

# EditCalc Version 2

TRUONG Jacky, MOITY Anthony, BOUSBAINE Rabah

Livrables

Lognes, mars 2020

**Centre GRETA MTE 77 Lognes**

**- Espace DIDEROT**

95-97 boulevard du Segrais 77185 Lognes

# Table des matières

Introduction.....	Page 3
Compréhension du besoin.....	Page 3
Diagramme de cas d'utilisation.....	Page 3
Diagramme de classes (généré).....	Page 4
Diagramme de séquence de conception.....	Page 5
Choix du thème.....	Page 6
Maquettes.....	Page 7
Organisation.....	Page 11
Ressources.....	Page 11
Planning.....	Page 12
Tableau de bord.....	Page 13
Présentation technique.....	Page 14
Création et disposition des éléments.....	Page 14
Programmation des éléments.....	Page 14
Test fonctionnel.....	Page 15
Descriptif du code.....	Page 16
Mémoires.....	Page 22
Mémoire sur l'intégration continue.....	Page 22
Mémoire sur les outils de déploiement.....	Page 25
Manuels.....	Page 35
Manuel d'installation.....	Page 35
Manuel utilisateur.....	Page 40
Bilan.....	Page 46
Conclusion.....	Page 47

# Introduction

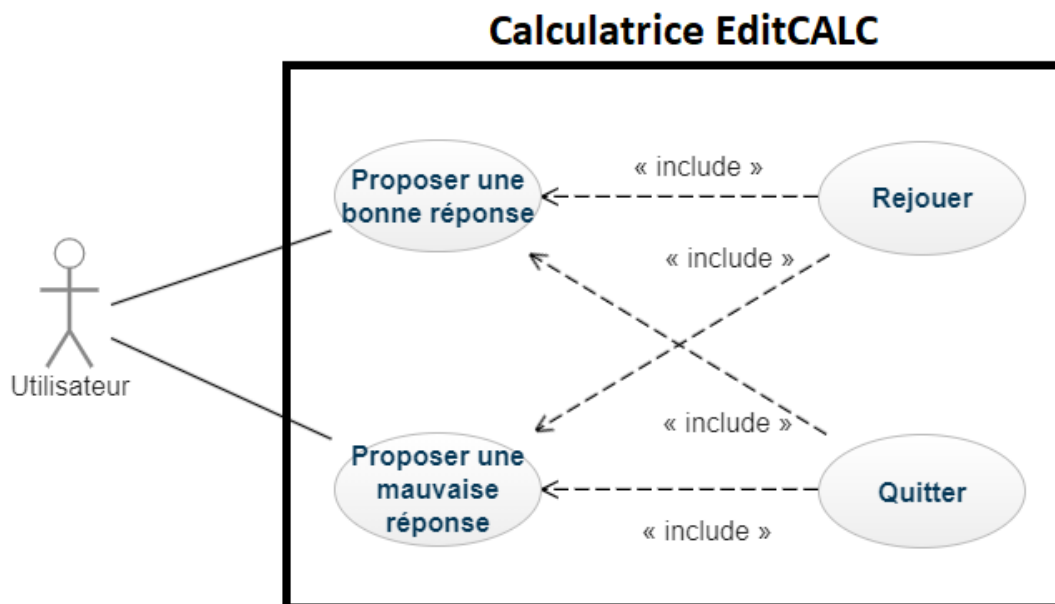
L'Editeur dénommé « EditCALC » propose des solutions éducatives pour les écoles, et souhaite proposer une application pour aider les élèves de « CP1 » à compter jusqu'à 10 à travers une interface Homme-Machine. Notre mission sera de créer cette application.

## Compréhension du besoin

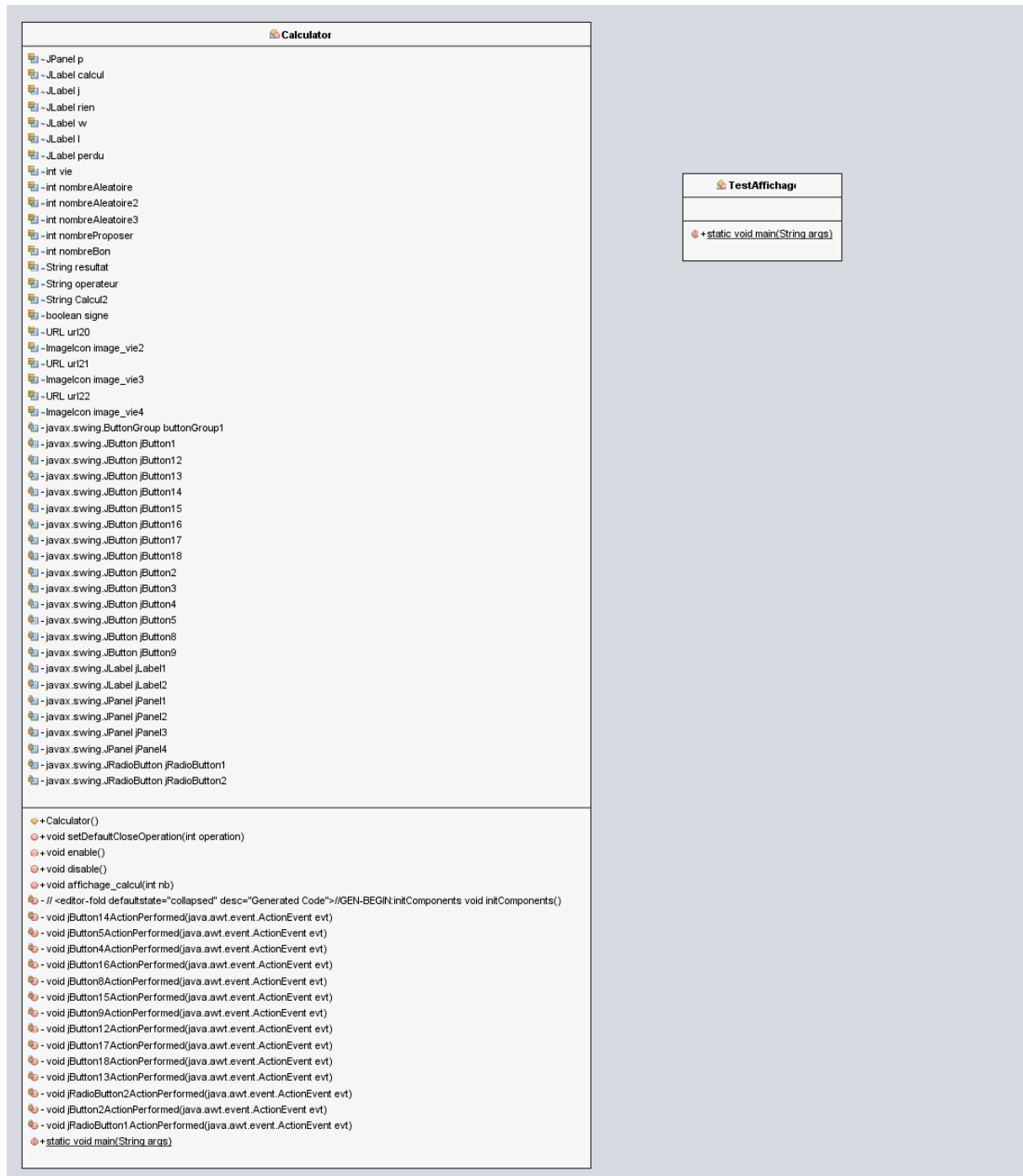
Après la proposition d'une première version, le client n'est pas satisfait de l'utilisation proposée de cette calculette bi fonction.

Le but du développement de cette deuxième version est de pouvoir laisser 3 chances à l'élève pour qu'il puisse trouver la réponse soi-même. Au-delà de 3 erreurs, la calculatrice affichera la réponse. Cette nouvelle fonctionnalité vient du concept que c'est en faisant des erreurs que nous apprenons mieux. Un système de félicitation qui suivra le thème de la calculatrice, ainsi que l'affichage d'une publicité en sortie d'application seront également intégrés dans le but de rendre l'application plus attrayante pour les élèves. Enfin, l'élève aura le choix de fermer manuellement son application s'il juge qu'il s'est assez exercé.

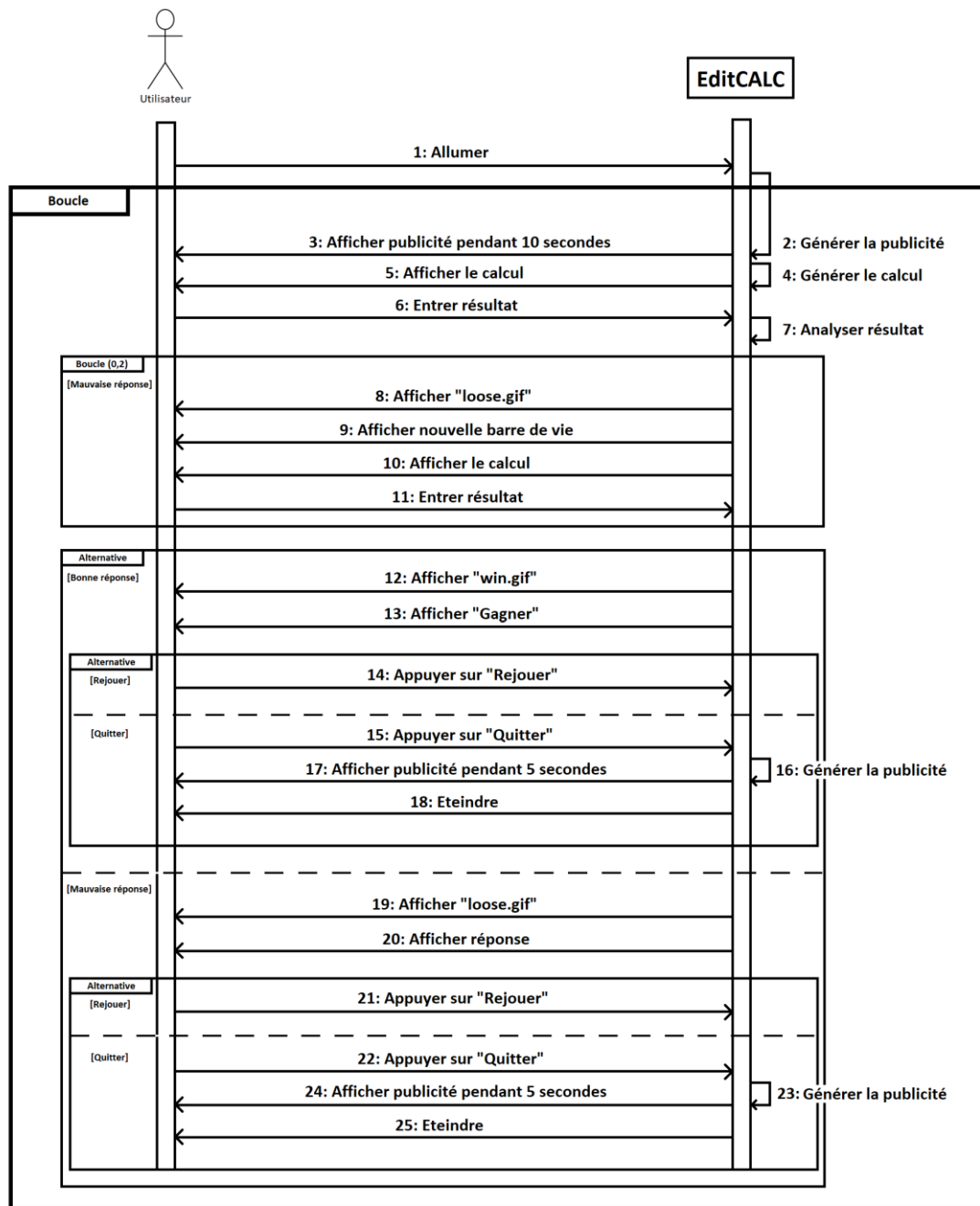
## Diagramme de cas d'utilisation



# Diagramme de classes (généré par easyUML)



# Diagramme de séquence de conception



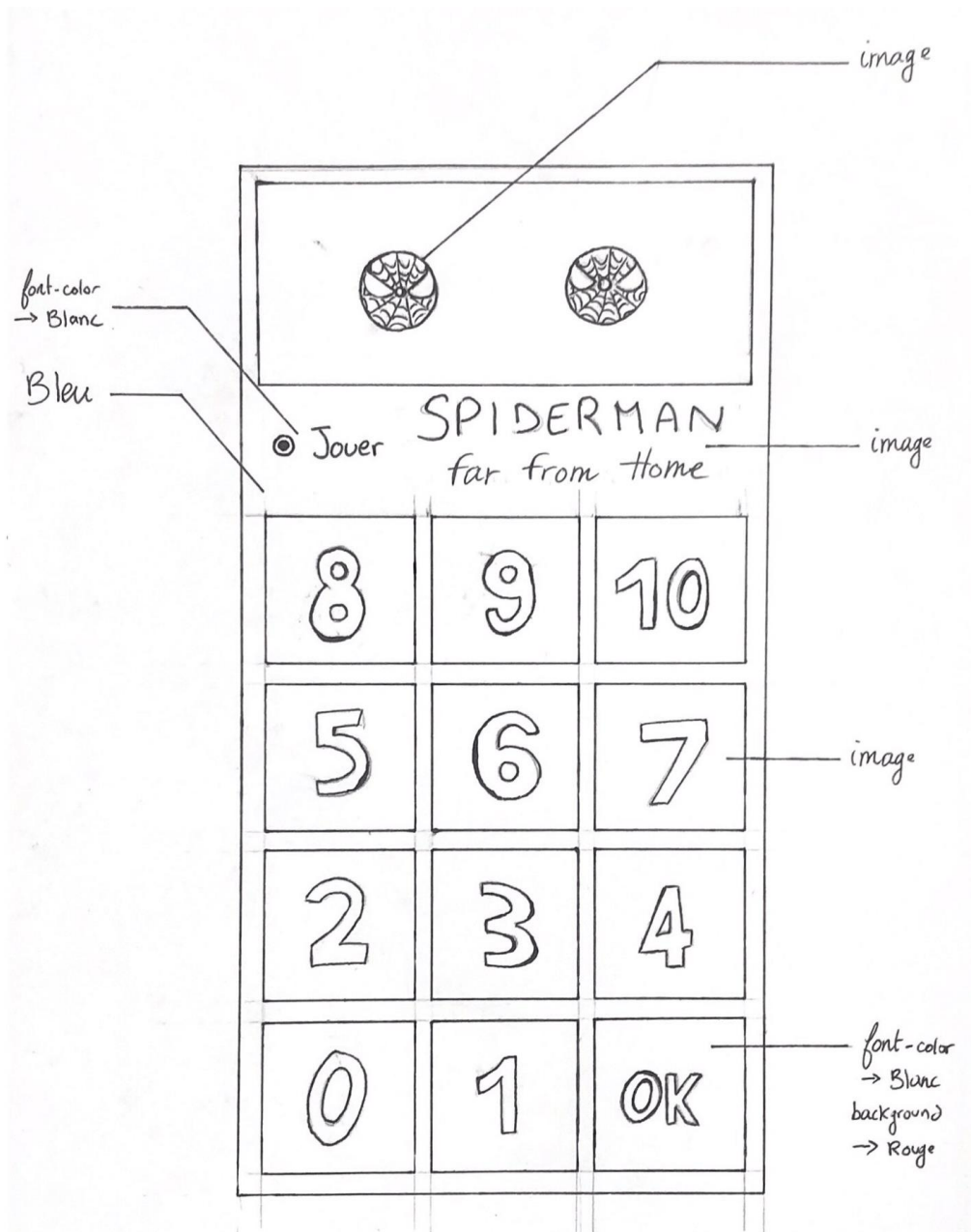
## Choix du thème

Nous avons opté pour un thème Spiderman, car selon nous, Spiderman est le héros préféré de tous les enfants. Le choix de ce thème permet de capter l'attention des enfants, ce qui pourrait les encourager à travailler plus à travers cette application.



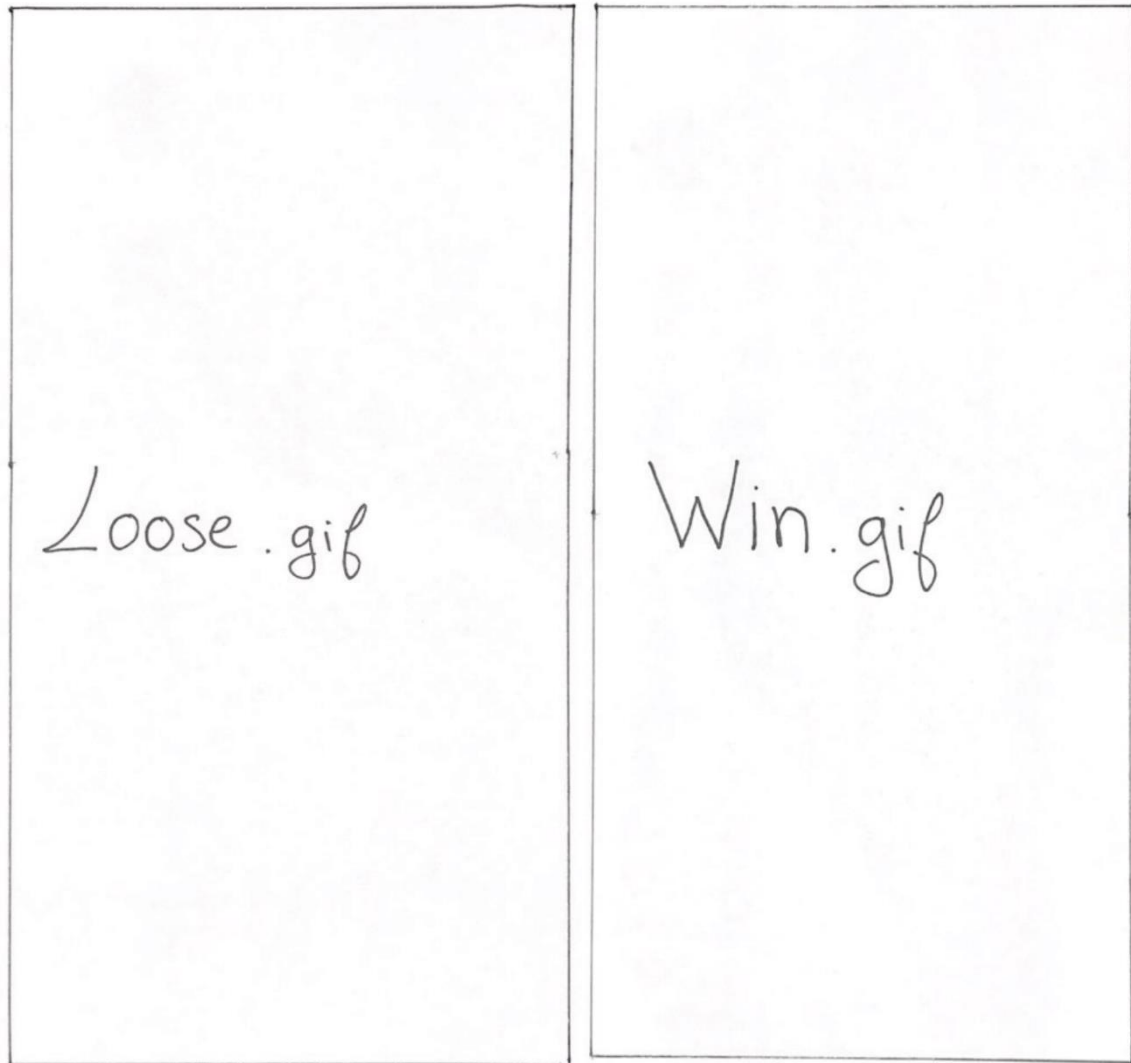
# Maquettes

Avant de développer la version 2, nous devons dans un premier temps concevoir des maquettes afin de valider et tester certains aspects de la calculatrice.



Maquette de la version 1

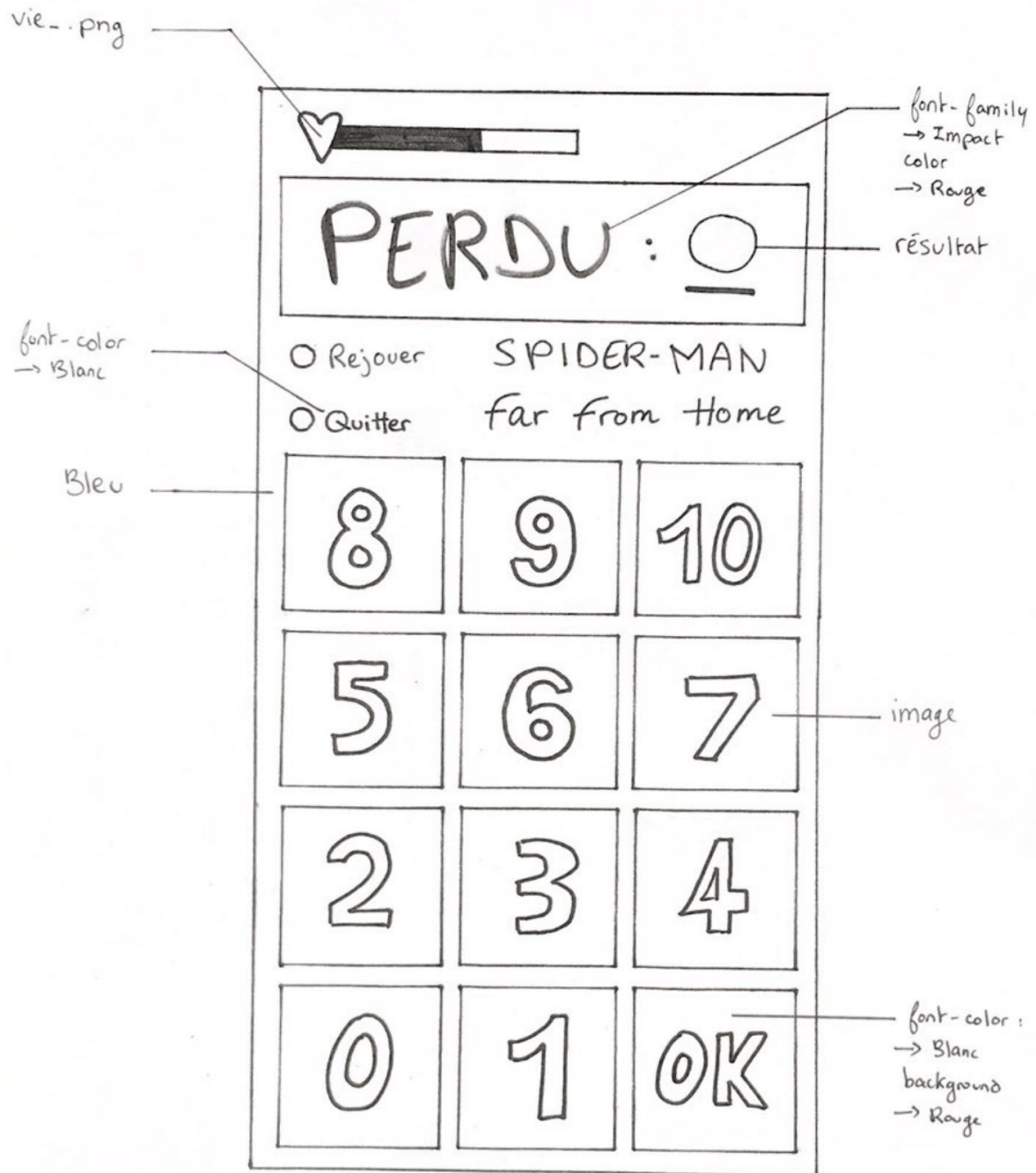
## Maquettes finales



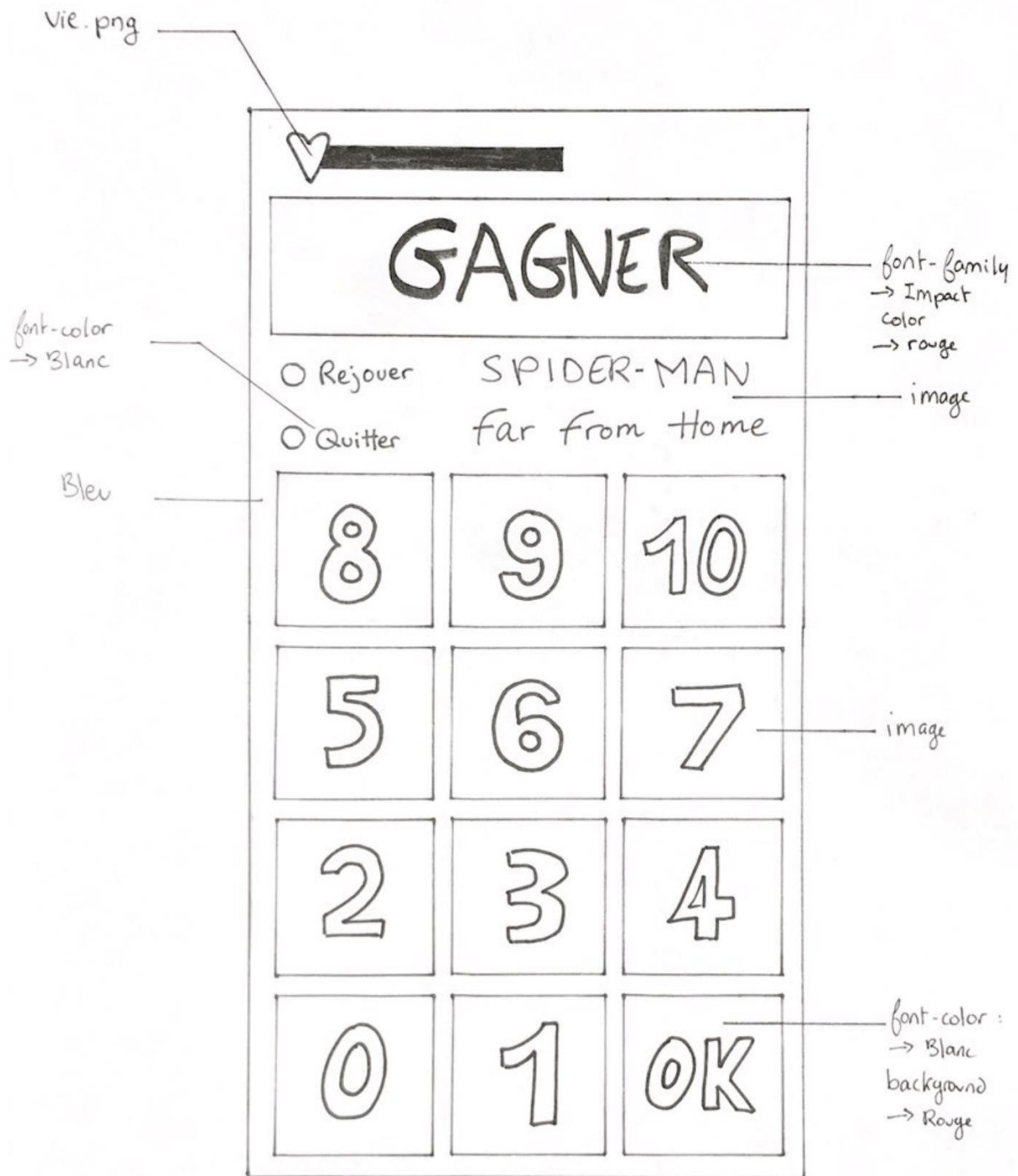
— S'affiche en cas de  
mauvaise réponse

— S'affiche en cas de  
bonne réponse





Maquette V2.1 (Perdu + réponse)



Maquette V2.2 (Gagner)

# Organisation

Avant de se lancer dans le développement de la version 2, nous devons savoir quelles ressources utilisées (ressources logicielles et humaines), et définir un planning.

## Ressources logicielles

- ❖ Netbeans IDE 8.2 + Swing : Nous avons tapé le code sur Netbeans puis pour l'interface graphique, nous avons utilisé les composants Swing
- ❖ Discord : Conçu initialement pour les « gamers », nous avons utilisé cette plateforme de communication pour s'envoyer des blocs de code, parler via un chat vocal, et travailler en partage d'écran
- ❖ Trello : C'est un logiciel de gestion de projet en ligne, fonctionnant avec des cartes et des colonnes. Nous l'avons utilisé pour nous assigner des tâches et voir l'avancé de chacun sans pour autant passer par Discord.

## Ressources humaines

- ❖ Jacky
- ❖ Anthony
- ❖ Rabah
- ❖ Les professeurs du GRETA
- ❖ Les contacts tiers

# Planning

## Planning pré-BTS Blanc


Du 28 février au 9 mars							
Jours	Lundi	Mardi	Mercredi	Jeudi	Vendredi	Samedi	Dimanche
Disponibilité	Indisponible	Disponible	Disponible	Indisponible	Indisponible	Indisponible	Disponible

## Planning durant le BTS Blanc

Du 9 mars au 15 mars							
Jours	Lundi	Mardi	Mercredi	Jeudi	Vendredi	Samedi	Dimanche
Disponibilité	Indisponible	Indisponible	Indisponible	Indisponible	Indisponible	Disponible	Disponible

## Planning post-BTS Blanc

Du 15 mars à aujourd'hui							
Jours	Lundi	Mardi	Mercredi	Jeudi	Vendredi	Samedi	Dimanche
Disponibilité	Disponible	Disponible	Disponible	Disponible	Disponible	Indisponible	Indisponible

	Disponible
	Indisponible



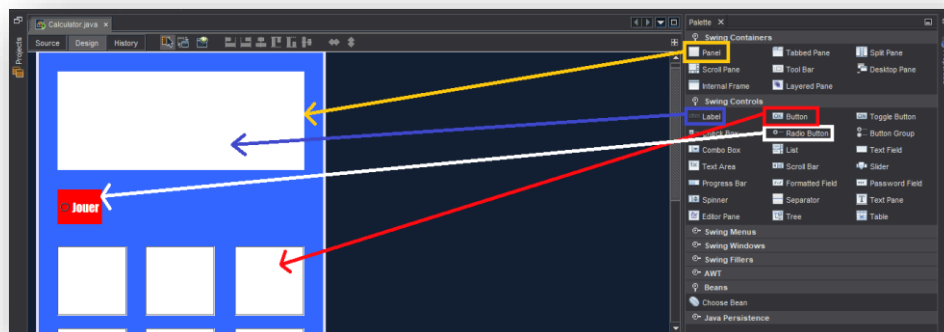
## Présentation technique

Dans la conception de cette version 2 de la calculatrice, nous avons utilisé les composants Swing, afin de créer une interface Homme-machine entre les utilisateurs et la calculatrice. Nous pourrions ainsi gérer le code source de la calculatrice ainsi que son design.

## La création et la disposition des éléments

Les composants Swing permettent de créer les éléments graphiques.

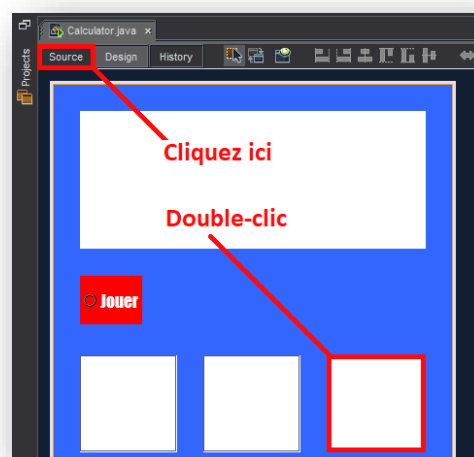
Les éléments sont disposés manuellement, et leurs apparences peuvent être modifiées depuis la palette Swing.



## La programmation des éléments

Disposer les éléments ne suffit pas, il faut les lier, leur attribuer des méthodes...etc

Pour cela, nous devons aller dans « Source » ou bien faire un double-clic sur l'élément qu'on veut programmer



## Descriptif des tests

TEST FONCTIONNEL V2			
ACTIONS	ATTENDU	PROBLÈME RENCONTRÉ	RESULTATS
Exécuter la calculatrice	Ouverture de la calculatrice	Problème trouvé en V1 mais pas en V2	OK
Appuyer sur le bouton « JOUER »	La calculatrice génère une publicité pendant 10 secondes et génère un calcul aléatoire	Problème trouvé en V1 mais pas en V2	OK
Appuyer sur un chiffre	Génère le nombre d'image demandé et appuie sur « OK » pour valider	Problème trouvé en V1 mais pas en V2	OK
Système de vie	L'utilisateur perd de la vie à chaque mauvaise réponse	La vie n'est pas bien affiché	OK
GIF : GAGNÉ et PERDU	Affichage de GIF lorsque l'utilisateur gagne ou perd	Problème au niveau des dimensions	OK
Résultat final (gagné)	L'utilisateur doit entrer la bonne réponse avant 3 choix. Cela affichera le GIF gagnant puis affichera « GAGNÉ »	Le mot « GAGNÉ » ne s'affiche pas	OK
Résultat final (perdu)	L'utilisateur doit entrer la bonne réponse avant 3 choix. Si ce n'est pas le cas, cela affichera le GIF perdant puis affichera « PERDU » en donnant la bonne réponse	La solution ne s'affichait pas  Le mot « PERDU » ne s'affiche pas	OK
Bouton rejoué et quitter	Après avoir connu son résultat (GAGNÉ ou PERDU), le système propose à l'utilisateur de rejouer ou de quitter	L'utilisateur ne pouvait pas rejouer	OK
La calculatrice se ferme	5 secondes après avoir quitté. La calculatrice régénère une pub puis se ferme	La pub ne s'affiche pas	OK

# Descriptif du code

```

JPanel p = new JPanel();
JLabel calcul = new JLabel();
JLabel j = new JLabel();
JLabel rien = new JLabel();
JLabel w = new JLabel();
JLabel l = new JLabel();
JLabel perdu = new JLabel();

int vie=4;
int nombreAleatoire;//on initialise une variable nombreAleatoire qui est un nombre aléatoire entre 0 et 9
int nombreAleatoire2;// on créer une nouvelle variable nombreAleatoire2
int nombreAleatoire3= (int) (Math.random() * (2 ));
int nombreProposer;
int nombreBon;

String resultat;
String operateur;
String Calcul3;

boolean signe;

URL uri20 = ClassLoader.getSystemClassLoader().getResource("images/vie2.png");
ImageIcon image_vie2 = new ImageIcon(new ImageIcon(uri20).getImage().getScaledInstance(225, 45, 225));

URL uri21 = ClassLoader.getSystemClassLoader().getResource("images/vie4.png");
ImageIcon image_vie3 = new ImageIcon(new ImageIcon(uri21).getImage().getScaledInstance(225, 45, 225));

URL uri22 = ClassLoader.getSystemClassLoader().getResource("images/vie3.png");
ImageIcon image_vie4 = new ImageIcon(new ImageIcon(uri22).getImage().getScaledInstance(225, 45, 225));

```

```

public Calculator() {
    initComponents();
    this.setDefaultCloseOperation(DO_NOTHING_ON_CLOSE);//pour ne plus pouvoir fermer la fenêtre

    URL url1 = ClassLoader.getSystemClassLoader().getResource("images/0.jpg");//on enregistre l'image qui est sur le bouton 0
    ImageIcon icon = new ImageIcon(new ImageIcon(url1).getImage().getScaledInstance(100, 100, 100));//on créer un icone et on met l'image dans l'icone et on défini la taille
    jButton12.setIcon(icon);//on ajoute l'icone au bouton

    URL url2 = ClassLoader.getSystemClassLoader().getResource("images/1.jpg");
    ImageIcon icon1 = new ImageIcon (new ImageIcon(url2).getImage().getScaledInstance(100, 100, 100));
    jButton9.setIcon(icon1);

    URL url3 = ClassLoader.getSystemClassLoader().getResource("images/2.jpg");
    ImageIcon icon2 = new ImageIcon(new ImageIcon(url3).getImage().getScaledInstance(100, 100, 100));
    jButton8.setIcon(icon2);

    URL url4 = ClassLoader.getSystemClassLoader().getResource("images/3.jpg");
    ImageIcon icon3 = new ImageIcon(new ImageIcon(url4).getImage().getScaledInstance(100, 100, 100));
    jButton14.setIcon(icon3);
}

```



```

URL url11 = ClassLoader.getSystemClassLoader().getResource("images/banniere.png");
ImageIcon iconLogo = new ImageIcon(new ImageIcon(url11).getImage().getScaledInstance(277, 97, 277));

jLabel1.setIcon(iconLogo);

URL url16 = ClassLoader.getSystemClassLoader().getResource("images/viel.png");
ImageIcon image_vie = new ImageIcon(new ImageIcon(url16).getImage().getScaledInstance(225, 45, 225));

jLabel2.setIcon(image_vie);

URL url1 = ClassLoader.getSystemClassLoader().getResource("images/logo_null.jpg");//chargement de l'image
ImageIcon zero = new ImageIcon(new ImageIcon(url1).getImage().getScaledInstance(40, 40, 40));//nouvelle icon, initialisation de l'image dans l'icon et définition de sa taille

rien.setIcon(zero);

URL url12 = ClassLoader.getSystemClassLoader().getResource("images/10.png");
ImageIcon icon10 = new ImageIcon(new ImageIcon(url12).getImage().getScaledInstance(100, 100, 100));

jButton3.setIcon(icon10);

URL url13 = ClassLoader.getSystemClassLoader().getResource("images/film.jpeg");
ImageIcon film = new ImageIcon(new ImageIcon(url13).getImage().getScaledInstance(412, 743, 412));

j.setIcon(film);

URL url28 = ClassLoader.getSystemClassLoader().getResource("images/win.gif");
ImageIcon win = new ImageIcon(new ImageIcon(url28).getImage().getScaledInstance(512, 743, 512));

w.setIcon(win);

URL url26 = ClassLoader.getSystemClassLoader().getResource("images/loose.gif");
ImageIcon loose = new ImageIcon(new ImageIcon(url26).getImage().getScaledInstance(512, 743, 512));

l.setIcon(loose);

```

```

jPanel3.setLayout(new GridLayout(1,1));//on initialise une grille de 1 sur 1 pour le JPanel qui affiche le resultat
jPanel3.setSize(358,93);

JLabel Hello = new JLabel(" ");//nouveau JLabel
Hello.setFont(new Font("Impact", Font.PLAIN, 72));//on initialise la police à 110

jPanel3.add(Hello);//ajout du JLabel dans le JPanel qui affiche le resultat

calcul.setSize(358,93);
calcul.setFont(new Font("Impact", Font.PLAIN, 72));
calcul.setForeground(java.awt.Color.red);
calcul.setHorizontalAlignment(JLabel.CENTER);

perdu.setSize(358,93);
perdu.setFont(new Font("Impact", Font.PLAIN, 72));
perdu.setForeground(java.awt.Color.red);
perdu.setHorizontalAlignment(JLabel.CENTER);

j.setSize(412,743);

p.setSize(412,743);

w.setSize(412,713);

rien.setSize(358,93);
rien.setFont(new Font("Impact", Font.PLAIN, 72));
rien.setForeground(java.awt.Color.red);
rien.setHorizontalAlignment(JLabel.CENTER);

l.setSize(412,743);

jLabel2.setSize(225,45);
}

```

```

jRadioButton2.setEnabled(true);

jButton4.setEnabled(true);
jButton5.setEnabled(true);
jButton6.setEnabled(true);
jButton9.setEnabled(true);
jButton12.setEnabled(true);
jButton13.setEnabled(true);
jButton14.setEnabled(true);
jButton15.setEnabled(true);
jButton16.setEnabled(true);
jButton17.setEnabled(true);
jButton18.setEnabled(true);
jButton2.setEnabled(true);

}

@Override
public void disable() { //Fonction pour bloquer tout les boutons
    jPanel13.setEnabled(false);

    jRadioButton2.setEnabled(false);

    jButton4.setEnabled(false);
    jButton5.setEnabled(false);
    jButton6.setEnabled(false);
    jButton9.setEnabled(false);
    jButton12.setEnabled(false);
    jButton13.setEnabled(false);
    jButton14.setEnabled(false);
    jButton15.setEnabled(false);
    jButton16.setEnabled(false);
    jButton17.setEnabled(false);
    jButton18.setEnabled(false);
    jButton2.setEnabled(false);

```

```

}

public void affichage_calcul(int nb)
{ //Fonction pour afficher le résultat avec des images

    for(int i=1;i<=nb;i++) //Boucle selon nb = nombre sur lequel on a cliqué
    {
        URL url = ClassLoader.getSystemClassLoader().getResource("images/spiderman.png");//chargement de l'image
        ImageIcon test = new ImageIcon(new ImageIcon(url).getImage()); //nouvelle icon et initialisation de l'image dans l'icon

        jPanel13.add(new JLabel(test)); //nouveau Label dans le Panel qui sert à afficher le résultat
    }

    jPanel13.repaint(); //remplace le JPanel vide par les images
    jPanel13.validate(); //validation du remplacement
}

/**
 * This method is called from within the constructor to initialize the form.
 * WARNING: Do NOT modify this code. The content of this method is always
 * regenerated by the Form Editor.
 */
@SuppressWarnings("unchecked")
Generated_Code

private void jButton14ActionPerformed(java.awt.event.ActionEvent evt) {
    jPanel13.removeAll();

    affichage_calcul(7);

    nombreProposer=7;
}

private void jButton5ActionPerformed(java.awt.event.ActionEvent evt) {
    jPanel13.removeAll();

    affichage_calcul(4);

    nombreProposer=4;
}

private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {
    jPanel13.removeAll();

```

```

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {

    jButton1.setEnabled(true);

    if(signe==false)
    {
        if(nombreAleatoire-nombreAleatoire2==nombreProposer || nombreAleatoire2-nombreAleatoire==nombreProposer)
        {

            p.add(" ");

            this.setContentPane(p);

            ScheduledExecutorService s = Executors.newSingleThreadScheduledExecutor();//fonction qui permet aux tâches de s'exécuter avec du délai

            s.schedule(new Runnable()
            {
                //on initialise le timer à 0
                @Override
                public void run() {

                    disable();

                    jButton2.setEnabled(true);
                    jButton2.setText("Rejouer");
                    jButton2.setSelected(false);

                    jButton1.setEnabled(true);

                    calcul.setText("GAGNER");

                    jPanel3.removeAll();

                    jPanel3.add(calcul);

                    jPanel2.setVisible(true);//met visible le JPanel qui affiche la calculatrice
                    w.setVisible(false);//rend invisible l'image de la pub

                }

            },(long) 2.8, TimeUnit.SECONDS);//timer en seconde, initialiser à 10 secondes
        }
    }
}

```

```

    }
    else
    {
        vie=vie-1;

        p.add(" ");

        this.setContentPane(p);

        ScheduledExecutorService s = Executors.newSingleThreadScheduledExecutor();//fonction qui permet aux tâches de s'exécuter avec du délai

        s.schedule(new Runnable()
        {
            //on initialise le timer à 0
            @Override
            public void run() {

                if(vie==3)
                {
                    calcul.setText(Calcul2);

                    jPanel3.removeAll();

                    jPanel3.add(calcul);//on ajout le JLabel calcul dans le JPanel qui sert à l'écriture
                    jPanel3.repaint();//on remplace
                    jPanel3.validate();//on valide le remplacement

                    jPanel2.setVisible(true);//met visible le JPanel qui affiche la calculatrice
                    l.setVisible(false);//rend invisible l'image de la pub

                    jLabel12.removeAll();

                    jLabel12.setIcon(image_vie2);
                    jLabel12.repaint();
                    jLabel12.validate();

                }

                if(vie==2)
                {
                    calcul.setText(Calcul2);//on initialise la police à 100

                    jPanel3.removeAll();
                }
            }

        },(long) 2.8, TimeUnit.SECONDS)
    }
}

```

```

    }
    if(vie==1)
    {
        if(nombreAleatoire>nombreAleatoire2)
        {
            nombreBon=(nombreAleatoire-nombreAleatoire2);
        }
        else
        {
            nombreBon=(nombreAleatoire2-nombreAleatoire);
        }

        resultat=(" "+nombreBon);

        perdu.setText(resultat);

        disable();

        jButton2.setText("Rejouer");

        jButton2.setEnabled(true);
        jButton1.setEnabled(true);

        JPanel3.removeAll();

        JPanel3.add(perdu); //on ajout le JLabel calcul dans le JPanel qui sert à l'écriture
        JPanel3.repaint(); //on remplace
        JPanel3.validate(); //on valide le remplacement

        JPanel12.setVisible(true); //met visible le JPanel qui affiche la calculatrice
        l.setVisible(false); //rend invisible l'image de la pub

        JLabel12.removeAll();

        JLabel12.setIcon(image_vie4);
        JLabel12.repaint();
        JLabel12.validate();
    }
}

```

```

public void run()
{
    nombreAleatoire = (int) (Math.random() * ( 10 ));

    if(nombreAleatoire == 9) //on test si nombreAleatoire est égale à 9
    {
        nombreAleatoire2 = (int) (Math.random() * ( 2 )); //si il est égale à 9 on initialise nombreAleatoire entre 0 et 1
    }
    else if(nombreAleatoire == 8)
    {
        nombreAleatoire2 = (int) (Math.random() * ( 3 ));
    }
    else if(nombreAleatoire == 7)
    {
        nombreAleatoire2 = (int) (Math.random() * ( 4 ));
    }
    else if(nombreAleatoire == 6)
    {
        nombreAleatoire2 = (int) (Math.random() * ( 5 ));
    }
    else if(nombreAleatoire == 5)
    {
        nombreAleatoire2 = (int) (Math.random() * ( 6 ));
    }
    else if(nombreAleatoire == 4)
    {
        nombreAleatoire2 = (int) (Math.random() * ( 7 ));
    }
    else if(nombreAleatoire == 3)
    {
        nombreAleatoire2 = (int) (Math.random() * ( 8 ));
    }
    else if(nombreAleatoire == 2)
    {
        nombreAleatoire2 = (int) (Math.random() * ( 9 ));
    }
    else {
        nombreAleatoire2 = (int) (Math.random() * ( 10 ));
    }
}

```

```

jPanel3.removeAll();//effacer tout ce qu'il y a dans le JPanel

if(nombreAleatoire3<1){
    //on ajout le calcul dans un nouveau JLabel
    if(nombreAleatoire<nombreAleatoire2)
    {
        operateur="-";
        signe=false;
        Calcul2=(nombreAleatoire1+operateur+nombreAleatoire+" = ");
    }
    else
    {
        operateur="-";
        signe=false;
        Calcul2=(nombreAleatoire+operateur+nombreAleatoire2+" = ");
    }
}
else{
    operateur="+";
    signe=true;
    Calcul2=(nombreAleatoire+operateur+nombreAleatoire2+" = "); //on ajout le calcul dans un nouveau JLabel
}

calcul.setText(Calcul2);

jPanel3.add(calcul);//on ajout le JLabel calcul dans le JPanel qui sert à l'écriture
jPanel3.repaint();//on remplace
jPanel3.validate();//on valide le remplacement

jPanel2.setVisible(true);//met visible le JPanel qui affiche la calculatrice
j.setVisible(false);//rend invisible l'image de la pub
}

},10, TimeUnit.SECONDS);//timer de 10 secondes

this.setVisible(true);//on met le JFrame en visible

j.setVisible(true);
jPanel2.setVisible(false);//on rend invisible l'affichage de la calculatrice

```

```

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    disable();//désactivation de tout les boutons

    p.add(b);

    this.setContentPane(p);

    ScheduledExecutorService s = Executors.newSingleThreadScheduledExecutor();//fonction qui permet aux tâches de s'exécuter avec du délais

    s.schedule(new Runnable()
    { //on initialise le timer à 0
        @Override
        public void run()
        {
            System.exit(0);//fermeture de l'application
        }
    },5, TimeUnit.SECONDS);//timer en seconde, initialiser à 10 secondes

    this.setVisible(true);//on met le JFrame en visible

    j.setVisible(true);

    p.add(jPanel2);//on ajoute le JPanel de la pub dans le JPanel de la calculatrice

    jPanel2.setVisible(false);
}

```

```

/**
 * @param args the command line arguments
 */
public static void main(String args[]) {
    /* Set the Nimbus look and feel */
    //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
    /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.
     * For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
     */
    try {
        for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {
            if ("Nimbus".equals(info.getName())) {
                javax.swing.UIManager.setLookAndFeel(info.getClassName());
            }
        }
    }
    catch (ClassNotFoundException ex) {
        java.util.logging.Logger.getLogger(Main.class).log(java.util.logging.Level.SEVERE, null, ex);
    }
    catch (InstantiationException ex) {
        java.util.logging.Logger.getLogger(Main.class).log(java.util.logging.Level.SEVERE, null, ex);
    }
    catch (IllegalAccessException ex) {
        java.util.logging.Logger.getLogger(Main.class).log(java.util.logging.Level.SEVERE, null, ex);
    }
    catch (ClassNotFoundException ex) {
        java.util.logging.Logger.getLogger(Main.class).log(java.util.logging.Level.SEVERE, null, ex);
    }
}

```

# Mémoires

## Intégration continue d'une application

### Introduction

Cet outil intervient dans le but de gagner du temps par l'automatisation de tâches récurrentes. Il est préférable de scripter l'ensemble de ces tâches et de laisser un outil les exécuter afin de ne pas créer d'erreurs lors du déploiement et des tests du projet informatique. Ce processus est généralement sans risque, car la plupart des outils fournissent une visualisation claire des actions en cours, et des éventuelles erreurs. C'est donc un outil qui s'intègre parfaitement dans une chaîne de déploiement continue ou d'intégration continue, les deux cas ayant pour but d'accélérer le processus de mise en production du projet informatique.

Si vous ne lancez vos tests qu'occasionnellement, l'utilisation d'un tel outil va vous permettre de rendre la chose systématique afin de rendre un projet de qualité supérieur. Une fois le changement effectué sur votre stack technique vous ne vous en passerez plus.

Tout comme la continuous delivery, l'intégration continue est avant tout privilégiée dans un environnement de développement logiciel agile. L'objectif de cette approche moderne est de progresser par étapes afin de concevoir le processus de développement plus efficacement et de pouvoir réagir aux modifications avec flexibilité. L'intégration continue a été évoquée pour la toute première fois dans la description de la méthode agile de l'Extreme Programming de Kent Beck. Mais a priori, l'idée d'une intégration continue serait antérieure. Elle intervient par exemple déjà dans la méthode Booch.

La continuous integration (en français « intégration continue ») est une technique de développement de logiciel agile. Pour ce type d'intégration, les développeurs intègrent les fragments de code terminés régulièrement, parfois plusieurs fois par jour, dans l'application au lieu de les intégrer tous en même temps à la fin du projet.

L'intégration continue nous fournit une bonne solution lorsque l'entreprise travaille sur un vaste projet ou un client souhaite avoir un logiciel à la fois complet et complexe. Différentes équipes travaillent à la conception de pans de l'application et les développeurs se chargent de programmer les différentes fonctionnalités. Après un travail de plusieurs mois voire de plusieurs années, l'intégralité du travail doit être regroupée et c'est alors que les problèmes surviennent. Dans un tel cas, la détection et la correction des erreurs, le regroupement de tous les fragments de code peut prendre plusieurs mois pour finalement se rapprocher de la phase de test finale et du déploiement.

Dans le cadre de la continuous integration, l'intégration du nouveau code est effectuée de façon bien plus précoce et pas uniquement lorsque toutes les parties prenantes ont terminé leur sous-domaine. Au lieu de cela, les développeurs intègrent leur code terminé une ou

plusieurs fois par jour dans la mainline, le code source qui est accessible par tous les programmeurs. Étant donné qu'il s'agit toujours dans ce cas de sections de code relativement courtes, l'intégration est elle aussi plutôt rapide. Seules quelques minutes sont nécessaires à un développeur pour mettre le résultat de son travail à disposition du reste de l'équipe. Si l'on découvre alors une erreur, elle peut être immédiatement localisée et, dans le meilleur des cas, corrigée rapidement.

### **Quand et pourquoi l'utiliser ?**

L'outil d'intégration continue, est-il en existe plusieurs avec chacun leurs avantages, va donc intervenir directement auprès des développeurs informatiques leur permettant d'automatiser des tâches récurrentes et répétitives. Bien configuré, cet outil va alors permettre la construction du projet puis la mise en place sur un environnement de préproduction. C'est sûr ce premier environnement que le projet va alors subir une batterie de tests qui peuvent être à la fois fonctionnels et techniques. Suivant l'outil mis en place il va soit effectuer les tests, ou tout simplement les déclencher en appelant un autre outil de test.

C'est une fois le projet testé et validé, que l'outil d'intégration continue va alors pouvoir déplacer le projet sur l'environnement de production où il sera alors exploité à 100% par les utilisateurs. Il est d'ailleurs possible que sur cet environnement final, le projet informatique subisse également d'autres tests.

Concernant l'outil à proprement parlé, il en existe des dizaines et on généralement tous une particularité. Cela peut-être le langage supporté ou les options supplémentaires de l'outil comme une supervision de l'ensemble du process ou la mise en place de tests supplémentaire. Cet outil intervient dans le but de gagner du temps par l'automatisation de tâches récurrentes. Il est préférable de scripter l'ensemble de ces tâches et de laisser un outil les exécuter afin de ne pas créer d'erreurs lors du déploiement et des tests du projet informatique. Ce process est généralement sans risque, car la plupart des outils fournissent une visualisation claire des actions en cours, et des éventuelles erreurs. C'est donc un outil qui s'intègre parfaitement dans une chaîne de déploiement continue ou d'intégration continue, les deux cas ayant pour but d'accélérer le processus de mise en production du projet informatique. Si vous ne lancez vos tests qu'occasionnellement, l'utilisation d'un tel outil va vous permettre de rendre la chose systématique afin de rendre un projet de qualité supérieur. Une fois le changement effectué sur votre stack technique vous ne vous en passerez plus.

### **Quels sont les avantages et les inconvénients ?**

Lors du travail quotidien, on constate souvent que l'intégration continue ne présente pas que des avantages en dépit de ses qualités. Si elle permet effectivement de faire l'économie d'une phase d'intégration longue et fastidieuse en fin de projet et de régler les problèmes de façon précoce, pour les équipes qui interviennent, le passage à l'intégration continue peut s'avérer très compliqué. Dans un tel cas, ce processus peut même demander plus de temps qu'il ne permet d'en économiser.

<b>Avantages</b>	<b>Inconvénients</b>
Possibilité de recherche précoce des erreurs	Conversion de processus habituels
Feedback permanent	Nécessite des serveurs et des environnements supplémentaires
Pas de surcharge contrairement à une seule grande intégration finale	Nécessité de mettre au point des processus de test adaptés
Enregistrement précis des modifications	Si plusieurs développeurs souhaitent intégrer leur code approximativement au même moment, des délais d'attente peuvent survenir
Disponibilité continue d'une version actuelle opérationnelle	
Nécessité d'un travail progressif	



# Liste et comparatif des outils de déploiement

## Introduction

Aujourd'hui, on trouve de nombreux outils d'intégration continue différents sur Internet. Ils ont tous vocation à aider les développeurs dans la mise en œuvre de l'intégration continue et y parviennent de différentes façons avec des fonctionnalités bien spécifiques. Mais les outils IC ne se distinguent pas uniquement par l'étendue de leurs fonctionnalités, on constate également de grandes différences en termes de prix et licence. Alors que bon nombre d'entre eux sont des logiciels open source disponibles gratuitement, certains fabricants proposent également des outils payants. Nous vous proposons un aperçu des programmes les plus appréciés et vous présentons leurs caractéristiques et leurs fonctionnalités.

### ❖ Jenkins



## Jenkins

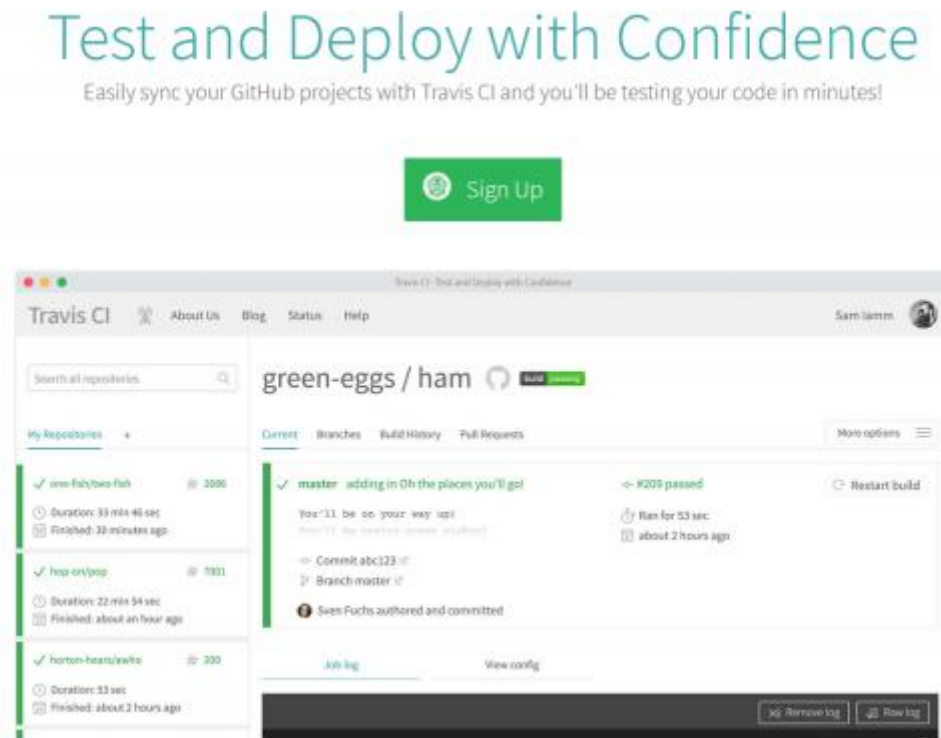
### 2019 Elections

- Governing Board
- Security Officer
- Events Officer

Le logiciel Jenkins est probablement l'un des outils IC les plus connus sur le marché. Depuis 2005 (à l'époque sous le nom de Hudson), le logiciel a été constamment amélioré. Aujourd'hui, ce logiciel programmé sous Java offre de nombreuses fonctionnalités et interfaces contribuant à faciliter non seulement l'intégration continue, mais aussi la livraison et le déploiement continus.

- Programmé sous Java
- Fonctionne dans un conteneur d'EJB
- Plus de 1 000 plugins
- Supporte également la livraison et le déploiement continus
- Peut-être combiné avec de nombreux systèmes de contrôle de version
- Contrôle via IGU (basée sur le Web), API REST ou commandes
- Hébergement sur le Cloud possible
- Gratuit
- Open source (licence MIT)

## ❖ Travis CI



Les personnes travaillant avec GitHub préféreront certainement Travis CI, car cet outil CI fonctionne parfaitement avec ce système de gestion des versions populaire. Le logiciel est paramétrable à l'aide d'un simple fichier YAML hébergé dans le répertoire racine du projet de développement. GitHub informe Travis CI de chaque modification apportée dans le dépôt et maintient toujours le projet à jour.

- Programmé sous Ruby
- Fonctionne sur toutes les plateformes
- Fonctionne avec GitHub
- Configuration à l'aide d'un fichier YAML
- Gratuit pour les projets open source
- Pour les projets commerciaux, coût compris entre 69 \$ et 489 \$ par mois
- Open source (licence MIT)

## ❖ Bamboo



La société Atlassian, qui propose également aujourd'hui le service d'hébergement de fichiers Bitbucket, distribue depuis 2007 l'outil d'intégration continue Bamboo. À l'instar de Jenkins, Bamboo assiste les développeurs dans l'intégration mais offre également des fonctionnalités pour le déploiement et la gestion des versions. Le travail avec cet outil s'effectue via une interface en ligne simple.

- Programmé sous Java
- Fonctionne sur toutes les plateformes
- Intégration simple d'autres produits Atlassian
- Grande quantité d'extensions
- Plusieurs tests possibles en simultanée
- Communication via une interface Web et API REST
- Gratuit pour les projets open source, les organisations à but non lucratif et les classes scolaires
- Pour toute autre utilisation, coût unique entre 10 \$ et 110 000 \$ en fonction du nombre de serveurs utilisés

## ❖ GitLab CI

### What Are The Advantages?

- **Integrated:** GitLab CI/CD is part of GitLab. You can use it for free on [GitLab.com](https://gitlab.com)
- **Easy to learn:** See our [Quick Start](#) guide
- **Beautiful:** GitLab CI/CD offers the same great experience as GitLab. Familiar, easy to use, and beautiful.
- **Scalable:** Tests run distributed on separate machines of which you can add as many as you want
- **Faster results:** Each build can be split in multiple jobs that run in parallel on multiple machines
- **Continuous Delivery (CD):** multiple stages, manual deploys, [environments](#), and [variables](#)
- **Open source:** CI/CD is included with both the open source GitLab Community Edition and the proprietary GitLab Enterprise Edition



GitLab CI est une composante du célèbre système de gestion des versions GitLab. En plus de l'intégration continue, GitLab offre un déploiement et une livraison continus. Tout comme pour Travis CI, la configuration de GitLab CI s'effectue via un fichier YAML. Par rapport à d'autres outils, le travail avec ce logiciel est également plus facile à d'autres égards.

- Composante de GitLab
- Programmé sous Ruby et Go
- Configuration à l'aide d'un fichier YAML
- Supporte également la livraison et le déploiement continus
- Open Core
- Auto-hébergement et hébergement sur le cloud disponible
- La version gratuite dispose uniquement de quelques fonctionnalités
- Le coût des autres versions est compris entre 4 \$ et 99 \$ par mois et par utilisateur

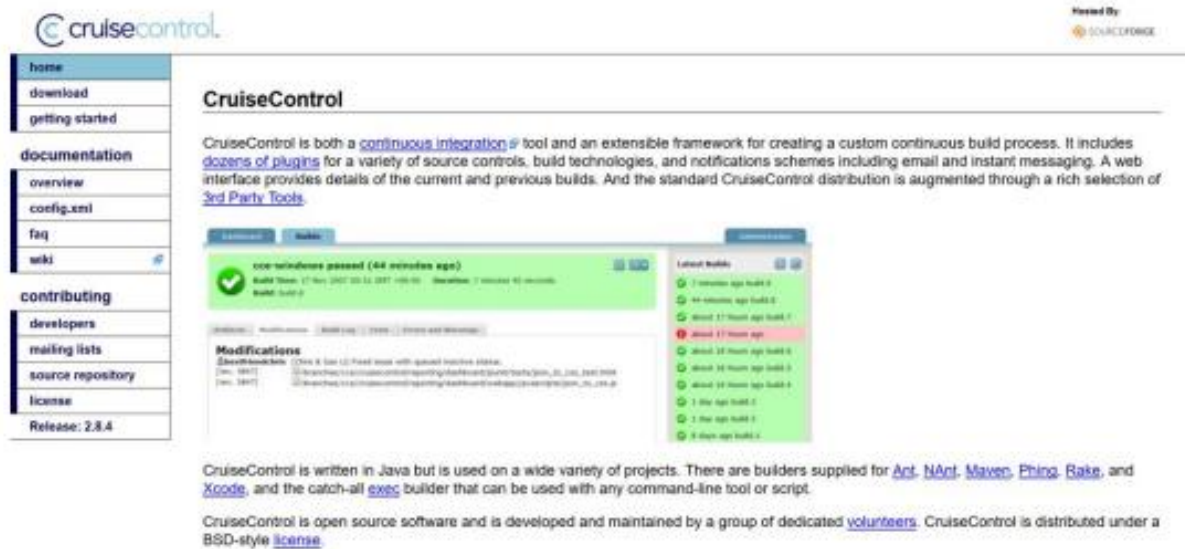
## ❖ Circle CI



L'outil d'intégration continue CircleCI fonctionne parfaitement avec GitHub et Bitbucket. Pour la phase de test, un conteneur ou une machine virtuelle sont utilisés. CircleCI accorde une grande importance à des processus de développement fluides, sans heurts, ce qui permet de mettre automatiquement à disposition des builds exempts d'erreur pour d'autres environnements.

- Configuration à l'aide d'un fichier YAML
- Supporte également le déploiement continu
- Auto-hébergèrent et hébergement sur le cloud disponible
- Fonctionne dans des conteneurs Docker, sous Linux VMs et MacOS VMs
- Gratuit pour un conteneur
- Autrement, coût compris entre 50 \$ et 3 150 \$ par mois

## ❖ CruiseControl



L'outil CruiseControl fait partie des plus anciennes applications proposant une intégration continue. Cet outil a été introduit sur le marché dès 2001 et a été constamment amélioré depuis, notamment par Martin Fowler qui est l'un des pionniers dans le domaine de l'intégration continue. Outre un tableau de bord clair, les développeurs disposent également de nombreux plugins facilitant leur travail.

- Programmé sous Java
- Fonctionne sur toutes les plateformes
- Tableau de bord basé sur le Web
- Des versions pour Ruby (CruiseControl.rb) et .NET (CruiseControl.NET) sont disponibles
- Open source (licence BSD)
- Gratuit

## ❖ Codeship

L'outil IC Codeship appartient aujourd'hui à CloudBee qui dispose également de Jenkins dans son portefeuille. Ce programme est disponibles en deux versions : la version de base offre une interface Web facile d'utilisation alors que la version pro est configurée à l'aide de fichiers dans le dépôt. Les développeurs souhaitant travailler avec un conteneur Docker devront opter pour la version pro.

- Interface Web dans la version de base
- Fichiers de configuration dans le dépôt pour la version pro
- Docker supporté dans la version pro
- Gratuit pour 100 builds par mois en cas de pipeline test
- Coût compris entre 75 \$ et 1 500 \$ par mois



## ❖ TeamCity



Le logiciel TeamCity séduit avant tout par ses « gated commits » : grâce à ces derniers, TeamCity teste les modifications apportées au code avant même qu'elles ne soient insérées dans la mainline. Le code source est uniquement intégré au code base pour toute l'équipe lorsqu'il est exempt d'erreur. TeamCity effectue les tests de façon autonome en arrière-plan de telle sorte que les développeurs peuvent poursuivre leur travail dans l'intervalle.

- Programmé sous Java
- Fonctionne sur toutes les plateformes
- Gated Commits
- Gratuit pour 100 builds avec 3 agents de build
- Coût unique compris entre 299 € et 21 999 €
- 50 % de remise pour les start-ups et gratuit pour les projets open source

	déploiement continu supporté	hébergement sur le Cloud	licence	prix pour l'offre payante	version gratuite	particularité
Jenkins	✓	✓	MIT	-	✓	nombreux plugins
Travis CI	✗	✓	MIT	69-489 \$ par mois	✓	connexion directe à GitHub
Bamboo	✓	✓	propriétaire	coût unique de 10-110 000 \$	✓	
GitLab CI	✓	✓	MIT/EE	4-99 \$ par mois	✓	connexion directe avec d'autres produits Atlassian
Circle CI	✓	✓	propriétaire	50-3 150 \$ par mois	✓	utilisation simple
CruiseControl	✗	✗	BSD	-	✓	entièrement gratuit
Codship	✓	✓	propriétaire	75-1 500 \$ par mois	✓	version de base et pro
TeamCity	✓	✗	propriétaire	coût unique de 299-21 999 €	✓	Gated Commits

## Conclusion

Pour conclure, ce qu'il faut retenir de l'intégration continue c'est le test permanent du code tout au long du développement du projet. On retiendra que plus les bugs seront découverts tôt dans la phase de projet, moins le coût des correctifs sera important et plus grande sera la productivité. Néanmoins maintenir une plateforme d'intégration continue n'est pas toujours simple. Il faut savoir peser les pour et les contre : à savoir de détacher une ressource pour administrer l'outil et le faire évoluer en fonction des besoins.

# Manuels

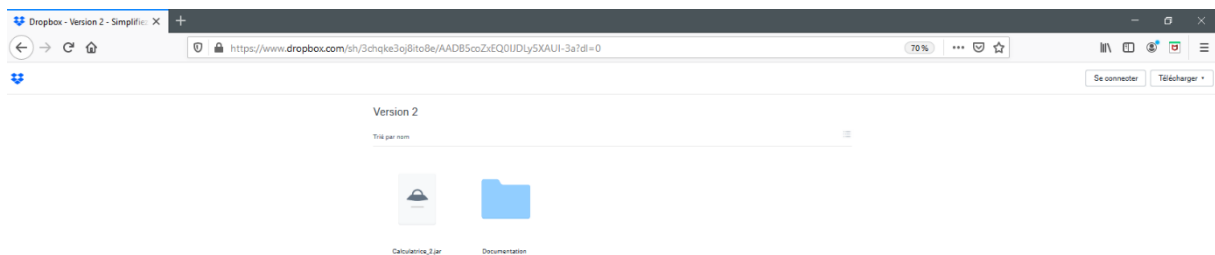
## Manuel d'installation

### Trouver les fichiers :

Entrez le lien suivant dans la barre de recherche :

<https://www.dropbox.com/sh/ya9iw5ogs3fnmjv/AACbWMceSm9KvCE-xP4TpzMNa?dl=0>

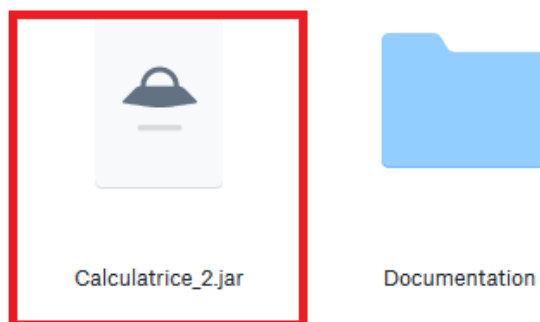
Vous arriverez ici :



Cliquez ensuite sur le fichier « Calculatrice\_1.jar »

### Version 2

Trié par nom



**Cliquez**

Vous arrivez ensuite sur cette page :

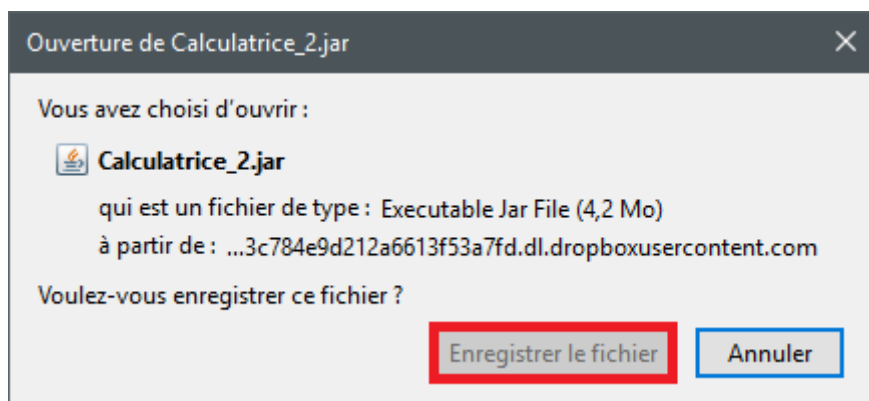


Cliquez sur « Télécharger » puis sur « Téléchargement direct »

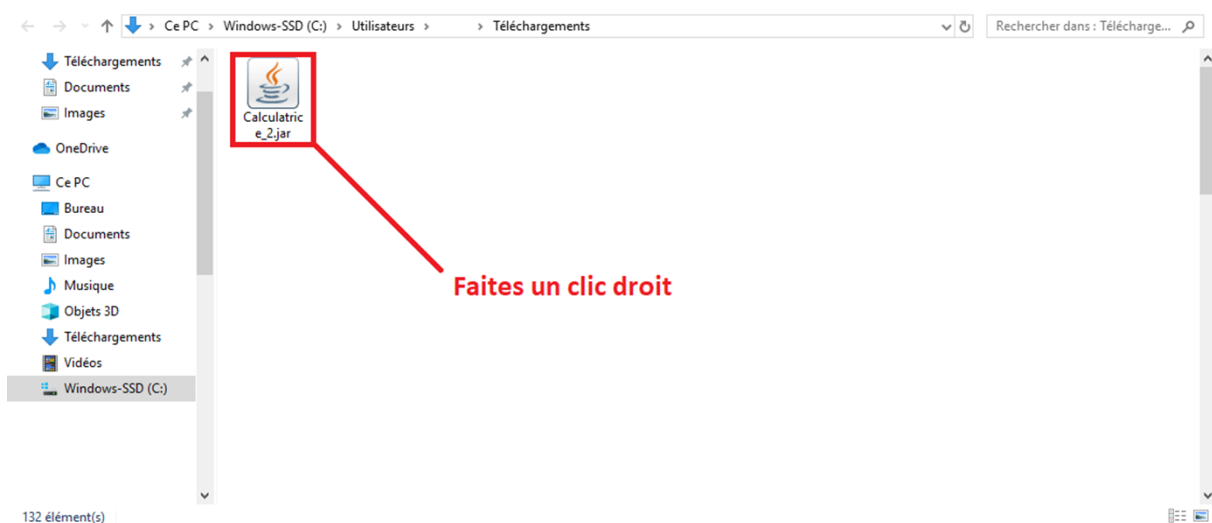




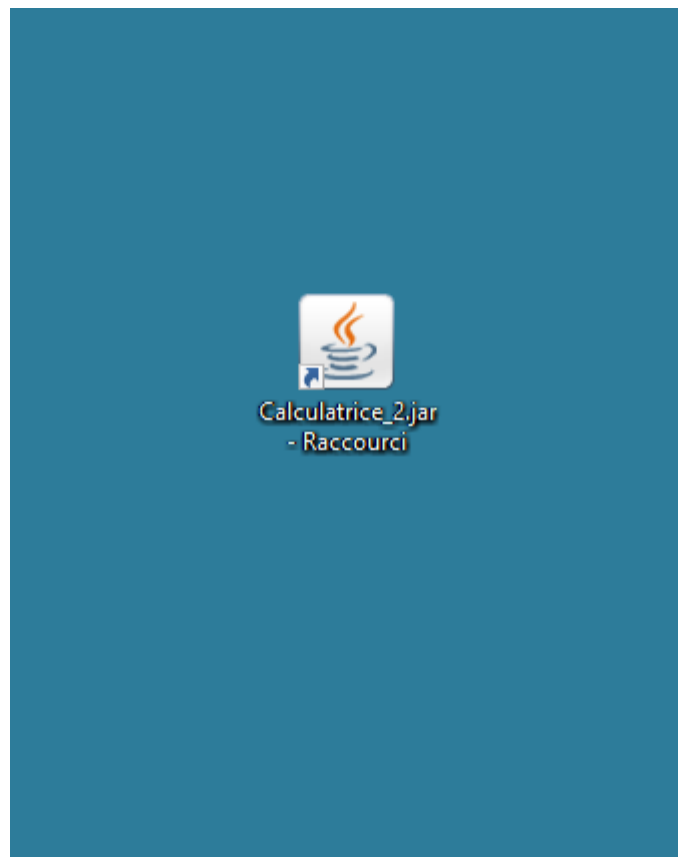
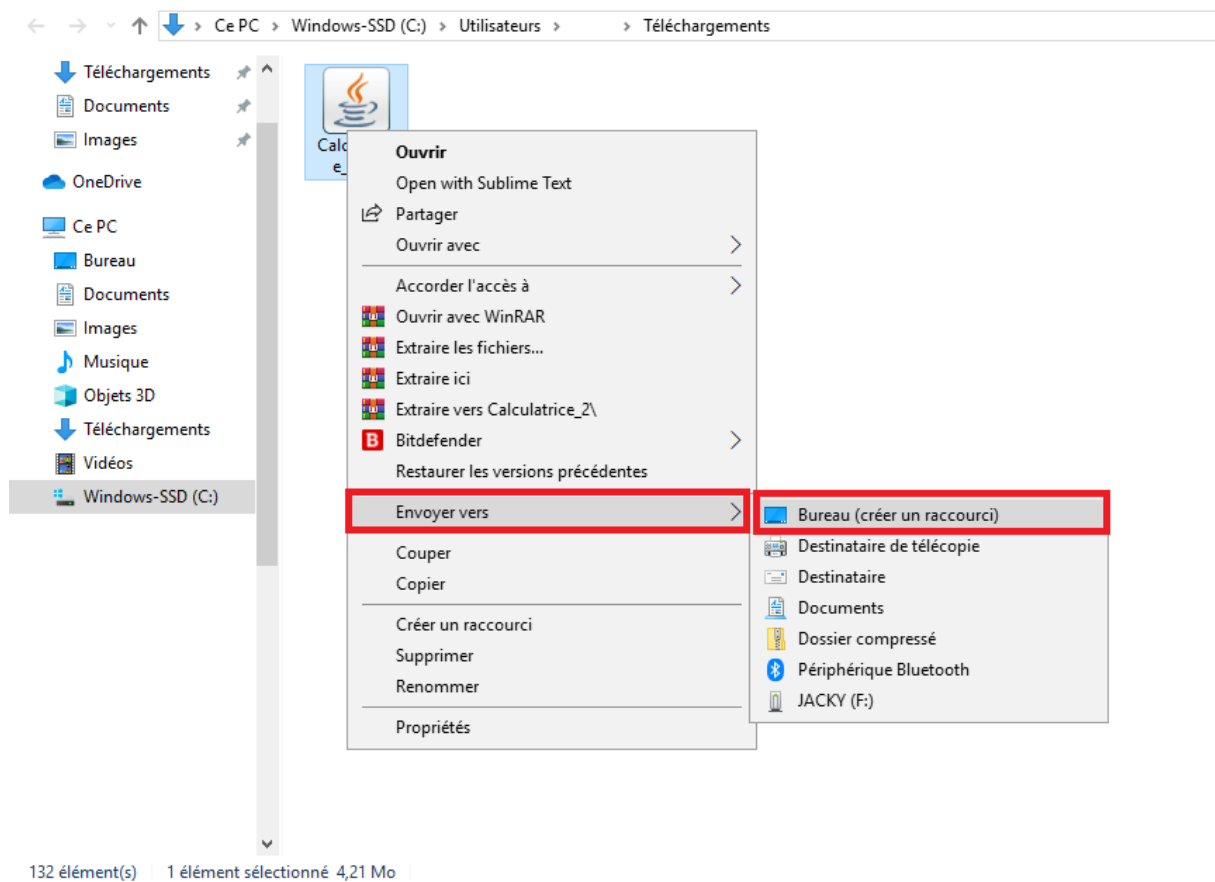
Enregistrez le fichier



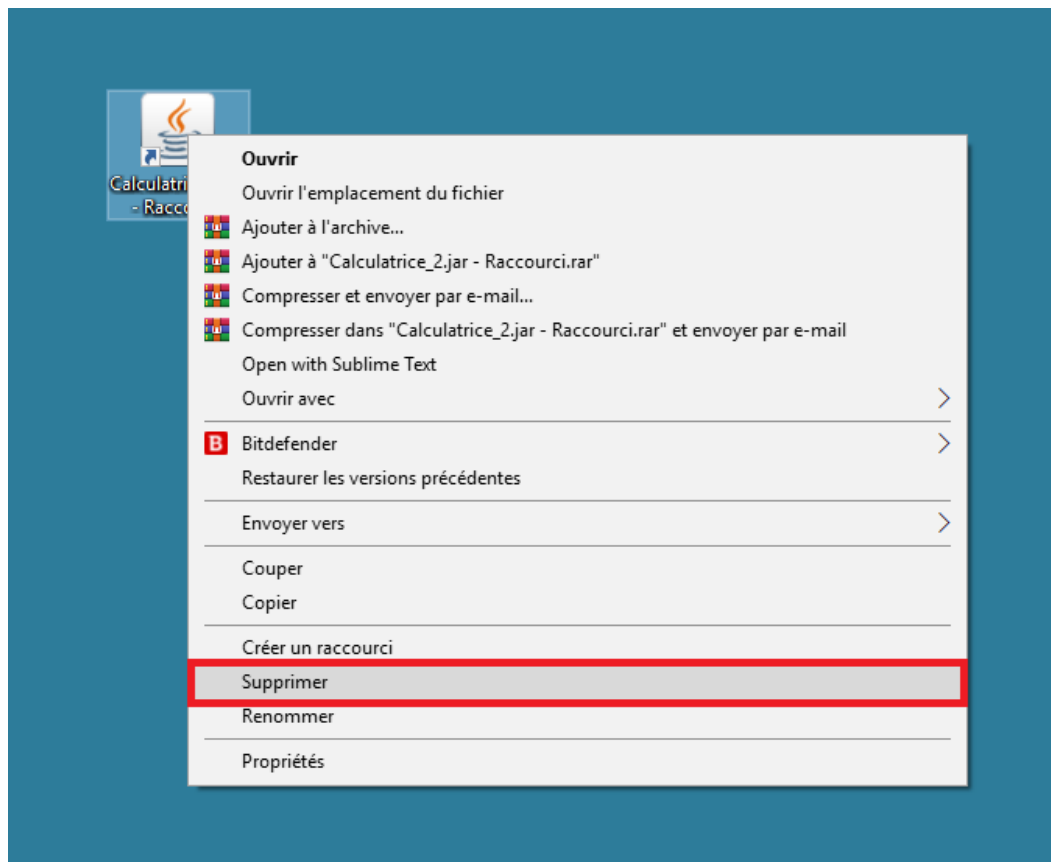
Ouvrez l'emplacement du fichier puis faites un clic-droit sur le fichier.jar que vous venez de télécharger



## Créez un raccourci (Envoyer vers > Bureau (Créer un raccourci))



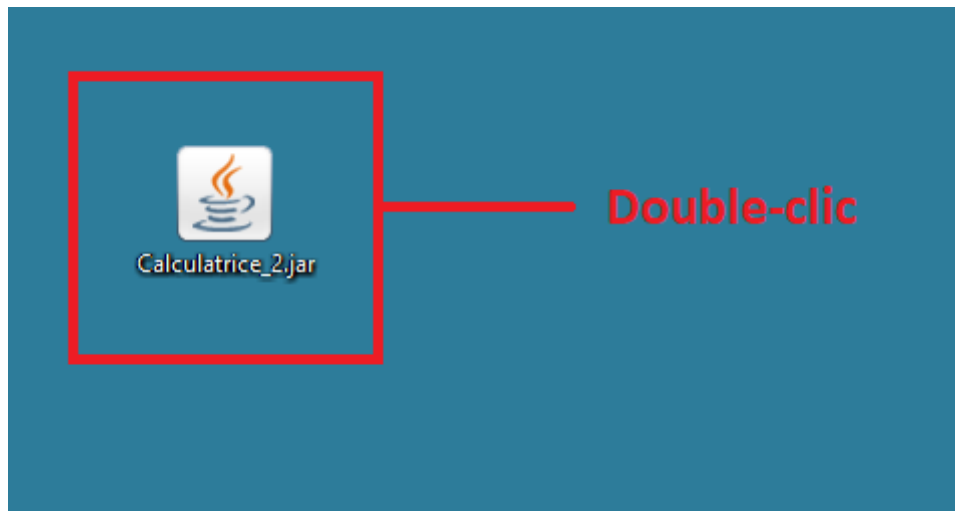
Pour le supprimer, faites un clic-droit sur le fichier puis appuyer sur « Supprimer »



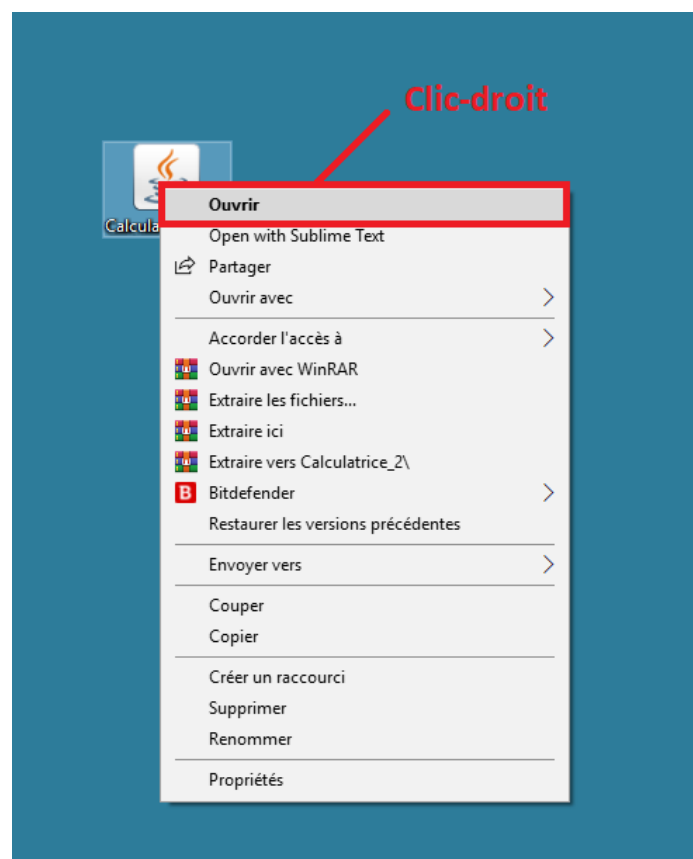
# Manuel utilisateur

- **Allumer la calculatrice**

Allez sur le bureau, puis double-cliquer sur l'application, où bien faites un clic gauche puis appuyez sur «ouvrir»



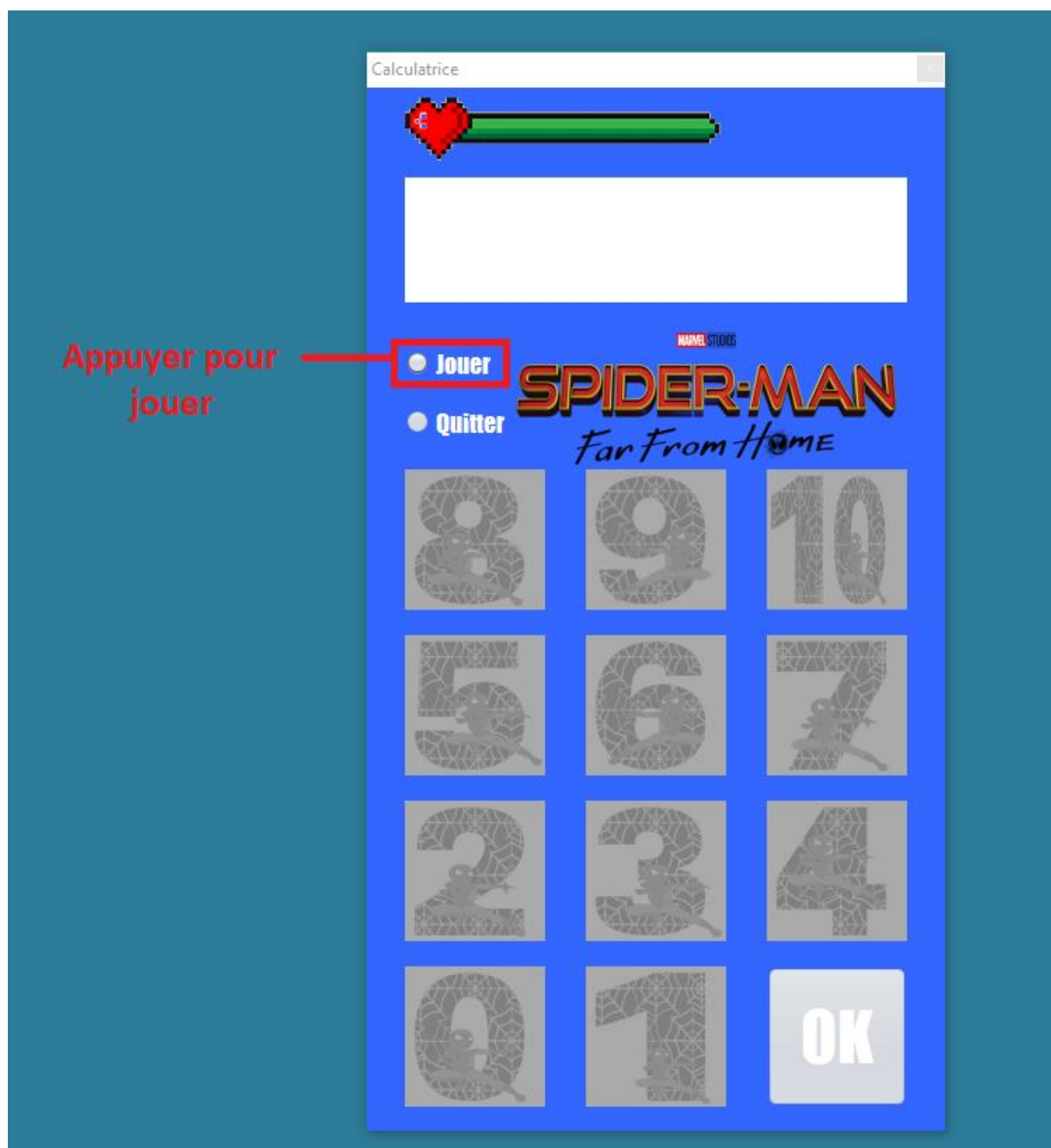
Ou





- Une fois dans la calculatrice

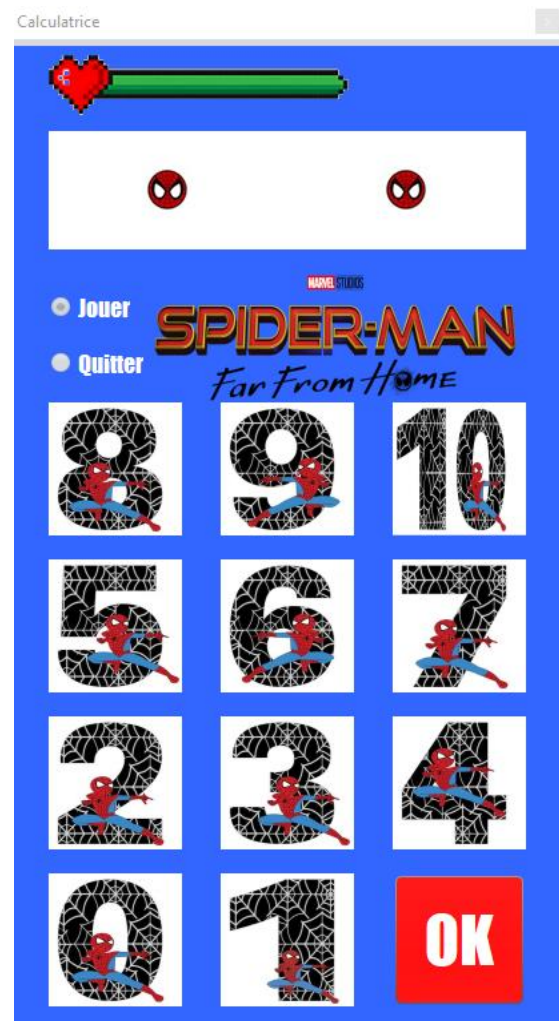
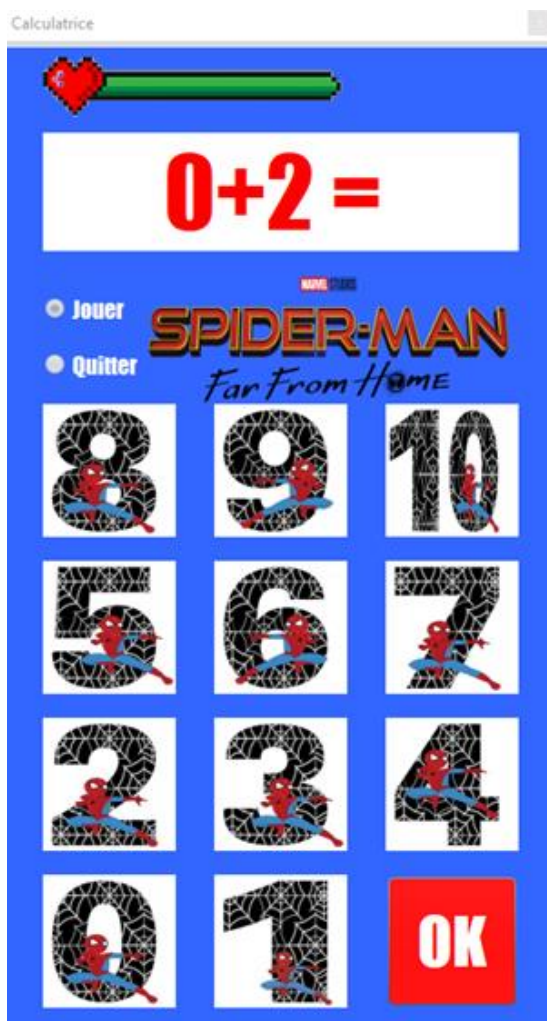
Appuyez sur le bouton « Jouer » pour mettre la calculatrice en marche



Une pub Spiderman s'affichera pendant 10 secondes

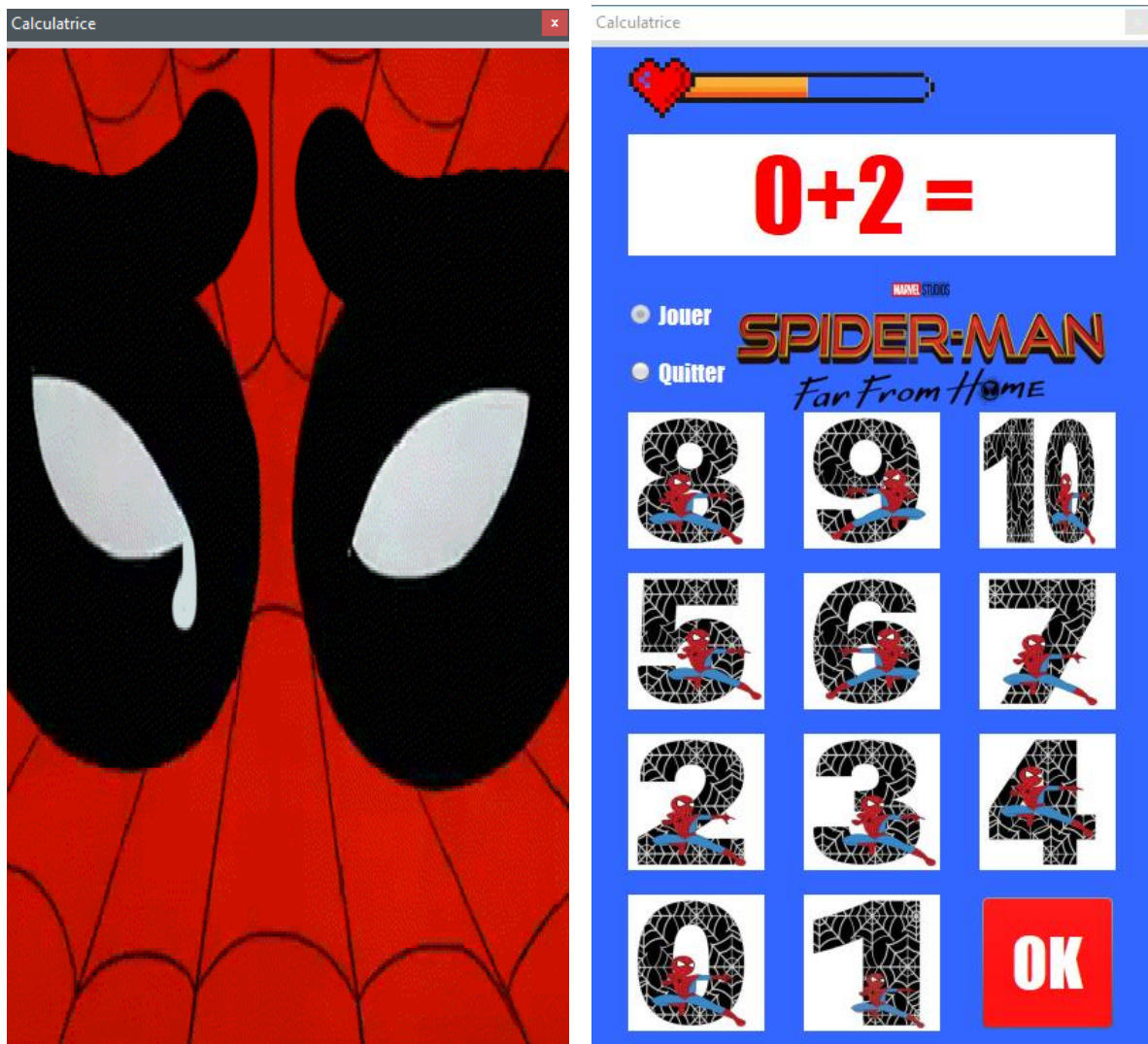


Après la pub, un calcul aléatoire sera généré, vous pourrez ensuite choisir votre réponse :



Pour valider la réponse, appuyez sur le bouton « OK »

❖ Si vous ne répondez pas correctement



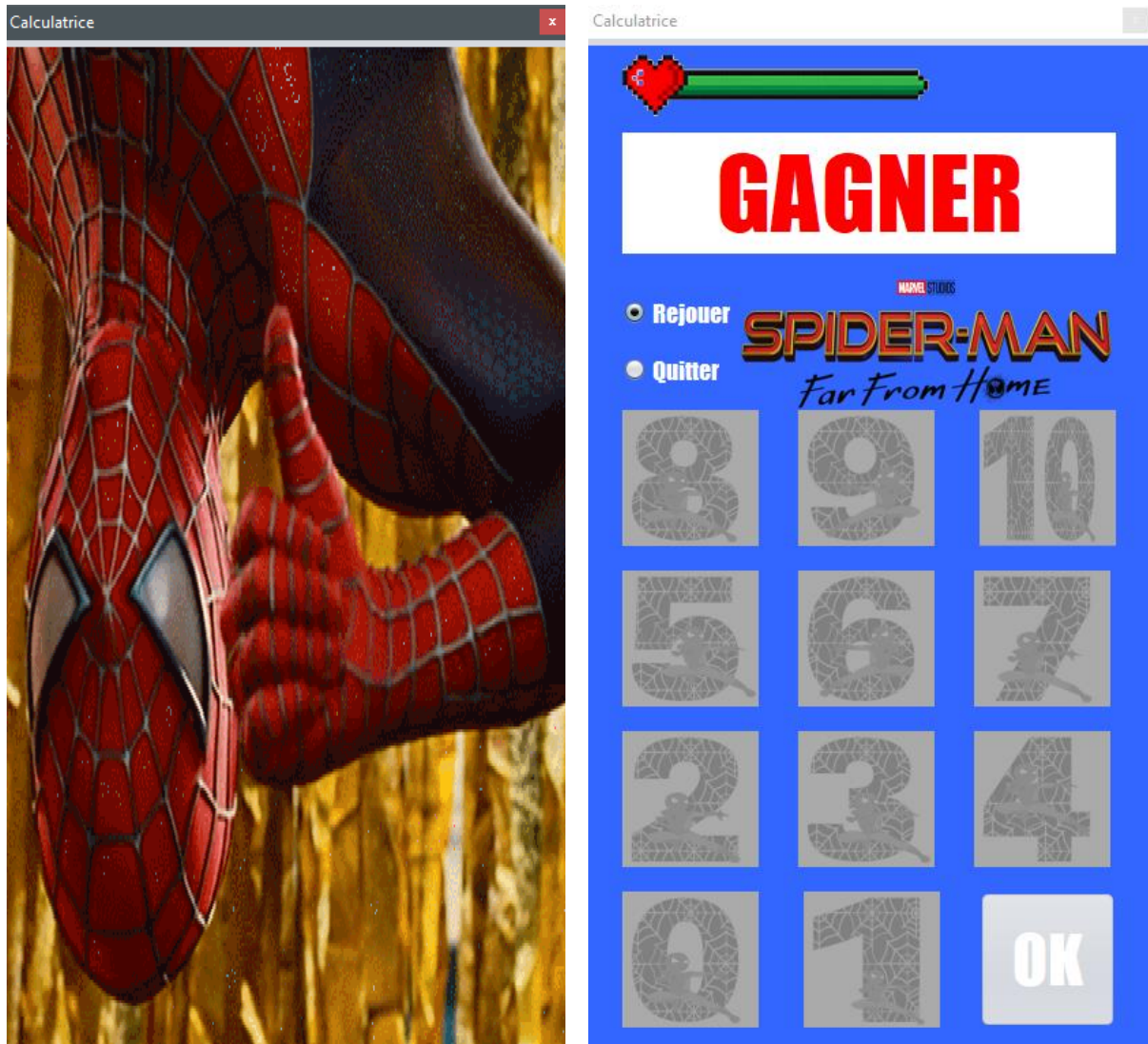
Si vous n'avez pas répondu correctement, vous aurez perdu une vie sur les trois qui vous sont proposées. Une fois les vies épuisées, la calculatrice affichera la réponse du calcul.



**Vous pourrez appuyer sur rejouer pour relancer la calculatrice ainsi qu'un nouveau calcul aléatoire, ou appuyer sur quitter pour l'éteindre**



❖ Si vous avez gagné



**Vous pourrez appuyer sur rejouer pour relancer la calculatrice ainsi qu'un nouveau calcul aléatoire, ou appuyer sur quitter pour l'éteindre**

# Bilan

## ❖ Synthèse

Le développement de la version 2 de la calculatrice EditCALC était intéressante car elle ajoutait des fonctionnalités que nous avons déjà envisagé lors du développement des premiers prototypes de la version 1.

## ❖ Les objectifs ont-ils été atteints ?

Oui, nous avons bien respecté le cahier des charges

## ❖ En quoi le développement du projet nous a apporté quelque chose

Nous avons pu en découvrir plus sur l'interface Homme-Machine avec Swing et nous avons pu renforcer nos connaissances en JAVA. Lors de la phase de développement, nous nous sommes rendu compte à quel point l'aspect fonctionnel mais également l'aspect esthétique étaient importants à prendre en compte lors du développement d'une application. Un client doit se sentir à l'aise avec l'application, l'application doit être attrayant, et facile d'utilisation, des facteurs clés de succès qui ne doivent pas être négligés.

## ❖ Quelles ont été les difficultés rencontrées

La plupart des nouvelles fonctionnalités étaient facile à faire et à intégrer. Sauf une qui nous a demandé un peu plus de temps et de ressources : l'intégration d'un système de vie et l'affichage de la réponse. Ayant beaucoup moins de temps que pour le développement de la version 1, nous devons nous organiser afin de mieux gérer ce risque. Dans un premier temps, nous avons donc accompli les tâches les plus facile, puis nous nous sommes tous concentrés sur cette même tâche.

## Conclusion

Le développement de la version 1 nous a appris beaucoup de choses que nous avons su mettre en application lors du développement de la version 2. En effet, nous avons su mieux nous organiser afin de mieux comprendre les attentes du client, et mieux organiser les tâches à faire afin de minimiser les pertes de temps inutiles.