**ANNEXE 2 - FORMULAIRES DISPONIBLES**



**Scénario**

**Titre** :

**Résumé** : . **Acteur** : Utilisateur (acteur principal)

**Pré-condition** :

**Post-condition** :

**Date de création** : **Date de mise à jour** : **Version** : .

**Créateur** :

|  |  |
| --- | --- |
| **Utilisateur** | **Système** |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *Element graphique correspondant* | boutonFeuille **OU** boutonPapier **OU** boutonCiseaux | boutonNouvellePartie | item de menu “itemAuSujetDe” | item de menu “itemJoueur” | item de menu “itemQuitter” |
| *Evénement* 🡪    *nomEtat* | Choix d’un signe | Clic sur nouvelle partie | clic sur a propos | clic sur Paramètre | clic sur Qutter |
| Etat initial | **Partie en cours** | ---- | ---- | ---- | ---- |
| Partie en cours | ---- | **Etat initial** | ---- | ---- | ---- |

**Description de l’interface de l’application – état par état**

*Copie d’écran de l’interface*

*Ici une première copie d’écran de l’interface, avec en légende les noms des éléments graphiques qui la composent*

*Ici une autre copie d’écran de l’interface, avec les sizers, dessinés et légendés*

*Etats des éléments graphiques lorsque le système est dans l’état* ***XX***

|  |  |
| --- | --- |
| **Elément d’interface** | **Description**  (visible / invisible ; activé / inactif ; focus ; …) |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

*Etats des éléments graphiques lorsque le système est dans l’état* ***XX***

|  |  |
| --- | --- |
| **Elément d’interface** | **Description**  (visible / invisible ; activé / inactif ; focus ; …) |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**Organisation des spécifications des classes – inspirée de MVC**

**class Additionneur**

**{**

public:

**///\* méthodes métier**

Additionneur();

virtual ~Additionneur();

int getOpG();

int getOpD();

void setOpG(int p\_op);

void setOpD(int p\_op);

int getSomme();

// retourne le résultat de l'opération opG+opD

protected:

**///\* attributs métier**

int opG; // opérande Gauche de l'opération opG + opD

int opD; // opérande Droit de l'opération opG + opD

// pour simplifier la classe, j’ai décidé de ne pas stocker la somme





public :

**///\* états du système**

**enum UnEtat {saisiesEnCours, erreur};**

**///\* Méthodes du Contrôleur**

// Pour gérer la logique de fonctionnement du système

**UnEtat getEtat();**

**void setEtat(UnEtat p\_etat);**

// Pour gérer les événements = actions de l’Utilisateur sur la Vue

**void demandeAddition(string p\_opG, string p\_opD);**

/\* Traite la demande d'addition des contenus des paramètres p\_opG et

p\_opD de type string \*/

**void accepteNotification();**

// Traite l’acceptation, par l’utilisateur, du message d’erreur

**void demandeEffacement();**

// Traite la demande d'effacement faite par l'utilisateur

// Pour gérer le lien entre le Modèle et la Vue

**Principale\* getVue();** // retourne pointeur sur vue

**void setVue(Principale\* p\_vue);**

// Permet à la vue de s'enregistrer auprès du modèle

protected:

**///\* Attributs liés au Contrôleur**

**Principale\* laVue;** // mémorise la Vue pour s’adresser à elle

**UnEtat etat;** // mémorise l'état du système

**};**

**class Principale : public wxFrame**

**{**

**///\* Méthodes et attributs 'classiques' de la Vue**

public:

// Le constructeur

**Principale(const wxString& title);**

// Le destructeur

**virtual ~Principale();**



private:

**///\* Gestionnaires d'événements**

**void OnClicBtonAddition (wxCommandEvent& event);**

/\* Maintenant, **notifie** le Contrôleur que l'utilisateur a demandé le  
 calcul \*/

**void OnClicBtonEffacer (wxCommandEvent& event);**

/\* Maintenant, **notifie** le Contrôleur que l'utilisateur a demandé

l'effacement \*/

**///\* Attributs membres**

// pas de changement par rapport à la version antérieure





**///\* Méthodes de mise à jour de la Vue**

public:

**void effacerZonesSaisie();**

// efface le contenu des 2 champs de saisie

**void afficherResultat(int p\_res);**

// afficher l'entier dans la zone label\_res

**void afficherEtatErreur();**

// affiche la vue de l'application lorsqu'elle est en état d'erreur

**///\* Méthodes et attributs pour faire le lien avec le Contrôleur**

public:

**Additionneur\* getAdditionneur();**

**void setAdditionneur (Additionneur\* p\_add);**

private:

**Additionneur\* additionneur;**

**};**