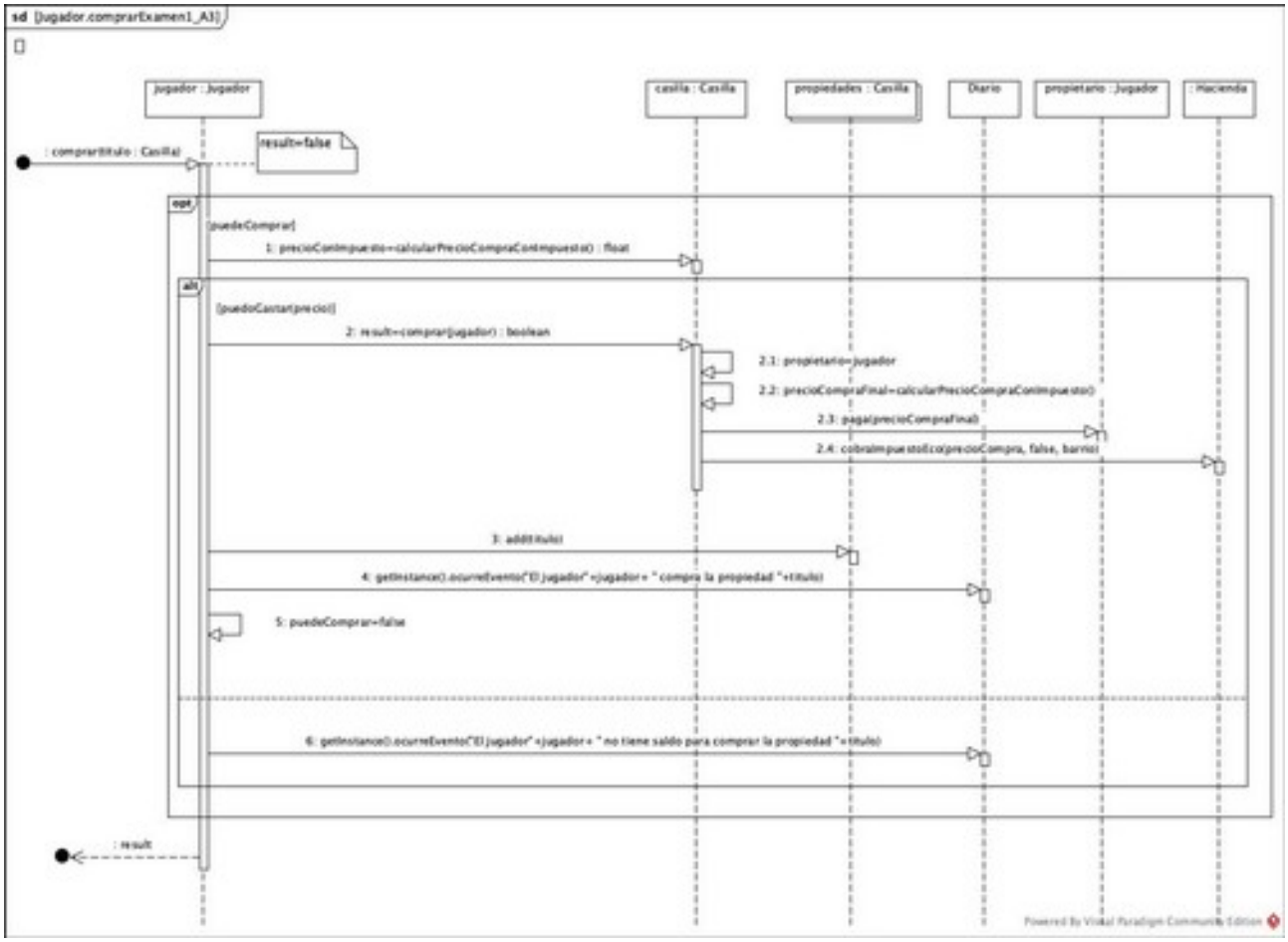
Añade al paquete *civitas* los elementos que aparecen en el diagrama de clases *DCCivitasExamen1_A3*, implementando los métodos según las reglas del juego adicionales y el diagrama de secuencias *Jugador.comprarExamen1_A3*. Deberás tener una versión completa del juego que funcione de acuerdo a las nuevas reglas. Añade los consultores y modificadores básicos necesarios, modifica/añade los métodos *toString* que se requieran y prueba el juego.

A continuación se detallan algunos pasos a realizar, que puedes seguir para facilitar el desarrollo del examen:

1. Para ejecutar un fichero Ruby *fichero.rb* desde la consola estando en la carpeta donde está el fichero, basta poner:
> ruby *fichero.rb*

1. Enumerado TipoBarrio.- Construye la clase con los valores del enumerado.
2. Clase Barrio.- Crea la clase con los atributos, constructor, método *toString y* consultor del atributo tipo.
3. Clase Hacienda, singleton.- Defínela (atributos y métodos) de forma correcta para que sea Singleton.
4. Clase Hacienda, método *cobraImpuestoEco*.- Debe incrementar la *caja* aplicando el porcentaje ECO al precio a pagar, según el barrio que se proporciona y si se trata de una compra (argumento *esAlquiler* a falso) o un alquiler (argumento *esAlquiler* a true).
5. Clase Casilla.- Modifica el constructor de calles, añade el nuevo atributo de referencia y modificador del mismo y modifica de forma consecuente el método *toString*.
6. Clase Tablero, método *añadeBarrio*.- Debe añadir el barrio a la lista de barrios.
7. Clase CivitasJuego, método *inicializaTablero* .- Crea 4 barrios y añádelos al tablero. Modifica la creación de las calles para que se les asigne un barrio (habrá dos barrios de 3 calles y otros dos de 4 calles).
8. Clase Casilla, método *calcularPrecioCompraConImpuesto*.- Debe incrementar al precio de compra lo que corresponda por impuesto ECO según la fórmula de la primera regla del juego adicional.
9. Clase Casilla, método *calcularPrecioAlquilerConImpuesto*.- Debe incrementar al precio de alquiler (obtenido con el método *getPrecioAlquilerCompleto*) lo que corresponda por impuesto ECO según las reglas del juego adicionales (la mitad que si fuera por compra).
10. Clase Jugador y clase Casilla, método *comprar*.- Modifica estos métodos siguiendo el diagrama de secuencias *Jugador.comprarExamen1_A3.*
11. Clase Casilla, método *tramitarAlquiler*.- Modifica el método para que el jugador pague un importe que incluya el impuesto ECO (método *calcularPrecioAlquilerConImpuesto*), y llame al método *cobraImpuestoEco* de la hacienda pública pasando como argumentos el importe del alquiler (sin impuesto), el valor true y el barrio al que pertenece la calle de la que se va a pagar el alquiler.
12. Añade consultores y modificadores simples a todos los atributos añadidos si te hacen falta.
13. Clase JuegoTexto, método *main*.- Prueba el programa y haz el resto de cambios necesarios para que el programa funcione de forma adecuada.





RUBY (laberinto):

Nuevas reglas del juego: Para no morir nunca dentro del laberinto, vamos a cambiar las reglas del juego de forma que, si el número de intentos fallidos (cada vez que método *pasar* devuelve false es un intento fallido) es igual a las vidas iniciales menos 1, cambie la pared del último intento por una puerta. Para ello, haz los cambios en los métodos y atributos de las clases del diagrama adjunto. Puedes seguir los siguientes pasos:

1) Añade un atributo de clase a Habitacion (total_intentos_fallidos)

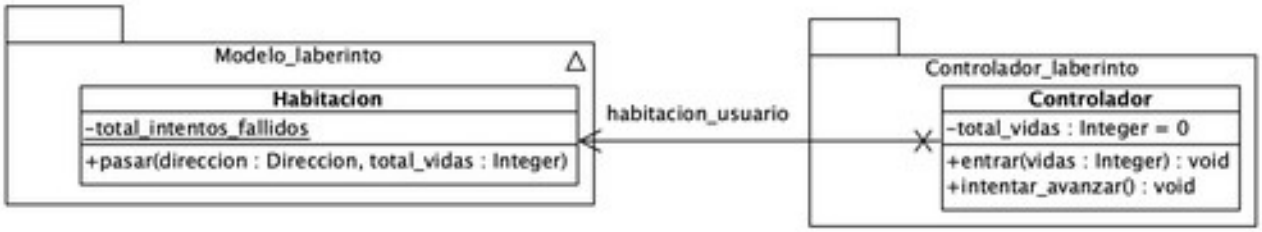
2) Modifica el método *pasar* de Habitacion, añadiéndole un segundo argumento con el total de vidas iniciales del jugador, de forma que, si hay una pared en la dirección del argumento incremente en 1 total_intentos_fallidos y si total_intentos_fallidos es mayor o igual que (total_vidas-1), cree una puerta "encantada"[2] y devuelva true.

3) Modifica el método *entrar* del Controlador, de forma que se inicialice también la variable total_vidas con el valor del argumento.

4) Modifica el método *intentar_avanzar* del Controlador, de forma que se llame siempre al método *pasar* usando el total de vidas con que se inicia el juego (atributo *total_vidas*) como segundo argumento.

5) Prueba el juego con la vista 2 y haz una captura de pantalla al conseguir salir del laberinto.

2. Es decir, una puerta solo desde esta habitación hacia la dirección donde intentamos movernos, quedando una pared si nos moviéramos en dirección contraria.


Modelo_laberinto
Habitacion
-total_intentos_fallidos
+pasar(direccion : Direccion, total_vidas : Integer)
habitacion_usuario
Controlador_laberinto
Controlador
-total_vidas : Integer = 0
+entrar(vidas : Integer) : void
+intentar_avanzar() : void