Pràctica 2 : Zombis

Volem crear un nou videojoc de zombis i volem reutilitzar unes classes que ja tenim disponibles d'altres vídeojocs:

- ZoombiCaminant representa un zombi que es desplaça. Aquesta classe no es pot modificar.
- ZoombiAtacant representa un zombi que ataca. Aquesta classe no es pot modificar.
- ZoombiAmagat representa un zombi que està amagat. Aquesta classe no es pot modificar.
- Personatge representa el nostre jugador.
- JocZoombis és el bucle de joc.
- FitxerProperties llegeix del fitxer que trobaràs dins del directori Files. Aquesta classe no es pot modificar.

Com podeu veure al diagrama UML les tres classes de zombis són similars, donat que no es poden modificar cal fer una adaptació per tal de poder utilitzar les tres classes com si fossin una mateixa.

Capes arquitectòniques

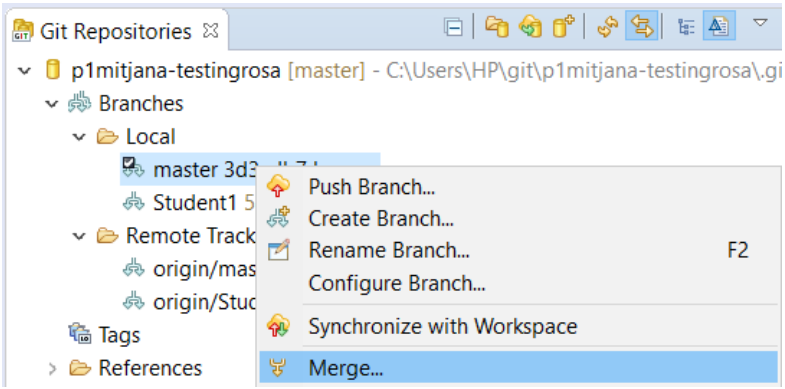
Al codi podràs veure que no hi ha cap Package creat, cal crear els packages necessaris per tal que separar les capes arquitectòniques i que no hi hagi una connexió directe entre la capa presentació i la capa persistència.

Ús de banques al repositori git

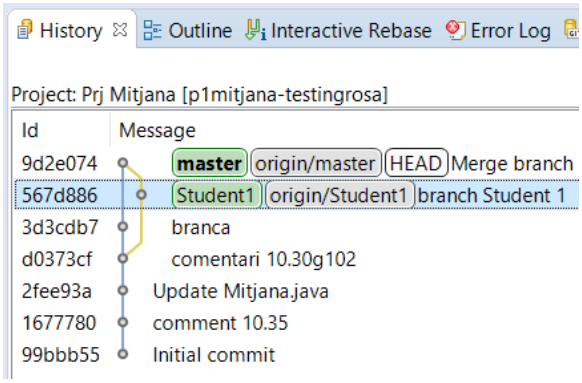
Per aquesta pràctica cal que cada estudiant creí una branca amb el seu nom i implementi una de les tres classe AdaptadorZoombi del Patró Adaptador, al finalitzar uneix la teva branca amb la branca per defecte.

Passos a seguir:

1. Abans implementeu la interfície IZoombi a la branca per defecte i després decidiu qui farà cada una de les tres classe AdaptadorZoombi del Patró Adaptador explicat a la següent secció.
2. Cada estudiant crea una branca on treballar:
   a. Botó dret sobre el projecte a Eclipse
   b. Team > Switch to > New Branch.
   c. Crea i implementa la classe acordada.
   d. Com a mínim realitza un Commit and Push.
   e. Feu un merge per unir la vostra branca amb la branca per defecte: cal donar amb el botó dret sobre la branca per defecte, i seleccionar Merge. Després selecciona la branca que vulguis unir.



   f. Tingueu en compte que potser caldrà resoldre conflictes, es convenient fer un Commit amb Push de la branca per defecte.
   g. A la següent imatge, podeu veure un exemple de les branques: Team > Show in History



Patró Adaptador

Com podeu veure al diagrama UML els mètode de les tres classes són similars però cal fer una adaptació. Ens demanen aplicar el patró adaptador de tal manera que es puguin manipular els tres tipus de zombis de la mateixa manera. Els elements del patró adaptador són els següents:

Objectiu: la nova interfície IZoombi que especifica el domini que usa el Client.

Adaptable: les classes existents que no volem modificar: ZoombiCaminant, ZoombiAtacant i ZoombiAmagat.

Adaptador: les noves classes AdaptadorZoombiCaminant, AdaptadorZoombiAtacant i AdaptadorZoombiAmagat que ens adapta les classes existents amb la especificació del Client.

Passos a seguir:

1. Cal crear una interfície IZoombi per definir les capçaleres de tots els mètodes necessaris.
2. Cada estudiant crea una de les tres noves classes adaptadores a una branca amb el seu nom (veure com fer-ho amb Eclipse a la pàgina 2). Entre tots els alumnes cal crear totes tres classes AdaptadorZoombiCaminant, AdaptadorZoombiAtacant i AdaptadorZoombiAmagat, que segueixen les següents indicacions:
    - Implementen la nova interfície IZoombi.
    - Defineix un atribut zombi de la classe que vol adaptar.
    - El constructor comprova que l'array d'entrada és de la mida desitjada (en cas contrari llança una excepció IllegalArgument) i inicialitza l'atribut zombi.
    - Els mètodes que ja existeixen a l'atribut zombi, simplement els invoquem.
3. Per últim cal crear un el testcase TestPatroAdaptador per tal de realitzar la comprovació de tots els possibles mètodes de les tres noves classes utilitzant una instància de la classe Personatge.

Patró Factoria

L'objectiu del patró fàbrica és retornar una instancia d'una classe en particular per mitja d'un identificador. Per fer-ho crearem una enumeració anomenada TipusZoombi que té definit com a constants els tres tipus de zoombi.

Els elements del patró factoria són els següents, pot visualitzar el diagrama de classes UML a l'última pàgina:

Producte: la interfície IZoombi

Producte Concret: les classes adaptador

Creador+Concret: la nova classe FactoriaZoombi

Usuari: la nova classe testFactoria

Passos a seguir:

1. Crea una enumeració TipusZoombi amb 3 valors de identifiquin els tres zombi. No cal crear cap constructor.
2. Crear una classe anomenada FactoriaZoombi per poder crear un dels tres tipus de motor. Aquesta classe és el creador+concret del patró factoria. Aquesta nova factoria tindrà un mètode amb les següents especificacions:
   - Nom del mètode: getZoombi
   - El primer paràmetre d'entrada és del tipus de la enumeració, que determina quina classe es crearà.
   - El segon paràmetre d'entrada és una array del tipus int que contindrà els paràmetres necessaris per instanciar la classe.
   - El paràmetre de sortida és del tipus de la interfície IZoombi perquè el mètode retorna un objecte d'una de les tres classes.

   Utilitza programació reflexiva per crear un nou objecte de la classe desitjada. Cal indicar al Class.forName el nom de la enumeració concatenat el string corresponent per indicar la seva ubicació exacta.

3. Acabar d'implementar la classe *testPatroFactoria* amb 3 testos, cadascú ha d'instanciar un personatge diferent, per comprovar si tot es correcte pots fer un assertTrue de les següents condicions:

- El zombi que s'ha creat és de la classe corresponent.
- Els atributs de les instàncies creades corresponen als paràmetres que has assignat.

4. En cas d'excepció, qui gestiona l'excepció?
5. Modifica la classe JocZoombis per tal que utilitzi les classes creades amb tots dos patrons.

Patró Singleton

L'objectiu del patró singleton és que la classe FactoriaZoombi sigui una única instància, i evitar la generació d'objectes innecessaris. Pots utilitzar qualsevol de les dues implementacions mostrades a classe: Lazy o Eager.