Compte rendu d'activité

TP : Prise en Charge Incident d'une Application Console C#

10/2020

Table des matières

Contexte

Une cliente réalise des applications d'apprentissage ainsi que de tests de connaissances dans divers environnement. Actuellement, une application est réalisée dans un environnement non graphique, qui permet de contrôler les connaissances en calculs mathématiques basiques : addition et multiplication. Suite à des tests réalisés à la demande du chef de projet, un ticket d'incidents a été généré par des erreurs détectées qu'il est nécessaire de corriger, puis de notifier les modifications effectués dans le rapport d'incidents.

Application console C#

Code existant

Ce fichier **Program.cs** rassemble l'ensemble de l'algorithme de l'application. Son code source initial est accessible ici

Les fonctionnalités existantes

Actuellement, à l'ouverture de l'application, un menu s'affiche :

- Offre le choix de contrôler ses connaissances en addition ou en multiplication.
- Suivant le choix, une opération est proposée et il faut saisir la réponse.
  - Si la réponse est correcte, "Bravo !" est affiché.
  - Dans le cas contraire, "Faux" est affiché, suivi de la réponse correcte.
- Dans tous les cas, le menu est à nouveau affiché, permettant de faire un nouveau test

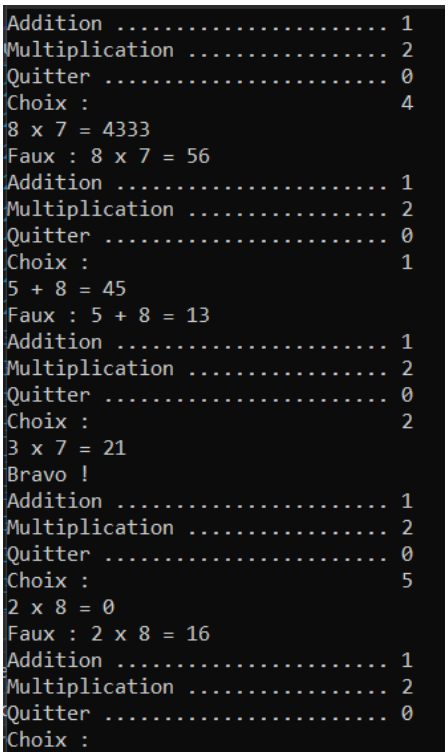

Figure 1: Exemple d'affichage de l'application à l'état initial

Expressions des besoins

Déterminer l'origine des dysfonctionnements

- **Incident 1:** Sur le menu affiché, la saisie d'un nombre différent que ceux proposés déclenche la fonctionnalité d'Addition.
- **Incident 2:** Sur le menu affiché, la saisie d'un caractère engendre un arrêt de l'application.

- **Incident 3:** Dans les fonctionnalités de test d'addition et de multiplication, la saisie d'un caractère engendre un arrêt de l'application.

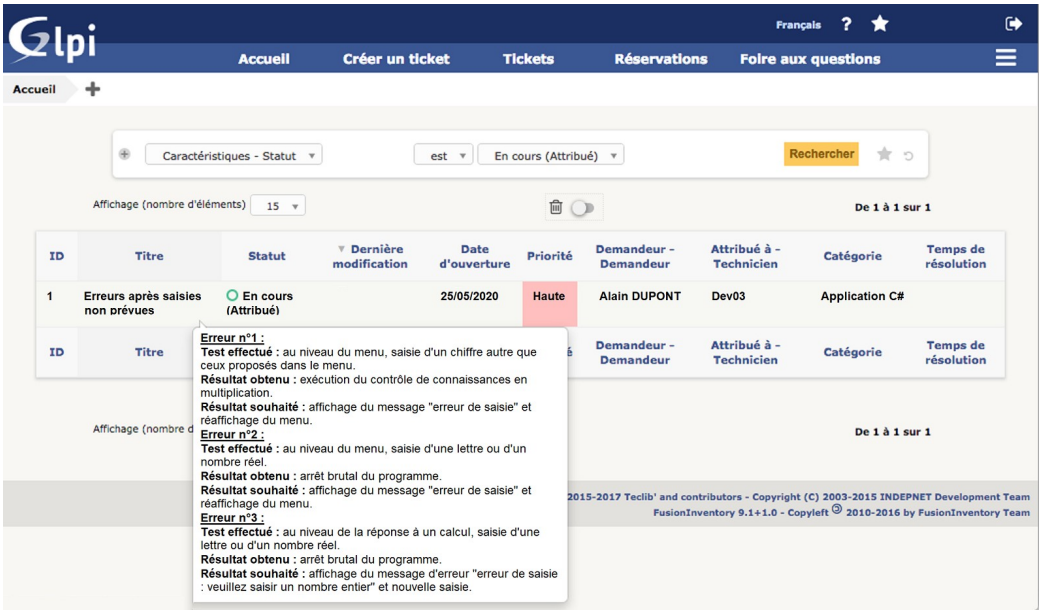

Figure 2: Ticket Incidents GPLI généré par l'application

Restituer les modifications nécessaires

Un rapport d'incident type ayant été fourni, il conviendra de le remplir avec les modifications apportées à l'application corrigeant les bugs détectés. Ces documents sont trouvables ici

Informations techniques et outils utilisés

Spécificités IDE

L'évolution de l'application récupérée a été effectué avec la version 16.11.10 de l'IDE Microsoft Visual Studio 2019.

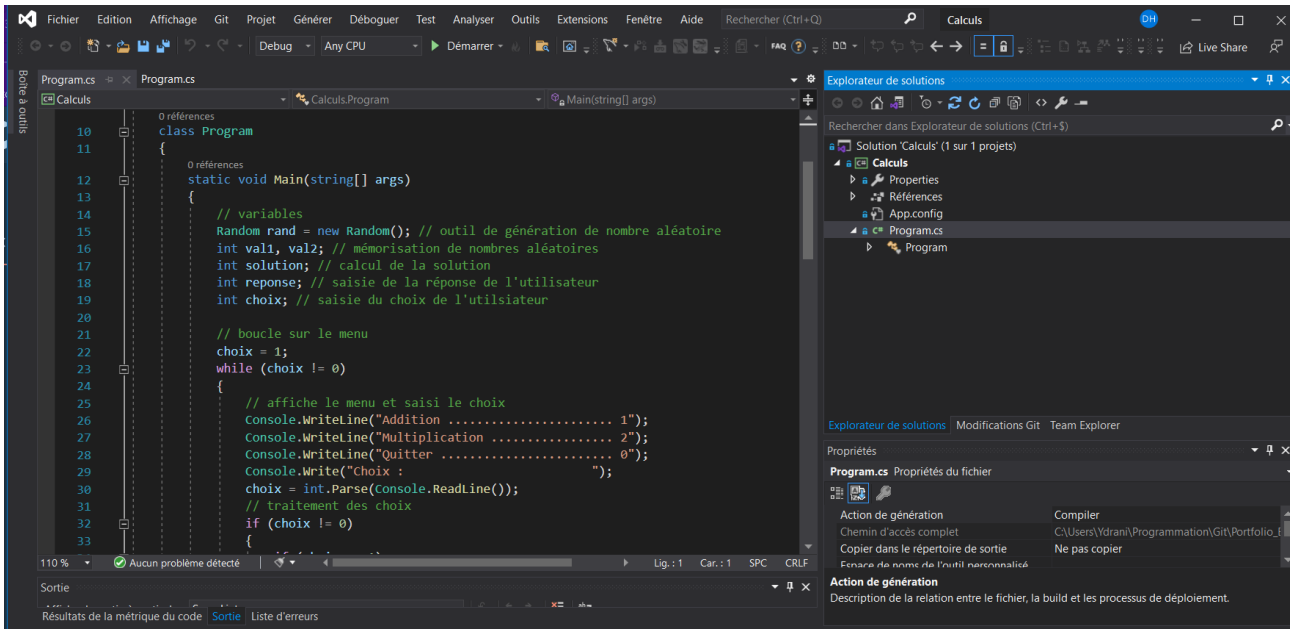

Figure 3: Affichage du fichier Program.cs dans Visual Studio

Spécificités outils et plugins

La revue du code produite a été effectuée par l'intermédiaire des outils de débogage intégrer par l'IDE : point d'arrêt, exécution pas à pas et suivi de variables par espions. Ces opération peuvent être effectués depuis l'onglet Débogage, puis on définissant des points d'arrêts dans la marge de gauche.

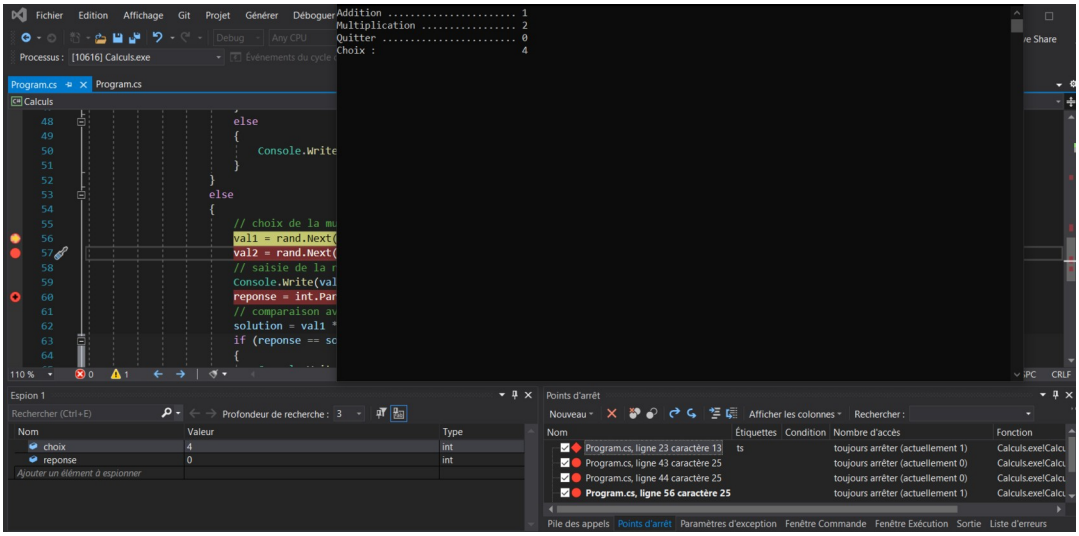

Figure 4: Exemple de débogage avec les outils intégrés de Visual Studio

Réalisation

Mise en place de l'environnement de développement

IDE Visual Studio

Visual Studio est un IDE permettant le développement d'application C#. L'IDE est téléchargeable à ce lien, cependant la version pourrait changer selon les évolutions de l'IDE. Après l'installation de cet IDE avec les options par défaut, il convient de le démarrer en modeadministrateur afin de simplifier les différents éléments de démarrage. De plus, ici le logiciel a été configuré pour s'ouvrir automatiquement dans ce mode. Pour cela, il convient d'effectuer un clic droit sur l'icone de lancement de Visual Studio, d'aller dans « Propriétés», puis à l'ouverture de la fenêtre des propriétés, de cliquer sur « Avancé » puis de cocher « exécuter en tant qu'administrateur comme le montre la *figure 15.*

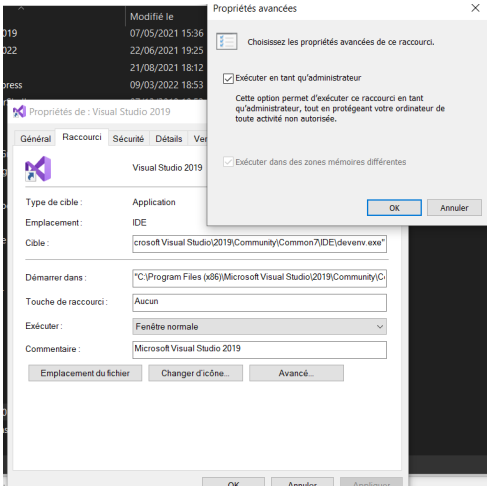

Figure 5: Fenêtre des propriétés de l'icône de lancement de l'IDE

Récupération du code source

La récupération du code source ce fait sur la plateforme CNED sous forme de dossier zipper. Après téléchargement,avoir dézipper le dossier et avoir placer le projet dans le dossier de traitement**,** il est possible d'ouvrir directement le projet dans l'IDE en cliquent sur le fichier finissant par **.sln, ici Calcul.sln.** Enfin, il suffit de cliquer sur l'icône de lecture verte à côté de démarrer afin de lancer l'application.

Lier le dépôt Github avec Visual Studio

Afin de pouvoir directement gérer le dépôt GitHub depuis Visual Studio, il convient d'aller dans l'onglet «**Modification Git** » puis de saisir un message de commit dans la fenêtre qui apparaît. A cet instant, les modifications non intégrés au dépôt s'affiche dans l'onglet **Modification**. Dans le cas présent, le dépôt étant préalablement créé et le compte de dépôt étant déjà lié à Visual Studio, il suffit de commiter puis de cliquer sur « **valider tout et pousser** ». A la réussite de l'envoi, une pop-up s'affiche.

Incident 1 : Modification du menu

Actuellement, le choix de saisie du menu est effectué par une condition if qui vérifie si la valeur de la variable choix est différente de 0. La fonctionnalité d'addition est exécutée dans le cas où la saisie du choix est de 1.

```
// boucle sur le menu
choix = 1;
while (choix != 0)
{
    // affiche le menu et saisi le choix
    Console.WriteLine("Addition ..... 1");
    Console.WriteLine("Multiplication ..... 2");
    Console.WriteLine("Quitter ..... 0");
    Console.Write("Choix :                    ");
    choix = int.Parse(Console.ReadLine());
    // traitement des choix
    if (choix != 0)
    {
        if (choix == 1)
        {
```

Pour le choix de la multiplication, la condition du if suivant ce block est sur else, ainsi tout choix différent de 1 ou 0 executera nécessairement la fonctionnalité de multiplication . Afin de remédier à cela, une fonction de **switch** remplacera cette succession de if. Ainsi, une valeur 1 de choix engendrera l'exécution du code Addition, le 2 pour la Multiplication et 0 pour quitter l'application.

```
while (choix != "0")
{
    // affiche le menu et saisi le choix
    Console.WriteLine("Addition ..... 1");
    Console.WriteLine("Multiplication ..... 2");
    Console.WriteLine("Quitter ..... 0");
    Console.Write("Choix :                    ");
    choix = Console.ReadLine();
    // traitement des choix
    switch (choix)
    {
        case "1": // addition
            val1 = rand.Next(1, 10);
            val2 = rand.Next(1, 10);
            // saisie de la réponse
            correct = false;
            while (!correct)
            {
```

```
            Console.WriteLine("Faux : " + val1 + " x " +
        }
        break;
    case "0": // demande de fin de programme
        break;
    default: // autre valeur donc erreur de saisie
        Console.WriteLine("Erreur de saisie");
        break;
```

Incident 2 : Arrêt par saisie de caractère sur le menu

Cet incident a été géré en plaçant comme choix par défaut du switch toute autres valeurs que 1, 2 et 0. Ainsi les caractères peuvent être gérés. La figure ci-dessus illustre cet ajout. Aussi, le type de choix a été remplacé par le type String afin d'éviter d'autre erreur de ce genre.

```
// variables
Random rand = new Random(); // outil de génération de nombr
int val1, val2; // mémorisation de nombres aléatoires
int solution; // calcul de la solution
int reponse = 0; // saisie de la réponse de l'utilisateur
string choix; // saisie du choix de l'utilsiateur
bool correct; // mémorise si la saisie est un entier

// boucle sur le menu
choix = "1";
while (choix != "0")
```

Figure 6: Modifications de type de la varaible choxi

Incident 3 : Arrêt par saisie de caractère dans la réponse d'opération

Cet incident survient par le fait que l'application attend une valeur chiffrée afin de convertir cet input en int. Or cet conversion est impossible lorsque la valeur à convertir n'est pas un nombre. Afin de gérer cela, il convient de mettre en place un bloc de try- catch qui tentera de convertir la saisie.

```
case "1": // addition
    val1 = rand.Next(1, 10);
    val2 = rand.Next(1, 10);
    // saisie de la réponse
    correct = false;
    while (!correct)
    {
        try
        {
            Console.Write(val1 + " + " + val2 + " = ");
            reponse = int.Parse(Console.ReadLine());
            correct = true;
        }
        catch
        {
            Console.WriteLine("Saisissez un entier");
        }
    }
```

Figure 7: Modifications du code par l'ajout d'un bloc try catch

Ainsi, en cas d'échec le catch récupére l'erreur et affiche un message d'erreur. A l'inverse, l'exécution s'opère normalement.

Bilan

Après modification du code dans **Program.cs**, l'affichage et l'algorithme mis en place permettent d'avoir l'affichage suivant lorsque l'on tente de réitérer les incidents.

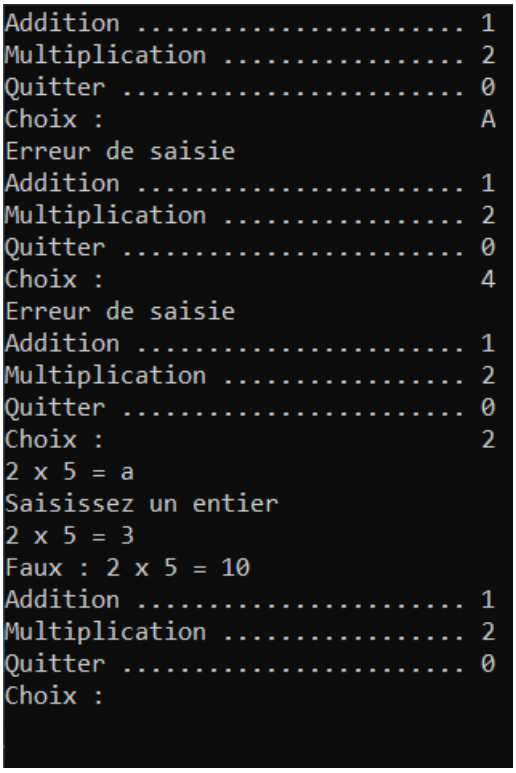

Figure 8: Affichage de l'application après gestion des différents incidents

N'ayant alors ni arrêt d'application à la saisie de caractère, ni d'autres bug du à des saisies non couvertes. Il est possible de statuer que le ticket est clôt.

Liste des compétences couvertes

B1-2 Répondre aux incidents et aux demandes d'assistance et d'évolution

- Traiter des demandes concernant les applications