**MODELE**

Modele :

* Récupérer le personnage qui l’attaque
* Attaquer aléatoirement
* Recevoir des dégats
* Afficher ses stats
* Classe Abstraite

Design pattern Oberser : à implémenter

Observable.java

Package observer

public interface Observable {

public void addObserver(Observer obs);

public void removeObserver();

public void notifyObserver(String str);

}

Observer.java

package observer;

public interface Observer {

public void update(String str);

}

AbstractModel.java

package model;

import java.util.ArrayList;

import com.sdz.observer.Observable;

import com.sdz.observer.Observer;

public abstract class AbstractModel implements Observable{

protected double result = 0;

protected String operateur = "", operande = "";

private ArrayList<Observer> listObserver = new ArrayList<Observer>();

//Efface

public abstract void reset();

//Effectue le calcul

public abstract void calcul();

//Affichage forcé du résultat

public abstract void getResultat();

//Définit l'opérateur de l'opération

public abstract void setOperateur(String operateur);

//Définit le nombre à utiliser pour l'opération

public abstract void setNombre(String nbre) ;

//Implémentation du pattern observer

public void addObserver(Observer obs) {

this.listObserver.add(obs);

}

public void notifyObserver(String str) {

if(str.matches("^0[0-9]+"))

str = str.substring(1, str.length());

for(Observer obs : listObserver)

obs.update(str);

}

public void removeObserver() {

listObserver = new ArrayList<Observer>();

}

}

Calculator.java

package model;

import observer.Observable;

public class Calculator extends AbstractModel{

//Définit l'opérateur

public void setOperateur(String ope){

//On lance le calcul

calcul();

//On stocke l'opérateur

this.operateur = ope;

//Si l'opérateur n'est pas =

if(!ope.equals("=")){

//On réinitialise l'opérande

this.operande = "";

}

}

//Définit le nombre

public void setNombre(String result){

//On concatène le nombre

this.operande += result;

//On met à jour

notifyObserver(this.operande);

}

//Force le calcul

public void getResultat() {

calcul();

}

//Réinitialise tout

public void reset(){

this.result = 0;

this.operande = "0";

this.operateur = "";

//Mise à jour !

notifyObserver(String.valueOf(this.result));

}

//Calcul

public void calcul(){

//S'il n'y a pas d'opérateur, le résultat est le nombre saisi

if(this.operateur.equals("")){

this.result = Double.parseDouble(this.operande);

}

else{

//Si l'opérande n'est pas vide, on calcule avec l'opérateur de calcul

if(!this.operande.equals("")){

if(this.operateur.equals("+"))

this.result += Double.parseDouble(this.operande);

if(this.operateur.equals("-"))

this.result -= Double.parseDouble(this.operande);

if(this.operateur.equals("\*"))

this.result \*= Double.parseDouble(this.operande);

if(this.operateur.equals("/")){

try{

this.result /= Double.parseDouble(this.operande);

}catch(ArithmeticException e){

this.result = 0;

}

}

}

}

this.operande = "";

//On lance aussi la mise à jour !

notifyObserver(String.valueOf(this.result));

}

}

**CONSTRUCTEUR**

Construteur :

* Agir lors d’un clic sur attaquer le monstre
* Agir lors d’un clic sur se protéger contre un monstre
* Agir lors de la capture d’un monstre apres sa mort
* Agir sur la récupération des items d’un mosntre qu’on vient de tuer
* Agir sur l’équipement d’un équipement trouver

AbstractController.java -> super classe de contrôle

**VUE**

<https://www.supinfo.com/articles/single/3-exemple-programme-packaging-fichiers-java>

public class MainFrame extends *JFrame* {

private static final long serialVersionUID = 1L //(Facultatif ici, utilisé pour la serialisation)

public MainFrame()

{

super("File Packaging");

this.setSize(new Dimension(800,150)); //Dimensions fenêtre

this.setLocationRelativeTo(null); //Centrage à l'ecran

this.setVisible(true); //On affiche la fenêtre

this.setDefaultCloseOperation(EXIT\_ON\_CLOSE); //On définit la fermeture du programme sur la croix rouge

}

public static void main(String [] args)

{

new MainFrame();

}

}

Vous pouvez résoudre ce problème en implémentant le contrôleur en tant que machine à états à quatre états:

* sélectionner le personnage
* sélectionnez l'action
* sélectionner la cible
* attendre la fin de l'animation