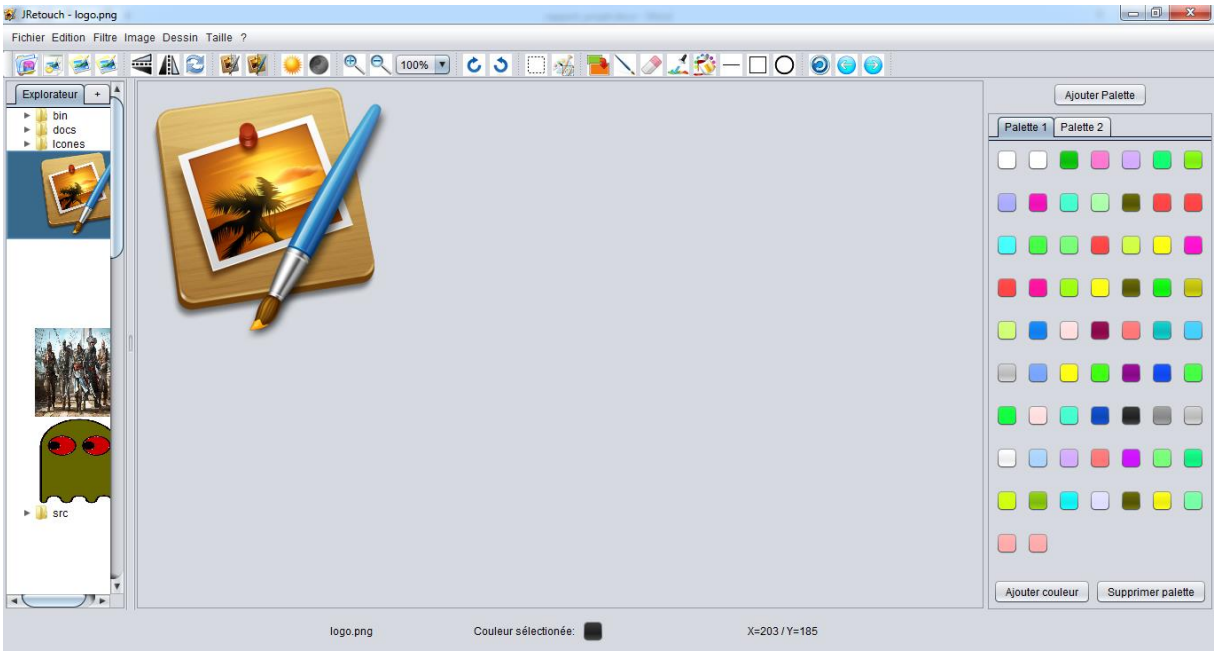
Thomas Janowiez

Géraud Métais

Marc Pescher

Jérémy Talabard

JRetouch



Tuteur du projet : Sylvain SPEH

IUT de Clermont-Ferrand Département informatique Année 2013-2014

UdA | Université d´Auvergne

Nous autorisons la diffusion de notre rapport sur l'intranet de l'IUT.

Nous remercions Sylvain SPEH pour nous avoir suivis tout au long du projet. Et de nous avoir apporté ses connaissances.

Nous remercions également l'ensemble des professeurs de l'IUT, grâce auxquels, nous avons acquis un ensemble de connaissances nécessaires pour bien commencer ce projet.

Nous remercions aussi l'IUT des Cézeaux pour nous avoir fourni une salle pour travailler sur notre projet.

Introduction

Dans le cadre de notre deuxième année de DUT informatique, nous avons eu la chance de réaliser un projet avec pour tuteur M. Sylvain SPEH, vacataire à l'IUT. Ce projet consiste à concevoir un logiciel de retouche d'images en Java, langage de programmation orienté objet. Un logiciel de retouche d'images permet, comme son nom l'indique, de modifier des photos ou images dans le but de les améliorer dans la majorité des cas.

Nous avons choisi ce sujet, car nous avons tous eu l'occasion d'utiliser ce type de logiciel et notre curiosité nous a poussés à apprendre et comprendre le fonctionnement de celui-ci.

La problématique du projet est la suivante : Comment réaliser un logiciel de retouche d'images en Java ? Pour répondre à cette question, nous avons élaboré le plan suivant : tout d'abord, nous présenterons le sujet, ensuite le travail qui a été réalisé et enfin, les résultats et les manques du projet.

Sommaire

I. Présentation du projet

1. Sujet

Le sujet du projet sur lequel nous avons travaillé est la réalisation d'un logiciel de retouche d'images. Ce logiciel doit être réalisé en Java (1). Il doit reprendre un ensemble de fonctionnalités présentes dans tout logiciel, à savoir le fait de pouvoir créer un fichier, ouvrir un fichier et sauvegarder un fichier, tout en adaptant cela au monde des images/photos. Il doit également posséder des fonctions propres aux logiciels de retouche d'image :

- Zoomer -/+
- Rotation horaire / antihoraire
- Effectuer des effets miroir
- Inverser une image (gauche/droite)
- Réduire le nombre de couleurs d'une image
- Redimensionner une image
- Remplacer une couleur par une autre
- Dessiner sur l'image
- Dessiner des formes diverses sur une image
- Inverser les couleurs
- Assombrir / Éclaircir
- Passer une image en niveau de gris
- Extraire des bouts d'image

Nous avons choisi de nommer notre application JRetouch., le J faisant référence au langage de programmation Java (1) avec lequel nous avons créé cette application et Retouch car ce logiciel sert à retoucher des images.

2. Le cahier des charges

Après avoir choisi le sujet de notre projet, nous nous sommes réunis avec M. SPEH et tout le groupe de projet. Durant cet entretien, notre tuteur nous a exposé le cahier des charges suivant.

A.D'un point de vue technique

L'application doit être réalisée en Java (1) ce qui va de pair avec une autre attente, le fait qu'elle soit multiplateforme (2). En effet, c'est une des principales particularités du langage Java (1). Cette portabilité est obtenue grâce à la compilation de ce langage : contrairement aux langages compilés traditionnels, pour lesquels le compilateur (3) crée un fichier directement exécutable, le code source Java (1) est compilé en un langage intermédiaire dans un fichier portant le même nom que le fichier source à l'exception de son extension (4). Ensuite, ce fichier est interprété par une machine virtuelle. Donc, pour peu qu'une plate-forme possède une machine virtuelle (5) fonctionnant sous son système, celle-ci est capable d'exécuter n'importe quel programme.

La réalisation de cette application doit utiliser au moins un fichier de paramétrage autrement appelé fichier de properties (6). Ce fichier va contenir un ensemble d'informations représentées par un ensemble de pairs clé=valeur et va servir à la configuration de l'application notamment pour le menu et la palette de couleur que nous verrons plus tard.

Un manuel utilisateur doit être également joint à notre logiciel pour permettre de guider l'utilisateur s'il en ressent le besoin. De plus, une Javadoc (7) doit accompagner le code et le livrable pour permettre une meilleure compréhension du code ce qui facilitera le travail d'un développeur qui pourrait éventuellement reprendre le développement de JRetouch.

La mise en place de la fenêtre de l'application est aussi fixée par le cahier des charges. L'interface doit être semblable à celle de la figure 1 de l'annexe, avec en haut un menu avec une barre d'outils, à gauche un explorateur avec une miniature des images du répertoire courant (au travers desquelles il sera possible d'ouvrir directement une image ou de la supprimer), à droite une palette de couleur et en bas une barre de statut (position de la souris, nom de l'image ouverte, couleur sélectionnée).

B.Point de vue organisationnel

Le cahier des charges prévoit la création d'un blog sur lequel nous avons renseigné l'avancement du projet. M. SPEH étant un intervenant extérieur à l'IUT, ce blog est aussi l'occasion de suivre l'avancement du projet car un compte-rendu est mis en ligne toutes les semaines avec le code et un exécutable du projet. De plus, la mise en place de ce blog nous permet également de partager notre travail et pourquoi pas donner l'envie à des personnes qui le consulteraient de se lancer aussi dans le développement d'une application semblable.

M. SPEH a voulu la réalisation d'un calendrier prévisionnel et réalisé pour savoir qui a fait quoi et quand. Le calendrier prévisionnel nous a permis de dégager les fonctions principales à réaliser ainsi que leur importance répercutée sur la durée que nous leurs avons attribués. Ensuite, le calendrier réalisé permet de refléter le travail réalisé tout au long du projet et de comparer par rapport au prévisionnel pour éventuellement voir les tâches qui nous ont posé des difficultés et faire ressortir les problèmes qui nous ont bloqués. Ce point correspond à une autre attente du cahier des charges, à savoir une liste des problèmes rencontrés et leur hypothétique résolution permettant à notre tuteur de comprendre les ralentissements causés par ceux-ci.

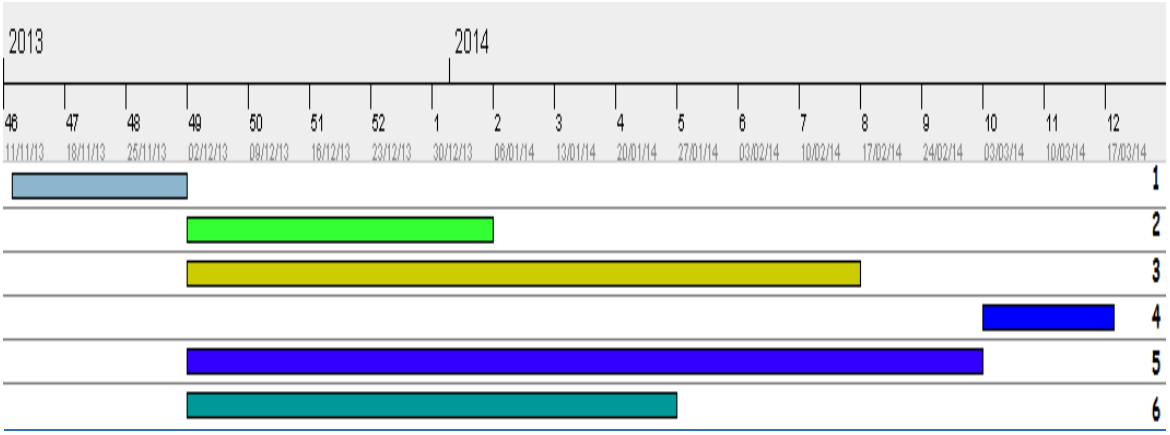
3. Organisation

A.Le diagramme de Gantt prévisionnel

Dès le début du projet, avant de se lancer directement sur le code, nous avons réalisé le diagramme de Gantt (8) suivant:

GANTT project

	Nom	Date de début	Date de fin
1	Analyse et recherche	12/11/13	01/12/13
2	Copier/coller/couper une zone, fonctions de dessins	02/12/13	05/01/14
3	Gestion des fonctions sur une image (rotation, filtres, rempli...	02/12/13	16/02/14
4	Rédaction du rapport	03/03/14	17/03/14
5	Rédaction du manuel utilisateur	02/12/13	02/03/14
6	Gestion de la fenêtre (menu, palette de couleurs, explorateur...	02/12/13	26/01/14



Ce diagramme prévoit la réalisation de six principales tâches :

- L'analyse et recherche : avant de commencer tout développement, nous nous sommes tous les quatre penchés sur l'analyse et la recherche. En effet pour pouvoir bien débuter, nous avons effectué une analyse qui nous a permis de dégager les acteurs de notre application et plus précisément les différents composants de l'interface de JRetouch. Ne sachant pas comment les images sont gérées en Java (1), une période de recherche a été nécessaire pour apprendre les bases sur cette gestion.
- Les fonctions de copier/couper/coller, de dessins (et formes géométriques). Marc s'est proposé de réaliser ces fonctions.
- Les fonctions sur une image c'est-à-dire les fonctions de retouche. Pour ces fonctions, Thomas et Géraud se sont occupés de leurs élaborations. Nous avons choisi de mettre deux membres du groupe pour cette tâche car c'est la partie la plus importante à réaliser.
- La rédaction du rapport : cette tâche sera réalisée par tous les membres du groupe deux semaines avant la fin du projet.
- Le manuel utilisateur qui devait être réalisé tout au long de l'avancement du projet.
- L'interface de JRetouch : Jérémy s'est occupé de réaliser la fenêtre de l'application.

B.Le diagramme de Gantt réalisé

Pour la répartition des tâches, le diagramme de Gantt (8) prévisionnel a bien été respecté à savoir, Jérémy s'est bien occupé de la fenêtre, Marc des fonctions de copier/coller/couper et des fonctions de dessins et Thomas et Géraud ont réalisé toutes les fonctionnalités de retouche d'images.

(Voir les figures 2 et 3 de l'annexe pour les diagrammes de Gantt (8) réalisé)

En revanche, nous nous sommes rendu compte que nous avions sous-estimé la durée de plusieurs tâches. En effet, pour la réalisation de l'interface le temps prévu a été largement dépassé en raison de la complexité de la conception de l'explorateur et il en est de même pour l'élaboration des fonctions pour copier/couper/coller une zone de l'image.

Enfin, nous nous sommes également aperçus que la rédaction du manuel utilisateur ne devait se faire que vers la fin du projet car au fur et à mesure de l'avancement du projet, d'une part le visuel de notre application changeait donc les copies d'écran auraient été à refaire et d'autre part quelques fonctionnalités se sont vues simplifiées ce qui aurait également entraîné une modification sans cesse du manuel utilisateur et donc une perte de temps non négligeable.

Nous allons maintenant voir le travail qu'il y a à réaliser sur ce projet, en commençant par l'analyse.

II. Travail

1. Analyse

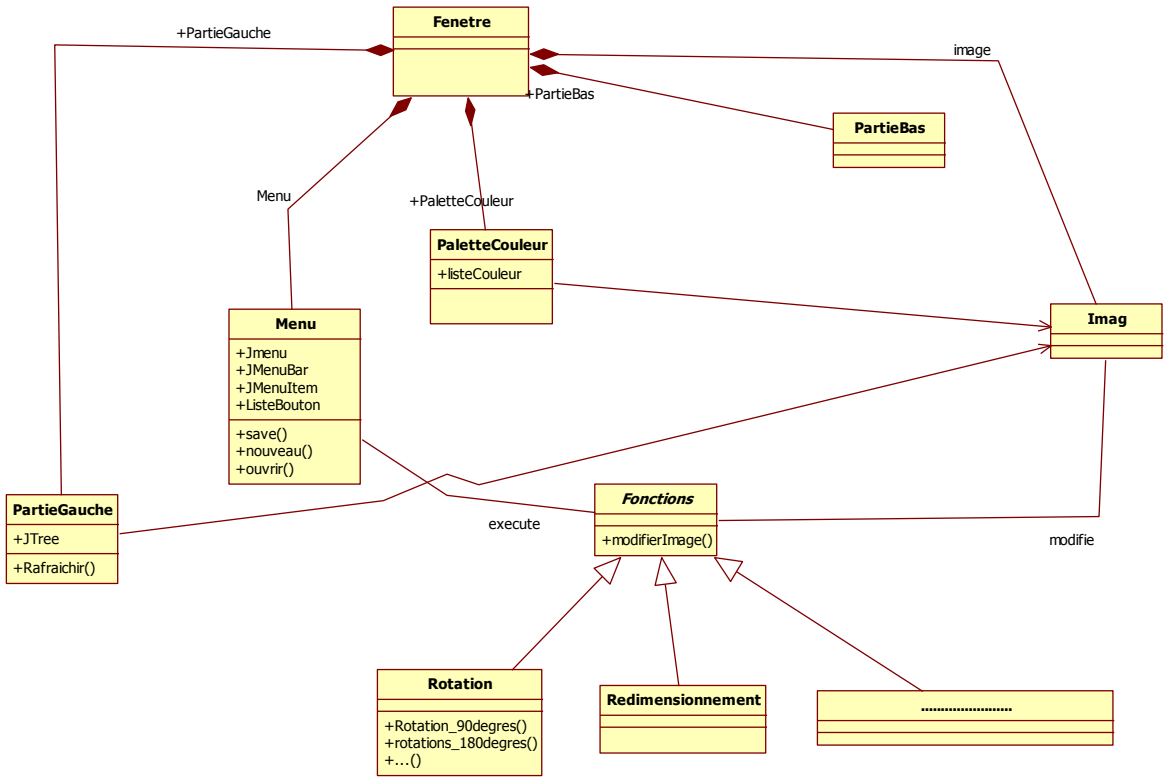
Partant d'un cahier des charges clair, il nous fallait maintenant déterminer les classes et les objets les plus adéquats pour notre projet.

D'après les schémas de M.SPEH, nous avons pu voir qu'il y avait cinq parties significatives à savoir un Menu, une Palette de Couleurs, la Partie Du Bas qui est la barre de statut, la Partie de Gauche qui est l'explorateur d'images et la partie centrale qui est la partie où l'on visualise l'image.

Les cinq parties sont des composantes de la Fenêtre.

Nous avons aussi identifié les Fonctions exécutables depuis le menu, toutes les fonctions héritent de la classe abstraite (9) Fonctions.

Voilà une esquisse de notre diagramme de classe (14).



Depuis ce diagramme basique va découler tout le reste de l'analyse.

A.Architecture en Package

Nous trouvons utile d'utiliser un MVC (12) (Modèle-Vue-Contrôleur). Pour cela on va structurer notre analyse et notre code avec des packages (Paquets).

Un package (13) Contrôleur où l'on trouvera tous les Listeners (Ecouteurs). Un Listener va écouter les composants (Menu, Palette couleur, …) et quand un de ces composants est modifié, le Listener en est informé et va exécuter la méthode correspondante au changement. C'est-à-dire par exemple lorsqu'on appuie sur un bouton un Listener va en être informé et exécute la fonction correspondante à ce bouton. Ce package (13) possède aussi la classe (10) Main, c'est cette classe (10) qui démarre l'application.

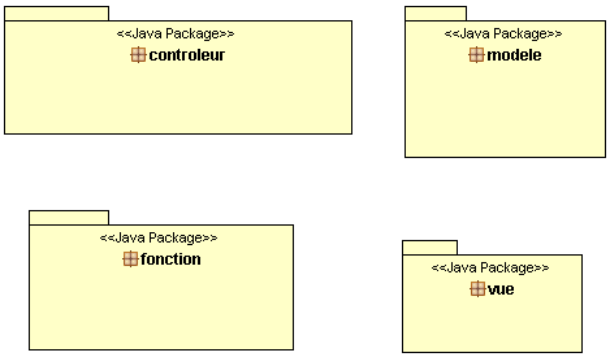
Un package (13) Vue qui contient toutes les classes (10) représentant des objets qui s'affichent à l'écran. (Menu, Fenêtre, Dialogue, …).

Un package (13) Modèle qui regroupe toutes les classes (10) qui seront chargés de stocker les données nécessaires à un calcul et d'avoir le résultat.

Un package (13) Fonction qui regroupe toutes les fonctions à exécuter sur l'image, c'est-à-dire les classes (10) exécutant des calculs sur l'image.

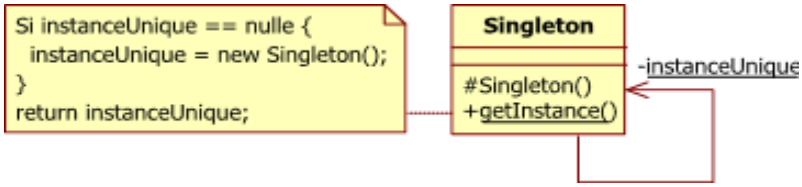
En fait, le package (13) Fonction et le package (13) Modèle pourraient être regroupés ensemble car leurs principes de fonctionnement est le même, faire des calculs ou des opérations modifiant l'état d'un composant de la Vue. Néanmoins pour plus de lisibilité du diagramme et par la suite pour la lisibilité du code, nous avons préféré les séparer.

Voici le diagramme des paquetages correspondant:



B.Patrons de conception

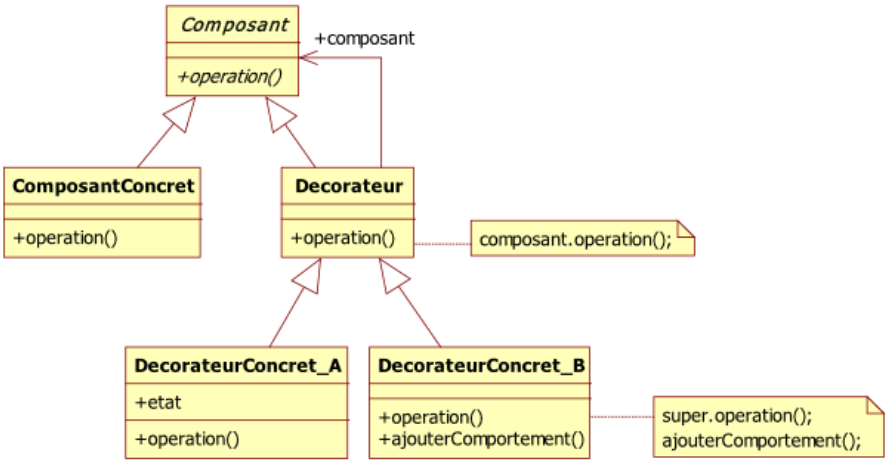
Il nous est judicieux d'utiliser dans notre application le patron de conception (15) Singleton.



Intention : Garantir qu'une classe (10) n'a qu'une seule instance (16) et fournir un point d'accès de type global à cette classe (10).

Etant donné que la Fenêtre possède un seul Menu, une seule Palette de couleur, une seule Image, et une seule barre de statut, il nous parait logique de mettre ces classes (10) en Singleton.

Puis il est nécessaire d'utiliser le patron décorateur :

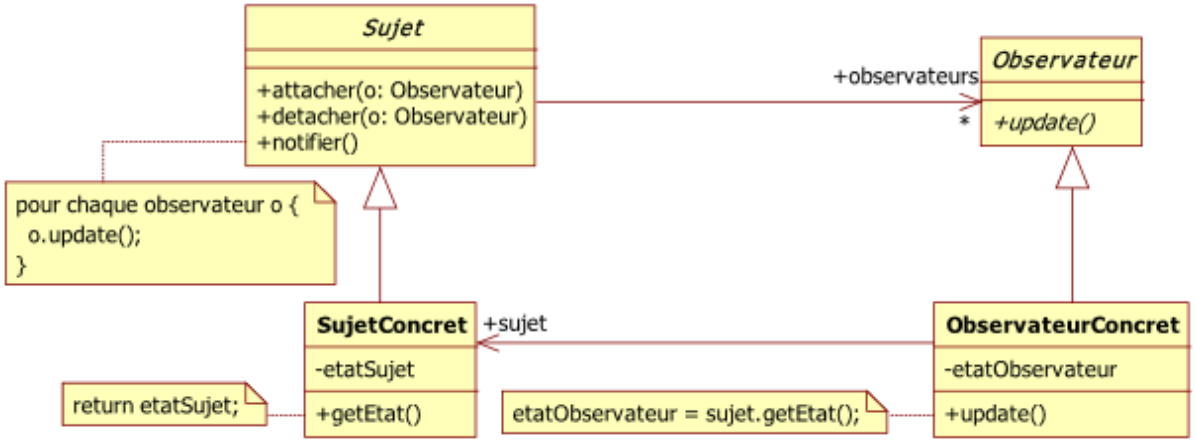


Intention : Attache dynamiquement des responsabilités supplémentaires à un objet. Les décorateurs fournissent une alternative souple à la dérivation, pour étendre les fonctionnalités.

Il est nécessaire d'utiliser ce patron pour utiliser les barres de défilements (Scroll Bar) au sein d'un panel (17) spécifique (JScrollPane).

Mais aussi pour décorer notre Explorateur d'image (Partie de Gauche) avec des images représentant les images du répertoire.

On utilise aussi le patron Observateur :

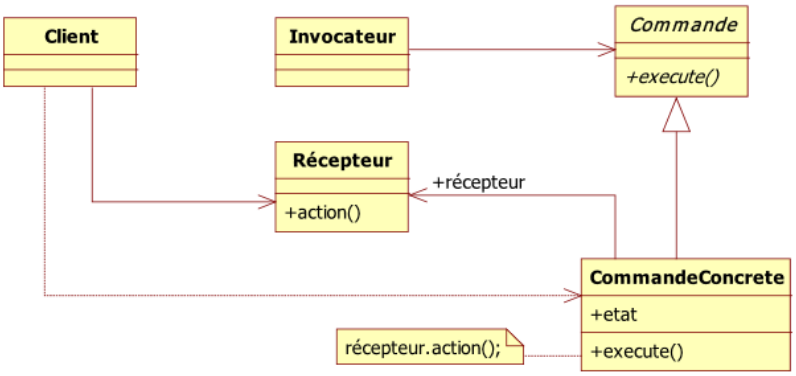


Intention : Définit une interdépendance de type un à plusieurs, de telle façon que quand un objet (11) change d'état, tous ceux qui en dépendent en soient notifiés et automatiquement mis à jour.

On utilise ce patron en premier lieu quand on utilise le zoom, la taille de l'image est modifiée et il n'est parfois plus nécessaire d'utiliser les barres de défilements donc elles sont supprimées. C'est-à-dire que dans ce cas-là, le sujet concret serait l'image (ou plutôt le panel (17) affichant l'image) et l'observateur concret serait la barre de défilement (implicitement se serait plutôt le panel (17) qui affiche les barres de défilements donc le JscollPane).

Mais aussi avec les dialogues qui auront besoins d'avoir des propriétés de l'image. Les dialogues seront les observateurs et le sujet sera l'image.

Et enfin, nous utilisons le patron Commande.



Intention : Encapsule une requête comme un objet, autorisant ainsi le paramétrage des clients par différentes requêtes, files d'attente et récapitulatifs de requêtes, et de plus, permettant la réversion des opérations.

Le patron commande va être utilisé pour implémenter le Annuler/Refaire (Undo/Redo). C'est-à-dire qu'on va mémoriser les fonctions qui ont été exécutés et annulés dans deux listes. Pour annuler, on enlève le dernier élément de la liste des commandes exécutées puis on ré exécute toutes les opérations.
Pour refaire, on ré exécute la fonction annulée.

Donc ici l'invocateur est Action Performed, c'est cette classe (10) qui exécute toutes les fonctions associées au boutons du menu, la méthode Action Performed est dans le contrôleur. La Classe abstraite (9) Commande, correspond à FonctionRetouche (classe (10) mère de toutes les fonctions). La commande concrète est la fonction qui a été exécutée par le bouton (fille de FonctionRetouche).

Le récepteur est l'objet (11) qui gère la liste de fonction (GestionAnnulerRefaire), il enlève ou modifie des éléments des listes.

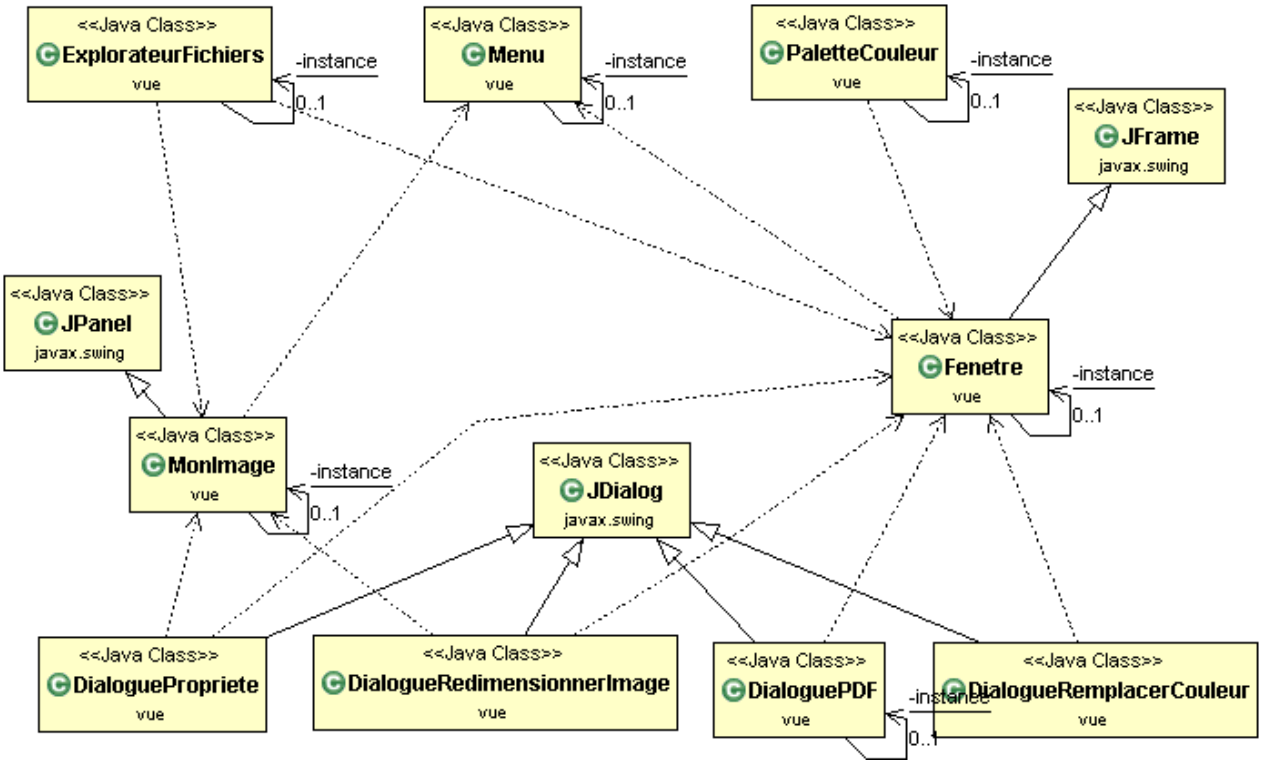
C.Ouverture des packages

A partir de l'analyse déjà réalisée, nous sommes en mesure d'y voir plus clair et de commencer à chercher les composants utiles pour le déroulement de notre application. Ainsi

que de comprendre comment chaque action va se dérouler et comment chaque classe (10) va se comporter quant à ces modifications.

Pour plus de lisibilité, nous ne mettons pas les attributs et les méthodes (voir annexe pour plus de détails). Mais nous mettons l'objet (11) parent de la classe (10) pour mieux comprendre ce que fait chaque classe (10).

Vue :



(Voir annexe figure 4)

Chaque élément est un élément visible de notre application. Ce sont tous des singletons sauf les dialogues Propriétés, Redimensionner Image et Remplacer Couleur car ils ne sont jamais identiques. En effet, DialoguePropriétés va afficher les propriétés de l'image en cours de traitement (nom, taille, date création, format) comme on est amené à changer d'image régulièrement au cours de l'utilisation du logiciel, il n'est pas nécessaire d'utiliser une instance (16) unique d'un objet (11) quand toutes les valeurs de ses attributs sont modifiées à chaque fois, autant recréer un nouvel objet.

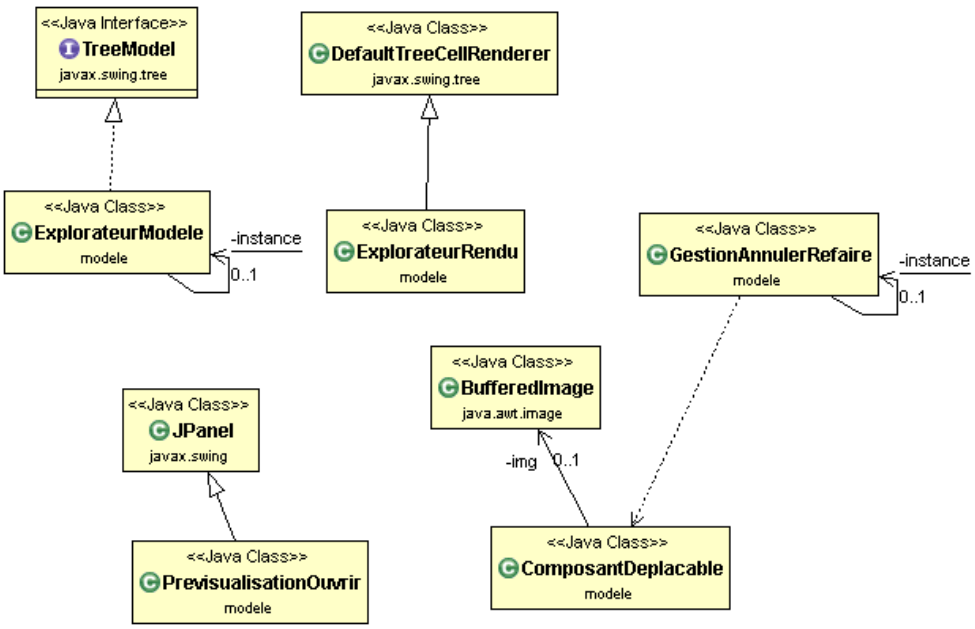
Même chose pour DialogueRedimensionnerImage (ce dialogue redimensionne l'image soit par pixels soit par pourcentages), car la taille de l'image est modifiée constamment avec le zoom et il en est de même pour DialogueRemplacerCouleur (remplace une couleur par une autre sur l'image) qui affiche la couleur actuellement sélectionnée or cette couleur est modifiable donc l'utilisation du Singleton n'est pas nécessaire.

Les patrons observateurs sont implicites car les sujets ou les observateurs ne sont pas directement des classes (10). Néanmoins on peut rajouter plusieurs utilisations de l'observateur car les dialogues Redimensionner Image et Propriétés observent les propriétés de l'image et se mettent à jour en fonction de celle-ci.

Dialogue PDF permet d'afficher un PDF en tant qu'image.

Contrairement à notre premier diagramme de classe (14), nous avons trouvés judicieux de ne pas mettre la barre de statut comme classe (10), car c'est un objet (11) simple qui trouve parfaitement sa place directement dans la Fenêtre.

Modèle :



(Voir annexe figure 5)

ExplorateurModèle est un singleton car il n'y a qu'une seule façon de faire un modèle d'explorateur d'image, il va établir l'ensemble des fichiers de l'arborescence et les communiqués à ExplorateurRendu par le biais de l'Explorateur Fichier.

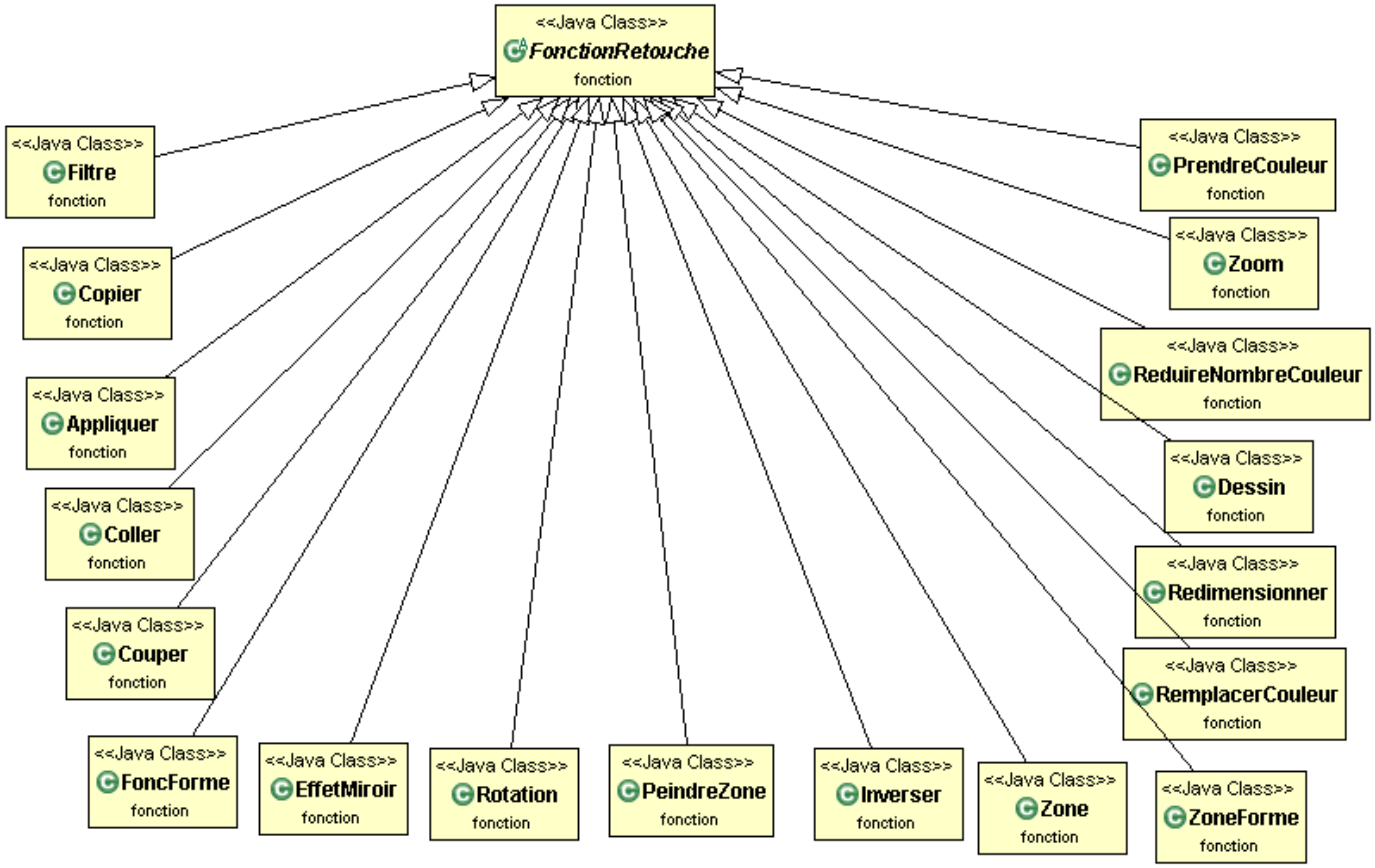
ExplorateurRendu permet d'afficher les images dans l'explorateur de Fichier.

Prévisualisation Ouvrir permet d'avoir quand on ouvre une image avec un JFileChooser une représentation de celle-ci avec quelques propriétés.

ComposantDéplaçable est la classe (10) qui gère le fonctionnement de la zone de sélection.

GestionAnnulerRefaire gère le fonctionnement des listes correspondant aux fonctions annulées et aux fonctions à refaire. C'est un des éléments du patron de Commande.

Fonction :



(Voir annexe figure 6)

On peut voir ici toutes les fonctions qui modifient l'image ou qui contribuent à la modifier.

FonctionRetouche est la classe (10) mère de toutes les autres fonctions, elle possède une méthode *exécuter( )* qu'on retrouve dans toutes les autres Fonctions.

Filtre va posséder toutes les méthodes qui permettent de flouter/dé flouter, éclaircir/assombrir et mettre en noir et blanc une image.

Couper/Coller/Copier/Appliquer et Zone vont servir pour la zone de sélection, Appliquer va fixer l'élément déplaçable au sein de l'image.

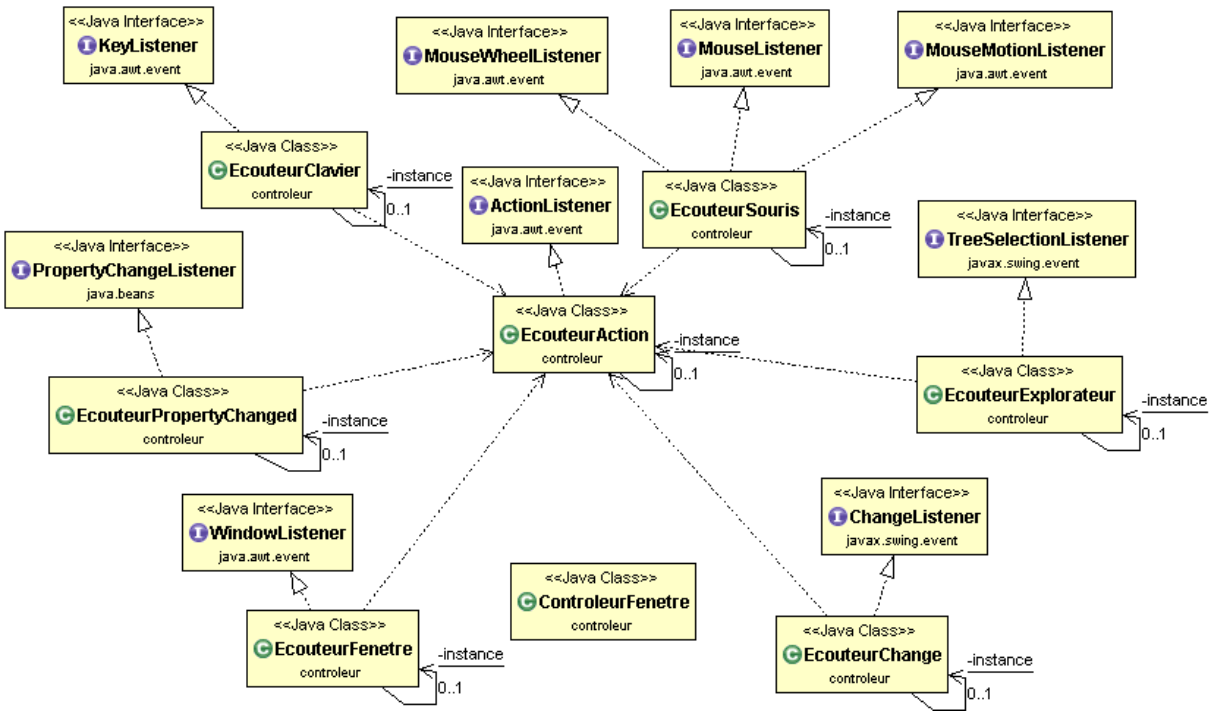
FoncForme et ZoneForm vont permettre de dessiner des formes sur une image.

EffetMioir, Rotation, PeindreZone, Inverser, Redimensionner, Zoom, RemplacerCouleur, Dessiner sont des fonctions classiques.

PrendreCouleur va prendre la couleur sur laquelle on clique.

RéduireNbCouleur va diminuer le nombre de couleur de l'image, c'est-à-dire diminuer d'un certain pourcentage l'entier représentant la couleur.

Contrôleur :



(Voir annexe figure 7)

Nous avons pris soin de diviser les Ecouteurs en fonction des principaux composants de la Vue. Ce qui nous donne sept classes (10) qui écoutent les composants de la Vue, la souris et le clavier pour exécuter les fonctions qui leur correspondent. Le ControleurFenetre contient juste la classe (10) Main qui lance l'application.

EcouteurClavier implémente KeyListener. Cette classe (10) gère si une touche est activée ou non. C'est à cette classe (10) qu'on associe la gestion des raccourcis clavier.

EcouteurSouris implémente MouseListener (qui gère tout ce qui est cliqué), MouseMotionListener (qui gère le déplacement de la souris), et MouseWheelListener (qui gère la molette).

EcouteurSouris doit être capable de faire un clic droit pour afficher un Popup, on doit connaître la position de la souris par rapport à l'image, on doit pouvoir dessiner avec la souris, et aussi zoomer.

EcouteurExplorateur implémente TreeSelectionListener, qui va permettre lors d'un clic sur un élément (image) de l'explorateur de fichier, d'ouvrir cette image en grand pour permettre de la modifier.

EcouteurChange implémente ChangeListener : classe (10) qui modifie la taille de l'image affichée dans le dialogue Redimensionner Image.

EcouteurPropertyChanged implémente PropertyChangedListener. Cette classe (10) va changer les propriétés à afficher dans le dialogue propriétés.

PropertyChangedListener et ChangeListener sont deux classes (10) très proches l'une de l'autre dans leur fonctionnement.

EcouteurFenetre implémente WindowsListener. Cet objet (11) gère la fermeture de la fenêtre. De sorte que si l'image n'est pas sauvegardée, on ne puisse pas quitter directement et l'écouteur permet aussi d'annuler la fermeture.

Enfin EcouteurAction implémente ActionListener. C'est dans cette classe (10) que sont exécutées toutes les fonctions associées aux boutons du menu. C'est l'écouteur principal sans lequel rien ne fonctionne.

Nous avons choisi de mettre tous ses écouteurs en singleton parce qu'il y a une et une seule façon d'exécuter les méthodes associées aux écouteurs. De plus les mettre en singleton permet un accès facile à ces objets partout dans le code.

D.Lien avec l'organisation

Nous avons tout de suite remarqué qu'il y avait quatre grandes parties à notre projet, c'est quatre parties correspondent aux quatre packages (Contrôleur, Vue, Modèle, Fonction). Cependant le package (13) Contrôleur allait se remplir au fur et à mesure de l'application

puisqu'il ne peut être codé qu'une fois que le composant qui lui correspond a lui aussi été créé. Comme le Modèle dépend de la Vue, il y a un lien entre ces packages donc on ne peut pas directement les séparer en créant l'un sans l'autre.

Ainsi nous nous sommes répartis les tâches d'après les deux autres parties.

Comme nous n'avions aucune idée de comment réaliser les fonctions correspondant à l'image, vu que nous n'avions jamais travaillé sur des images avant, ce sont Géraud et Thomas qui se sont occupés dans un premier temps de leur réalisation. L'autre partie des fonctions, plus orientées dessin, forme, zone de sélection ont été réalisé par Marc.

Enfin, tous ce qui est Vue a été confié à Jérémy, il était seul pour le package (13) Vue car nous avions déjà des bases dans ce domaine.

2. Les formats d'image

Il existe un grand nombre de formats d'images. Dans le cadre de notre projet nous nous sommes contentés de traiter les formats jpeg, png, gif et bmp.

Le format JPEG (Joint Photographic Experts Group) est un format de compression très efficace mais avec perte de qualité. Plus l'image est compressée, plus la qualité de l'image diminue. JPEG est particulièrement adapté et recommandé pour les photographies, il peut supporter 16,7 millions de couleurs et donne un bon rendu pour les images nuancées et les dégradés. Ce format offre un bon compromis entre la taille des images et le temps compression/décompression ce qui en fait un standard pour les sites web car l'affichage des images est rapide.

Le format PNG (Portable Network Graphics) utilise une compression sans perte de données. PNG peut supporter également 16,7 millions de couleurs. En revanche, la taille des fichiers peut vite devenir importante puisque ce format n'altère pas la qualité de l'image et le temps de compression/décompression est plus lent que le format JPEG.

Le format GIF est un format qui utilise une compression sans perte de qualité. Les images au format GIF peuvent contenir un maximum de 256 couleurs, ce qui rend ce format peu adapté pour les photographies. Il permet également de créer des animations : les GIFs animés.

Le format BMP est un des premiers formats d'image utilisé sous Windows. Cette technologie a pour principal avantage la qualité des images fournies : il n'y a pas de compression donc pas de perte de qualité. Mais sans compression, les fichiers sont de grande taille et impossible à afficher sur internet pour un utilisateur ayant une connexion bas débit. Par exemple, une image 800x600 pixels pèsera 1.37Mo.

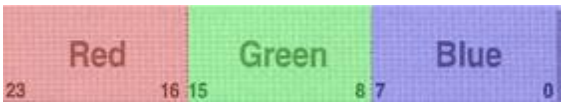
3. Les images en Java

En Java (1), il existe une classe abstraite (9) Image qui possède trois implémentations :

- BufferedImage : tableau de pixels stocké en mémoire
- VolatileImage : image stockée dans la carte graphique
- ToolkitImage : permet de charger des images en lecture seule

Pour la réalisation de ce projet nous avons utilisé des BufferedImage qui nous ont permis de récupérer directement les pixels d'une image dans un tableau et de pouvoir y travailler directement dessus. En effet chaque BufferedImage possède deux éléments :

- un Raster (trame en français) qui possède le tableau de pixels que l'on pourra modifié en utilisant un WritableRaster qui est tout simplement un Raster sur lequel la modification est autorisé
- un ColorModel qui définit la façon d'interpréter les images c'est-à-dire comment est gérer la valeur d'un pixel. Par exemple, un modèle RGB (Red Green Blue) saura interpréter chaque pixel en trois valeurs de données en rouge, vert et bleu. Un modèle de couleur en niveaux de gris saura interpréter une valeur de donnée unique comme un niveau de gris.



Ci-dessus, une représentation d'un pixel d'une image en RGB. Chaque couleur est représentée sur 8 bits soit une valeur variant de 0 à 255. Donc, la valeur d'un pixel est un ensemble des valeurs Rouge, Vert et Bleu. Par exemple, la valeur 0,0,255 va représenter du bleu, la valeur 255,0,255 du magenta, etc…

C'est donc grâce à ces deux éléments que nous avons pu réaliser une bonne partie des fonctions de retouche d'images de notre logiciel.

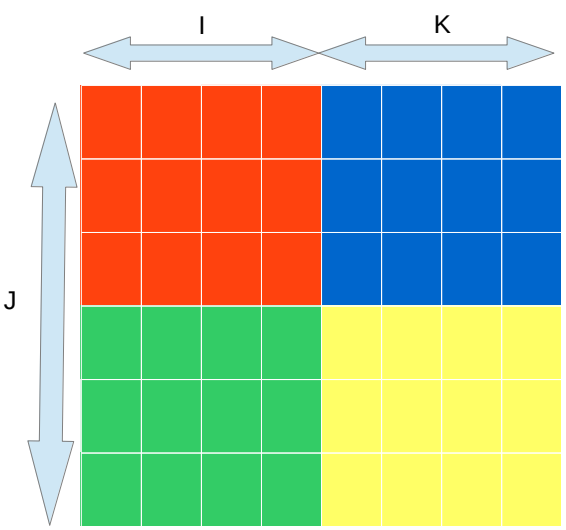
4. Quelques fonctions de JRetouch

A.Des échanges de pixels

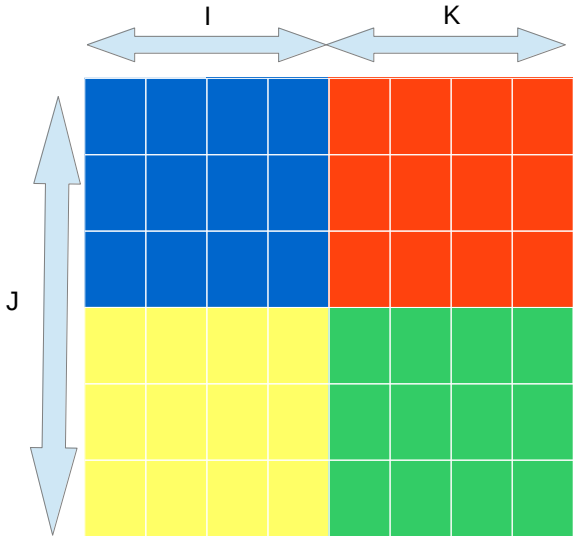
Plusieurs fonctions de JRetouch reposent sur de simples inversements de pixels, comme les fonctions d'effets miroir, de rotation et d'inversement. Nous allons maintenant voir quelques algorithmes pour réaliser ces fonctionnalités.

Algorithme pour inverser une image :

Pour J=0 ; J < hauteur de l'image ; J=J+1 =>parcours de l'image (hauteur)
  K=largeur de l'image
  Pour I=0 ; I < largeur de l'image/2 ; I=I+1 =>parcours moitié gauche
    On prend les pixels ayant pour coordonnées (I ; J) et (K ; J)
    On inverse ces éléments
  Fin Pour
Fin Pour



Avant avoir inversé l'image



Après avoir inversé l'image

Pour les effets miroir, le principe de l'algorithme est le même, il ne faut pas inverser les éléments mais seulement prendre les pixels de coordonnées (I ; J) et les recopier en (K ; J).

B.Les filtres

En Java (1), il existe une classe (10) nommée ConvolveOp qui permet d'appliquer une opération à chaque pixel de l'image en lui indiquant la façon dont il va être modifié tout en sachant que cette modification va s'effectuer en se basant sur le voisinage du pixel. ConvolveOp utilise un objet (11) Kernel qui contient la matrice et sa taille. C'est cette matrice qui va définir la modification à effectuer. Par exemple, pour flouter une image, la matrice sera de cette forme :

$$M=\begin{bmatrix} 1f/9, 1f/9, 1f/9, \\ 1f/9, 1f/9, 1f/9, \\ 1f/9, 1f/9, 1f/9 \end{bmatrix}$$

Il faut voir cette matrice de la façon suivante : la valeur du milieu (surlignée en jaune) représente le pixel de l'image à traiter et les autres valeurs représentent ses pixels voisins. Pour cet exemple, on va donc affecter à chaque pixel la moyenne de ses pixels voisins.

Il existe également d'autre moyen pour faire des filtres grâce à la classe (10) RescaleOp. Les opérations *rescale* modifient les valeurs des composantes RGB de chaque pixel : $c' = c*m+d$ Cette opération permet d'éclaircir ou d'assombrir une image, en gardant ou non le contraste.

- **0< m <1** : l'image est assombrie.
- **m > 1** : l'image est éclaircie.
- **0<= d <256** : d>0 l'image est éclaircie et le contraste est diminué.

  Ainsi en prenant m=0,8 on va assombrir l'image et avec m=1,2 on va l'éclaircir.

C.Les fonctions nécessitant un parcours de l'image

a.Réduire le nombre de couleur d'une image

Pour réduire le nombre de couleur d'une image, il suffit de parcourir entièrement l'image et de prendre la composante RGB de chaque pixel. Ensuite, nous faisons un ET logique entre la valeur récupérée du Rouge, Vert et Bleu avec la valeur 191 ce qui va recopier tous les bits de la valeur récupérée sauf le deuxième. Par exemple, 191 (1011 111) ET 234 (1110 1010) = 170 (1010 1010). Ainsi, nous remplaçons tous les pixels de l'image par sa valeur trouvée en faisant le ET logique et cela réduit le nombre de couleurs de l'image. Au début, la première fonction que nous avons réalisée consistait à prendre l'arrondi à la dizaine de chaque composante RGB mais le résultat obtenu n'était pas satisfaisant car l'opération effectuée n'était pas linéaire. Donc, la version avec le ET logique, étant linéaire, correspond exactement au résultat que nous souhaitions.

b.Inverser les couleurs d'une image

Le principe de cette fonction est le même que la fonction précédente sauf que nous réalisons un OU exclusif logique avec la valeur récupérée et la valeur 255 ce qui inverse tous les bits de la valeur récupérée. Par exemple, 255 (111 111) OU exclusif 234 (1110 1010) = 21 (0001 0101).

c.Remplacer une couleur par une autre

Pour réaliser le remplacement d'une couleur par une autre, il faut parcourir toute l'image et remplacer la couleur à remplacer par la nouvelle couleur. Vu qu'il existe un nombre très important de couleurs, il est nécessaire de réaliser un arrondi sur les couleurs (à remplacer et la nouvelle) pour avoir un résultat plus visible.

d.Peindre une zone de l'image

Pour cela, il faut récupérer l'endroit où l'utilisateur a cliqué pour peindre la zone et parcourir tous les pixels voisins de celui cliqué, et le peindre de la couleur voulue et cela réciproquement pour chaque pixel voisin jusqu'à ce que l'on trouve un changement de couleur qui indiquera que la fin de la zone est atteinte.

D. Exemple de fonction de dessin : les formes

Les classes (10) FoncForme et ZoneForme permettent de dessiner une ligne, un rectangle ou un ovale. Dessiner une forme se divise en trois étapes: le tracer, la modification, l'application. Tracer une forme se fait grâce à l'EcouteurSouris et sa fonction mouseDragged, qui va appeler paintComponent de MonImage. A chaque fois que celle-ci sera appelée, une instance (16) de ZoneForme sera créée puis exécutée, ce qui créera une forme.

Ensuite, si on lâche le clic de la souris, la fonction mouseReleased de l'EcouteurSouris va créer une instance (16) FoncForm ce qui va rendre la forme choisi redimensionnable. Cette forme est délimitée dans un rectangle. Deux points suffiront à construire la forme (celui en haut à gauche et celui en bas à droite)

Chaque fois que la forme sera redimensionner, le paintComponent de MonImage va appeler la fonction executer() de l'instance (16) de FoncForme, ce qui aura pour effet d'afficher la nouvelle forme redimmensionner (en modifiant les deux points).

Ensuite si l'option Appliqué du JPopMenu ou si l'on choisit une autre fonction de dessin, la forme sera appliquée à l'image grâce à la méthode executer() de la classe (10) Appliquer (qui appellera la méthode impression() de l'instance (16) de FoncForme)

5. Fenêtre

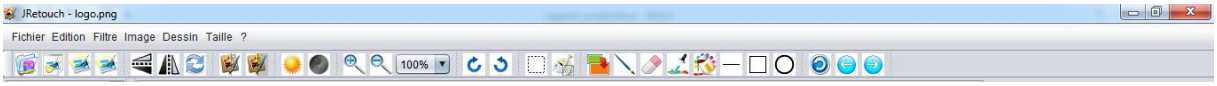
A.Barre de statut

La barre de statut est un élément simple du logiciel, elle permet de voir le nom de l'image qui est en train d'être retouché, de connaître la couleur qui a été sélectionnée depuis la palette de couleur mais aussi de savoir la position du curseur de la souris à l'intérieur de l'image.

Elle est composé majoritairement de JLabel qui est une classe (10) permettant d'afficher du texte, ce texte pouvant être modifié à chaque instant avec la méthode *setText (String NouveauTexte)* et la barre de statut possède aussi un bouton (JButton) représentant la couleur

sélectionnée, ce bouton n'a pas d'événement qui lui est associé c'est-à-dire qu'il ne sert à rien si ce n'est afficher la couleur sélectionnée.

B.Menu



Nous avons commencé dans la Vue par créer le menu. C'est un objet (11) simple à utiliser mais aussi à réaliser, néanmoins il est crucial à notre projet, puisque c'est de là que nous aurons accès à toutes les fonctions permettant de modifier l'image.

Il était nécessaire de le créer rapidement pour avoir un rendu de notre application et de nos fonctions de retouche d'images.

Nous avons donc étudié les classes (10) et les méthodes permettant d'obtenir un menu et de le décorer d'une manière plus esthétique.

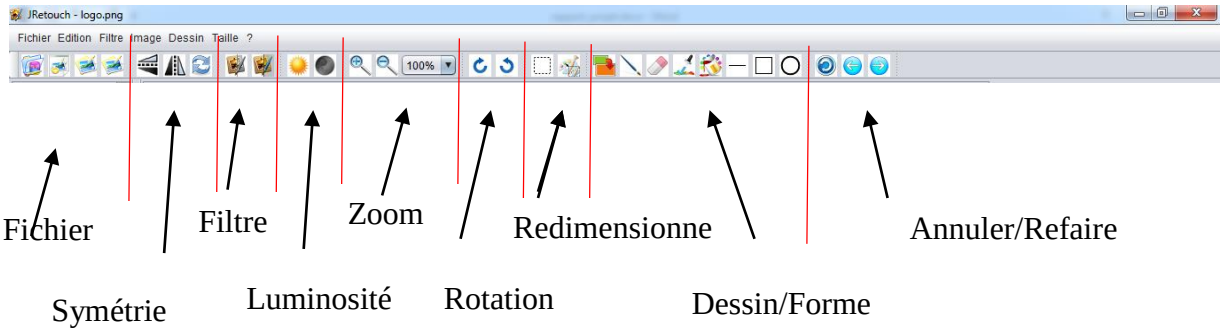
De cette étude ressort les composants suivant :

- JMenuBar : c'est un des composants père du menu. Il permet de créer la barre de menu auquel on associe des sous-menus.
- JToolBar : C'est l'autre composant père puisqu'il est placé sous la barre de menu. Il permet d'associer à une barre de menu des boutons ce que ne peux faire directement le JMenuBar. Cela revient à dire que c'est une barre d'icône.
- JMenu : C'est le fils de JMenuBar. Il permet de créer les sous-menus principaux, soit Fichier, Edition, Image, Filtre, Dessin…
- JMenuItem : Il est le fils d'un JMenu. C'est l'élément qui va remplir la liste déroulante du JMenu. On lui associe un évènement, lorsqu'on clique dessus l'écouteur (Contrôleur) qui lui fait référence (EcouteurAction) va pouvoir se déclencher et exécuter la fonction qui correspond à l'élément sur lequel on a cliqué.
- JButton : C'est l'élément qui permet de créer un icone dans la JToolBar (Barre d'icône). On lui associe un évènement, lorsqu'on clique dessus l'écouteur (Contrôleur) qui lui fait référence (EcouteurAction) va pouvoir se déclencher et exécuter la fonction qui correspond à l'élément sur lequel on a cliqué.

- ImageIcon : Il va permettre d'associer à un JButton ou à un JMenuItem une image.

Nous avons cherché comment améliorer ce menu en mettant par exemple des messages lorsqu'on positionne la souris sur un bouton ou un élément de sous-menu. Il suffit d'associer à ces éléments la méthode *setToolTipText (String texte)*. Ce qui permet dans savoir plus sur le bouton sur lequel nous allons cliquer.

Nous avons aussi ajouté des séparateurs dans la barre d'icône pour mettre ensemble les icônes qui avaient le même thème.



Il nous a été conseillé par la suite de charger le menu dynamiquement à l'ouverture du logiciel à l'aide de fichiers Properties (6), un pour la barre de menu et l'autre pour la barre d'icônes.

Pour la barre de menu, il suffit de saisir dans un fichier un ensemble de lignes clé/valeur correspondant pour la clé à l'ordre dans lequel il faut mettre l'élément dans le menu et pour la valeur, elle correspond au nom du JMenu puis au nom du JMenuItem et enfin le chemin de l'image le tout séparé par des points virgules ce qui donne :

#Clé=JMenu;JMenuItem;Chemin Image

Pour la barre d'icônes, il en est quasiment de même puisque les lignes correspondent à cela :

#Clé=nomBouton;Chemin Image; texte ToolTip

Ou alors

#Clé=séparateur

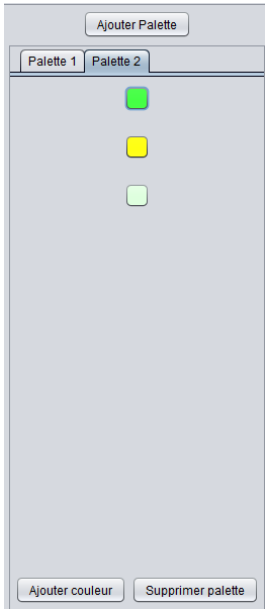
(Voir Annexe figures 8 et 9)

Il suffit ensuite de charger ces fichiers dans notre application pour cela on utilise en autre la fonction *load (InputStream (18) fichier).* On parcourt ce fichier dans une boucle pour permettre de lire toutes lignes et on récupère le contenu de ces lignes pour créer notre menu.

Si une ligne contient la valeur « séparateur » on ajoute un séparateur entre les icônes.

C.Palette de couleurs

La palette de couleurs est le deuxième élément de la Vue qui a été réalisé. Nous sommes parties de l'idée que la palette devait juste avoir un ensemble de bouton, ces boutons devant être colorés, et avoir plusieurs palettes pour permettre à l'utilisateur de hiérarchiser ces couleurs.



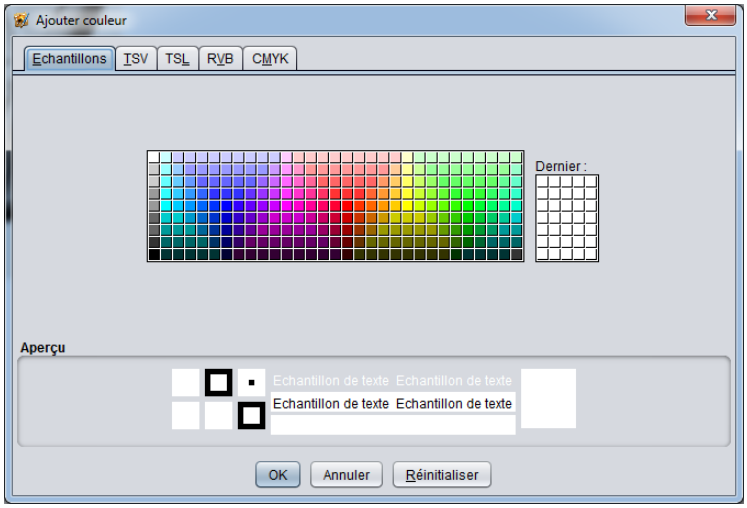
Pour concevoir cette palette nous avons utilisé différentes classes (10) :

- JTabbedPane : Cet objet (11) permet de créer des onglets, ces onglets nous serons utiles dans la mesure où chaque palette sera un nouvel onglet.
- JPanel : Il représente toute la partie interne de la palette c'est-à-dire celle qui contient les couleurs. Bien sûr comme un JPanel correspond à l'intérieur de l'onglet d'un JTabbedPane, il faudra autant de JPanel qu'on a d'onglets donc nous avons choisi de faire une liste de JPanel (ArrayList<JPanel> arrayList étant une liste en Java (1)).
- JButton : C'est l'élément qui va nous permettre d'afficher les couleurs dans le JPanel. En effet à chaque JButton donc à chaque bouton, on peut associer une

couleur. Quand on clique sur un de ces boutons, la couleur sélectionné de la barre de statut va en être modifiée.

- JColorChooser : Un dialogue Java (1) déjà réalisé, qui permet de sélectionner une couleur

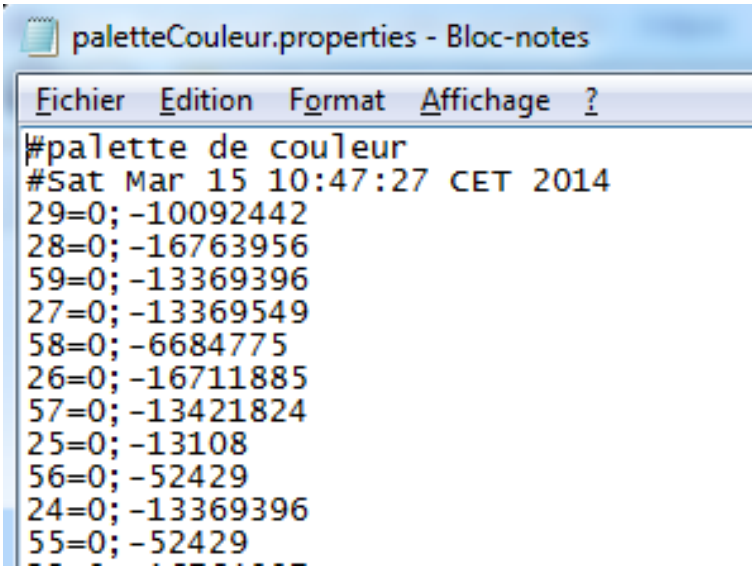
Nous récupérons la couleur sélectionnée pour créer un bouton avec cette couleur.



Comme pour le menu, il nous a été conseillé de créer cette palette de couleur dynamiquement avec un fichier de propriétés (6). Cependant cette fois-ci, nous allons gérer ce fichier de propriété (6) en lecture et en écriture. De sorte que lorsqu'on quitte le programme la palette et ses couleurs soient sauvegardées.

Dans ce fichier de propriétés (6), il y a un ensemble de clé/valeur, la clé représente l'indice de la couleur dans la liste, et la valeur est le numéro de la palette suivi d'un entier représentant la couleur RGB du bouton de la palette.

Voici un exemple du fichier :



Pour la lecture, il suffit ensuite de charger le fichier avec la méthode *load* pour récupérer chaque couple de clé/valeur qui permet de créer un bouton avec chaque couleur du fichier.
Chaque bouton est ensuite inséré dans la liste de panel (17) qui correspond au numéro de la palette.
Pour l'écriture, il suffit de parcourir la liste de boutons et d'insérer dans le fichier chaque valeur RGB (que l'on récupère grâce à la méthode *getRGB ( )*) de chaque bouton en fonction de l'indice de la palette.

D.Explorateur de fichiers

Cette étape fut la plus nouvelle pour nous pour ce qui est de la Vue. En effet, nous avons réalisé une prévisualisation des images du répertoire courant. Avec possibilité de changer de répertoire et/ou de couleur de fond avec l'onglet +.
On change la couleur de fond dans le cas où comme ici nous avons une image blanche sur un fond blanc.

Pour réaliser cet explorateur de fichier nous utilisons :

- JTree : Il donne une structure d'arbre à l'explorateur de fichier, chaque image ou repertoire est un nœud de l'arbre.
- JTabbedPane : C'est l'élément qui permet d'avoir des onglets, ici nous en avons deux un pour afficher les images, l'autre qui contient les boutons changer racine et changer couleur
- JScrollPane : Panel (17) qui permet d'avoir la barre de défilement.
- TreeModel : Elément qui calcule et trouve tous les éléments de l'arbre c'est-à-dire que le modèle donne au JTree tous les fichiers du répertoire.
- DefaultTreeCellRenderer : Cet élément va traiter tous les éléments du JTree et va transformer en image tous les fichiers dont l'extension (4) correspond à une image.
- Toolkit (19) : il permet de faire une miniature d'image pour que ce soit moins consommateur en mémoire grâce à :
  Image = toolkit.getImage (Chemin du fichier)
- JFileChooser : c'est l'élément qui change le répertoire de l'explorateur. Il affiche tous les fichiers du Disque. Seulement nous ne désirons pas récupérer un fichier mais un répertoire donc il suffit de modifier un des attributs du JFileChooser avec la méthode
  SetFileSelectionMode (JFileChooser.DIRECTORY_ONLY).

E.Amélioration de la vue

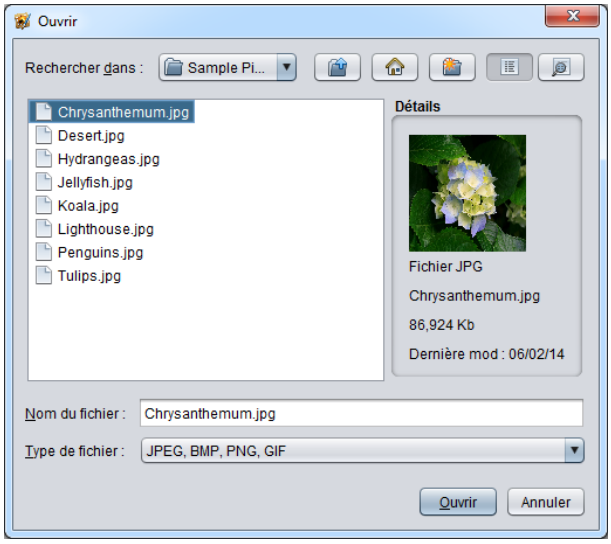
a.Design de l'application

Le Look And Feel (20) de base en Java (1) n'est pas très jolie. (Voir annexe figure 9) C'est pourquoi nous avons décidé de chercher, un Look And Feel (20) capable de fonctionner sur toutes les plateformes mais aussi de rendre notre logiciel plus professionnel. Nous avons testé Nimbus qui est le design que nous avons choisi, il est simple, propre et respecte les critères cités.

b.Prévisualisation JFileChooser

Le JFileChooser est un objet (11) tout prêt de Java (1), il permet de visualiser l'ensemble des fichiers du disque. Néanmoins on ne peut voir que le nom des fichiers et pas directement l'image. C'est pourquoi nous avons eu l'idée de créer une zone de prévisualisation à l'intérieur du JFileChooser pour avoir quelques propriétés de l'image mais aussi avoir un aperçu de celle-ci. (Zone Détails)

On relie le JFileChooser et le panel (17) de prévisualisation avec la méthode *setAccessory (JPanel panelPrévisualisation).*



III. Résultats

1. Ce qui a été réalisé

A.Le Blog

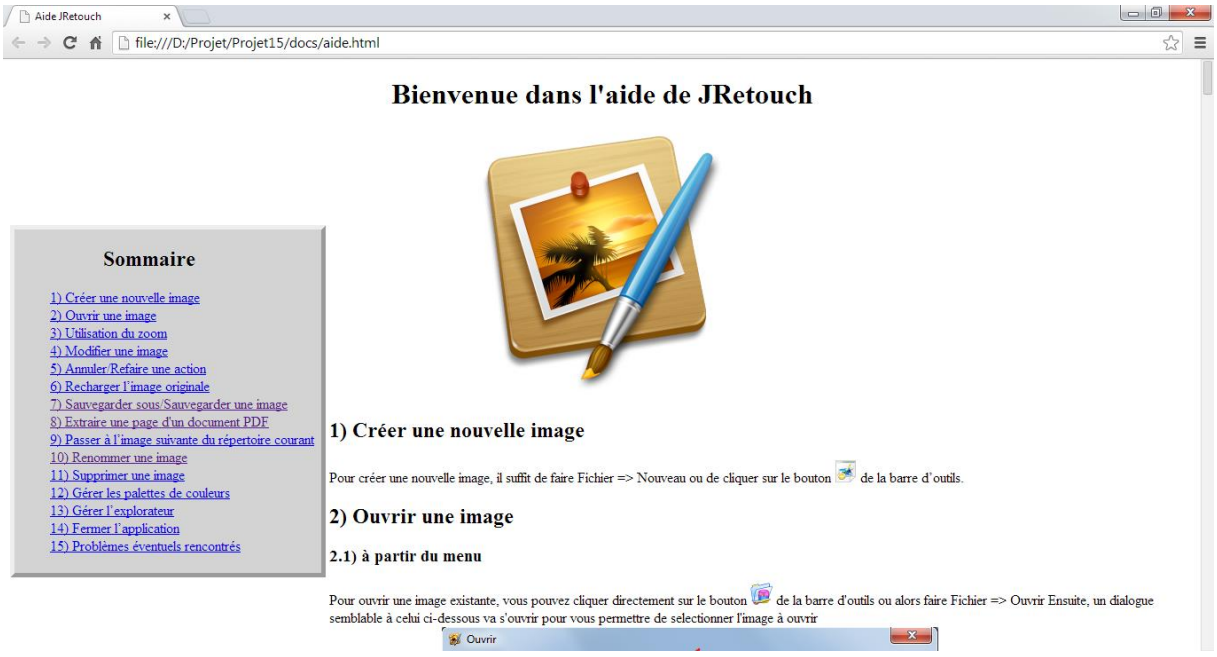
Nous avons créé et mis à jour chaque semaine un blog de type Blog Spot. Il a été créé suite à la demande de M.SPEH.

Il nous a permis de communiquer notre projet au monde mais aussi de nous permettre de voir l'avancement du projet au fur et à mesure du temps. Chaque semaine ou presque nous mettions un jar (21) du projet téléchargeable ainsi qu'un commentaire des améliorations apportés à notre projet.
Ainsi notre tuteur pouvait laisser des commentaires et consulter le code de notre application.

Lien du blog : http://projetiutretoucheurimages.blogspot.fr/

B.Le manuel Utilisateur

Nous avons réalisé un manuel utilisateur (aide) consultable depuis l'application avec le menu ? → Aide ou avec F1. Il permet de découvrir et de comprendre notre application, puisque chaque fonction est décrite et il explique comment utiliser cette fonction. (Voir plan du manuel utilisateur en annexe figure 11)



C.Notre application

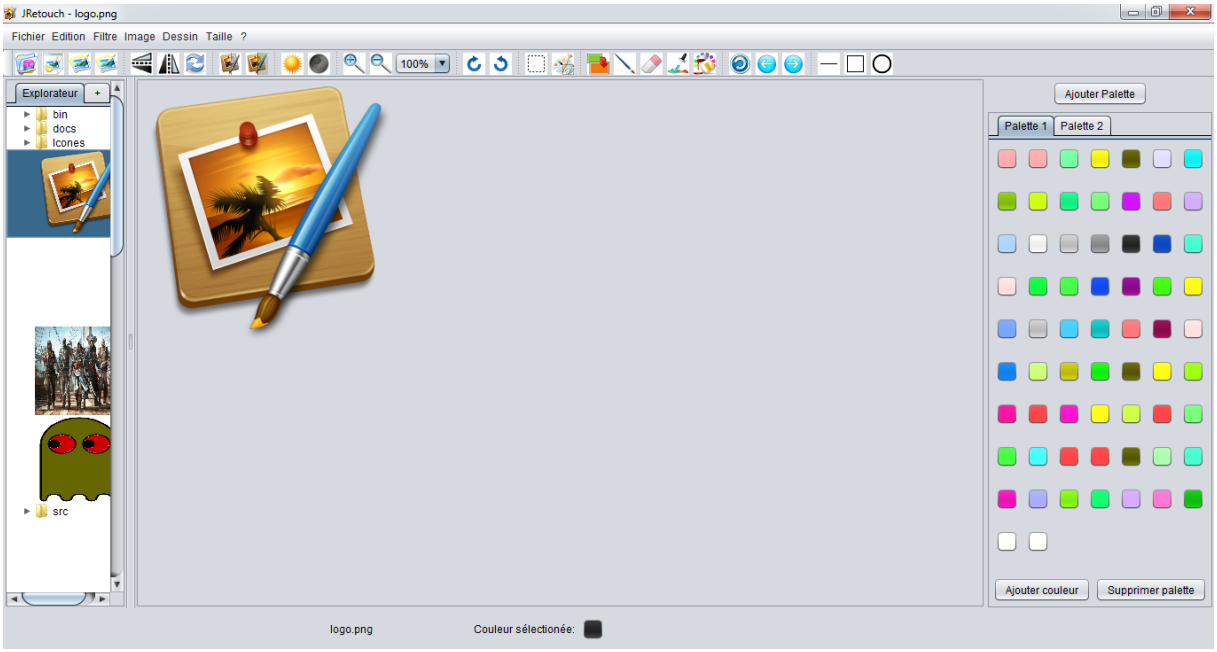
Nous avons réalisé une application respectant le cahier des charges, c'est-à-dire une application possédant un explorateur de fichier avec la visualisation de toutes les images de ce répertoire et la possibilité de changer la racine du répertoire donc de changer le répertoire d'affichage des images.

Notre application possède une palette de couleur qui a la possibilité d'avoir plusieurs palettes (une infinité théoriquement), chaque palette ayant une capacité maximale de cent cinq couleurs.

On retrouve aussi une barre de statut qui permet de connaitre le nom de l'image en cours de traitement, la couleur sélectionnée par l'utilisateur (noir par défaut) et la position de la souris sur l'image.

JRetouch possède une zone de visualisation de l'image, où l'on voit l'image et les modifications apportées par les fonctions.

Enfin un menu possédant une barre de sous menu (Fichier, Edition, Filtre, Image, Dessin, Taille et ?) qui permet d'exécuter toutes les fonctions et une barre avec des icônes qui permet d'exécuter les fonctions les plus courantes.



En soit toutes les fonctions ont été réalisés sauf le fait de lister les couleurs d'une image par fréquence qui était jugé comme moins prioritaire.

(Voir liste des fonctions en annexe figure 12)

Mais nous avons rajouté quelques fonctionnalités, en effet le dialogue Propriétés n'était pas prévu initialement ainsi que la prévisualisation de l'image dans le navigateur (Ouvrir Image), les raccourcis clavier et le Annuler/Refaire. Nous trouvions que cela apporter un réel plus à l'application, c'est pourquoi nous avons préféré faire ces fonctionnalités.

2. Les manques du projet

Notre projet ne possédait pas de chef. C'est-à-dire personne ne dictait à l'autre ce qu'il devait faire. Nous respections chacun nos tâches sans être suivi par quelqu'un. Nous avons choisi de ne pas avoir de chef car nous sommes quatre bons amis et nous ne voulions pas qu'il y est conflit. Néanmoins à certains moments, il aurait fallu que quelqu'un nous pousse. Nous avons peut-être perdu de la vigueur comparée à un projet dirigé. Cependant notre tuteur a eu aussi ce rôle et qu'il a su jouer.

Il y avait aussi un manque d'organisation, peut-être lié au manque de leader. En effet, durant les premières semaines du projet, les tâches n'étaient pas entièrement définies et tout le monde faisait ce que lui plaisait au sein du projet. Cependant une fois la routine de travail installée et les tâches planifiées, chacun a su quoi faire et le projet a pu se dérouler normalement.

De plus, il y avait un manque de communication avec le tuteur. En effet durant les deux premiers mois, nous préférions comprendre Swing (22) et le Java (1) par nous mène car c'était pour nous un nouveau langage. C'est-à-dire que l'on démarrait presque de zéro et nous ne voulions pas déranger le tuteur avec des problèmes futiles ou que nous arrivions à résoudre par nous-même. Puis nous nous sommes aperçus qu'il pouvait nous donner une autre façon de voir et comprendre les choses alors nous avons pallié à ce manque.

Puis, il aurait été intéressant d'utiliser SonarQube (23), car ce logiciel nous aurait permis d'avoir un meilleur code.

Enfin, nous aurions pu utiliser un logiciel de gestion de version (24). Les personnes du groupe modifient le code du projet chacun de leur côté, cet outils nous aurait permis de rassembler toutes les modifications dans une seule version. (Exemple : GIT)

3. Difficultés rencontrés

Il nous a été difficile de trouver les bons composants à utiliser pour notre application. En effet, Swing (22) et la Javadoc (7) ont un large éventail de composants. Et comme c'était une nouvelle façon de programmer pour nous au départ, cela a pu nous troubler ou tout au moins nous faire perdre du temps à chercher sur internet ou sur la Javadoc (7). Nous sommes conscients que l'apprentissage passe par la recherche et la découverte néanmoins, il nous est aujourd'hui

beaucoup plus facile de coder qu'au début car notre connaissance de Swing (22) n'en est que meilleure.

Très vite nous nous sommes confrontés à des problèmes de mémoire. En effet, notre application était vraiment gourmande à telle point qu'elle prenait toute la mémoire qui est initialement alloué à Java (1). D'une part à cause de l'explorateur de fichier et d'autre part à cause de beaucoup de méthode repaint (25) inappropriés.

Une fois que la source des problèmes a été identifiée, il était plus simple de les résoudre. Nous avons remplacé les images de l'explorateur d'images par des miniatures grâce à l'objet (11) Toolkit (19) (boîte à outils).

Quant aux méthodes repaint, il suffisait simplement d'en enlever certaines ou sinon de faire des repaint (25) plus spécifiques, c'est-à-dire au lieu de recharger la fenêtre en entier, il fallait juste recharger le composant qui a été modifié.

Nous avons aussi eu des problèmes avec le zoom et la zone de sélection. En effet, quand on coupait une partie de l'image, qu'on zoomer ou dé zoomer et qu'on appliquait pour coller la zone de l'image, il y avait des problèmes de proportion. En fait, la zone de l'image coupée n'avait pas été zoomée. La réponse à ce problème nous a été fournie par M. SPEH. Qui nous a proposé d'ajouté une variable facteurZoom.

4. Prolongements

Notre application est une application fonctionnelle mais cependant elle est incomplète par rapport à d'autres outils de retouche d'image comme par exemple Photoshop.

Nous pourrions lui rajouter de nombreuses fonctionnalités en plus :

-Insérer du texte sur l'image avec toutes les fonctionnalités associées au texte (centrer, taille, police…)

-Dessiner d'autres formes que lignes, rectangles et cercles

- copier/coller à partir d'une image d'internet ou d'un autre logiciel

- Apporter des fonctions de dessin plus précises

- Ajouter une fonction de zoom qui permet d'adapter une image d'une taille importante à la taille de l'écran

Conclusion

Tout au long de ces semaines de projet, nous sommes arrivés à développer une application fonctionnelle respectant le cahier des charges. Nous avons apprécié réaliser cette application qui nous a permis de découvrir le fonctionnement d'un logiciel de retouche d'images.

Ce projet nous a donné un avant-goût du développement d'une application réalisé en entreprise. Il nous a aussi apporté de nombreuses connaissances techniques mais aussi organisationnelles. En effet, nous avons pu conforter nos compétences en Java (1) et nous confronter à la réalisation d'un projet en équipe, ce qui nous a permis de consolider notre aptitude à travailler en équipe.

Même si nous avons connu des difficultés dans la réalisation de ce projet, nous mettrons en avant le savoir-faire accumulé et nous retiendrons les erreurs commises pour ne pas les reproduire par la suite.

Résumé en Anglais

For our second year of university diploma in Computer Science, we had to do a supervised project which duration is fourteen weeks. To do it, we worked in a little team of four students: Thomas JANOWIEZ, Marc PESCHER, Jérémy TALABARD and Géraud MÉTAIS.

The aim of our project was to develop in Java (1) programming language a little software which can be able to edit pictures and which looks like to Adobe Photoshop or Paint from Windows. The project's supervisor was Mr. Sylvain SPEH.

During the fourteen weeks, we separate the work between us. Jérémy was in charge to make the window of JRetouch with the toolbar, the menu, the color palette and the explorer. Marc did the copy/cut/paste functions and made all the things to draw directly on a picture. And Thomas and Géraud develop the editing functions.

We improved our knowledge in Java (1) programming language and we learnt a lot regarding the way like Java manages pictures. Moreover, we improved our teamwork skills in order to develop an application, which will be used in our future jobs.

At the end of the project, we manage to carry out a working software. With it, we can do the basis things that do an editing picture software like transforming a picture in black and white, resizing a picture, drawing directly on a picture for example.

Bibliographie / Webographie

http://docs.oracle.com/javase/7/docs/api/

Fonctions de retouche :

http://www.liafa.univ-paris-diderot.fr/~carton/Enseignement/InterfacesGraphiques/MasterInfo/Cours/Swing/images.html

http://slim-boukettaya.developpez.com/tutoriels/traitement-images-java/#LIV-B

http://java.developpez.com/faq/gui/?page=graphique_general_images

http://imss-www.upmf-grenoble.fr/prevert/Prog/Java/swing/image.html

http://www-igm.univ-mlv.fr/~paumier/IG/C12.pdf

http://programmationjava.1sur1.com/Java/Tutoriels/AWT/TraitementImages.php

http://www.java2s.com/Tutorial/Java/0261__2D-Graphics/CreatingaImageZoomerusingGraphics2D.htm

Interface/Fenêtre :

http://www.developpez.net/forums/d514624/java/interfaces-graphiques-java/awt-swing/composants/arbres/jtree-debutant-parcourir-l-arborescence-disque/

http://www.developpez.net/forums/d711241/java/interfaces-graphiques-java/awt-swing/composants/arbres/definir-image-arriere-plan-d-jtree/

http://fr.openclassrooms.com/informatique/cours/apprenez-a-programmer-en-java/les-menus-et-boites-de-dialogue

http://www.jmdoudoux.fr/java/dej/chap-swing.html

http://docs.oracle.com/javase/tutorial/uiswing/components/tabbedpane.html

http://www.jmdoudoux.fr/java/dej/chap-swing.html

http://baptiste-wicht.developpez.com/tutoriels/java/swing/debutant/

http://java.developpez.com/faq/gui/?page=awtSwingJTableJTree

http://stackoverflow.com/questions/2199987/changing-the-node-image-of-a-jtree-dynamically

http://docs.oracle.com/javase/7/docs/api/java/awt/Toolkit.html

http://www.developpez.net/forums/d475738/java/interfaces-graphiques-java/awt-swing/composants/arbres/jtree-afficher-image/

http://www.developpez.net/forums/d140064/java/interfaces-graphiques-java/awt-swing/swing-meilleurs-look-and-feel-java/

Création d'exécutables JAVA :

http://baptiste-wicht.developpez.com/tutoriels/java/outils/executables/#LIII-A

http://devwizard.free.fr/html/fr/JavaExe.html

Fichier de properties :

http://patatos.over-blog.com/article-lire-un-fichier-de-proprietes-en-java-47382363.html

http://www.mkyong.com/java/java-properties-file-examples/

http://stackoverflow.com/questions/1318347/how-to-use-java-property-files

http://docs.oracle.com/javase/tutorial/essential/environment/properties.html

http://yoann.thomann.free.fr/index.php?article=15

http://www.mkyong.com/java/java-properties-file-examples/

Fonction de Dessin :

http://stackoverflow.com/questions/874360/swing-creating-a-draggable-component

http://java2s.com/Tutorials/Java/java.awt/Desktop/Java_Desktop_open_File_file_.htm

http://www.java2s.com/Code/Java/2D-Graphics-GUI/Thisprogramdemonstratesthevarious2Dshapes.htm

http://stackoverflow.com/questions/11006496/select-an-area-to-capture-using-the-mouse

http://www.coderanch.com/t/344873/GUI/java/draw-straight-line-curved-line

Réalisation de l'aide :

http://www.w3.org/Style/Examples/007/menus.fr.html#FAQ

Lexique

Java (1)

Le langage Java est un langage de programmation informatique orienté objet créé par James Gosling et Patrick Naughton. La particularité de Java est que les logiciels écrits dans ce langage doivent être très facilement portables sur plusieurs systèmes d'exploitation tels que UNIX, Windows, Mac OS ou GNU/Linux, avec peu ou pas de modifications.

Multiplateforme (2)

Le fait qu'une application soit multiplateforme signifie qu'elle va pouvoir être installée sur n'importe quel système d'exploitation et fonctionner correctement pour tous ceux-ci.

Compilateur (3)

Un compilateur est un programme informatique qui transforme un code source écrit dans un langage de programmation (le langage source) en un autre langage informatique (le langage cible).

Extension (4)

Une extension de fichier est un suffixe ajouté au nom d'un fichier pour identifier son format.

Machine virtuelle (5)

Une machine virtuelle est une illusion d'un appareil informatique créée par un logiciel. Le logiciel simule la présence de ressources matérielles et logicielles permettant d'exécuter des programmes dans les mêmes conditions que celles de la machine simulée. Un des intérêts des machines virtuelles est de pouvoir s'abstraire des caractéristiques de la machine physique utilisée permettant une forte portabilité des logiciels.

Fichier Propriétés ou Properties (6)

Extension de fichier utilisée en Java et qui permet aux technologies compatibles de stocker les paramètres de configuration d'un logiciel. Ils sont également utilisés afin de stocker les chaînes de caractères standardisées.

Javadoc : (7)

C'est un outil qui permet, en inspectant le code Java des classes (10), de produire une documentation très complète du code en une page web. Cette documentation permettra, entre autre, une reprise simplifiée d'une application par un autre développeur.

Diagramme de Gantt (8)

Le diagramme de Gantt est un outil utilisé en ordonnancement et en gestion de projet, permettant de visualiser dans le temps les diverses tâches composant un projet. Il s'agit d'une représentation d'un graphe connexe, valuée et orientée, qui permet de représenter graphiquement l'avancement du projet.

Classe abstraite (9)

En programmation orientée objet, une classe abstraite (9) est une classe (10) dont l'implémentation n'est pas complète et qui n'est pas instanciable. Elle sert de base à d'autres classes dérivées (héritage).

Classe (10)

Une classe (10) décrit les responsabilités, le comportement et le type d'un ensemble d'objets. Les éléments de cet ensemble sont les instances (16) de la classe (10).

Une classe (10) est un ensemble de fonctions et de données (attributs) qui sont liées ensemble par un champ sémantique. Les classes (10) sont utilisées dans la programmation orientée objet. Elles permettent de modéliser un programme et ainsi de découper une tâche complexe en plusieurs petits travaux simples.

Objet (11)

Un objet est un conteneur symbolique, qui possède sa propre existence et incorpore des informations et des mécanismes en rapport avec une chose tangible du monde réel, et manipulés dans un programme. C'est le concept central de la programmation orientée objet.

MVC (Modèle-Vue-Contrôleur) (12)

C'est un modèle destiné à répondre aux besoins des applications interactives en séparant les problématiques liées aux différents composants au sein de leur architecture respective. Ce qui correspond tout à fait à notre type d'application et à la majeure partie des applications.

Package (Paquet) (13)

C’est un ensemble de fichiers informatiques contenant les informations et procédures nécessaires.

Diagramme de classe (14)

Le diagramme de classes est un schéma utilisé en génie logiciel pour présenter les classes et les interfaces des systèmes ainsi que les différentes relations entre celles-ci. Ce diagramme fait partie de la partie statique d'UML (26) car il fait abstraction des aspects temporels et dynamiques.

Patron de conception (15)

Un patron de conception est la meilleure solution connue à un problème de conception récurrente.

Instance (16)

En programmation orientée objet, on appelle instance d'une classe, un objet avec un comportement et un état, tous deux définis par la classe. Il s'agit donc d'un objet constituant un exemplaire de la classe.

Panel (17)

Un panel désigne une unité graphique rectangulaire sur un écran, exploitée pour représenter un ensemble logique de données.

InputStream (18)

InputStream ou Flux d'entrée représente le fichier qui est chargé dans une application.

Toolkit (19)

Classe servant de boite à outils avec de nouvelle méthode qui ne sont pas dans Swing standard

Look and Feel (20)

Le look and feel est un ensemble de règles qui régissent la présentation visuelle ainsi que le comportement des interfaces graphiques. Les règles de présentation concernent en particulier l'usage des couleurs, la typographie, la présentation et la signification des logos et des icônes et la présentation des fenêtres.

Jar (21)

Un fichier JAR est un fichier compressé utilisé pour distribuer un ensemble de classes Java. Ce format est utilisé pour stocker les définitions des classes, ainsi que des métadonnées, constituant l'ensemble d'un programme. Il est exécutable.

Swing (22)

Swing est une bibliothèque graphique pour le langage de programmation Java, inclus dans J2SE. Swing offre la possibilité de créer des interfaces graphiques identiques quel que soit le système d'exploitation sous-jacent.

SonarQube (23)

Logiciel permettant de mesurer la qualité du code source en continu.

Logiciel de version (24)

Permet de stocker un ensemble de fichiers en conservant la chronologie de toutes les modifications qui ont été effectuées dessus. Il permet notamment de retrouver les différentes versions d'un lot de fichiers connexes.

Repaint (25)

C’est une méthode qui recharge un composant graphique. Elle est utile quand un composant est modifié et qu’il faut le réafficher.

UML (26)

Langage de modélisation unifié, est un langage de modélisation graphique à base de pictogrammes. Il est utilisé en développement logiciel, et en conception orientée objet. UML est couramment utilisé dans les projets logiciels.

Annexes

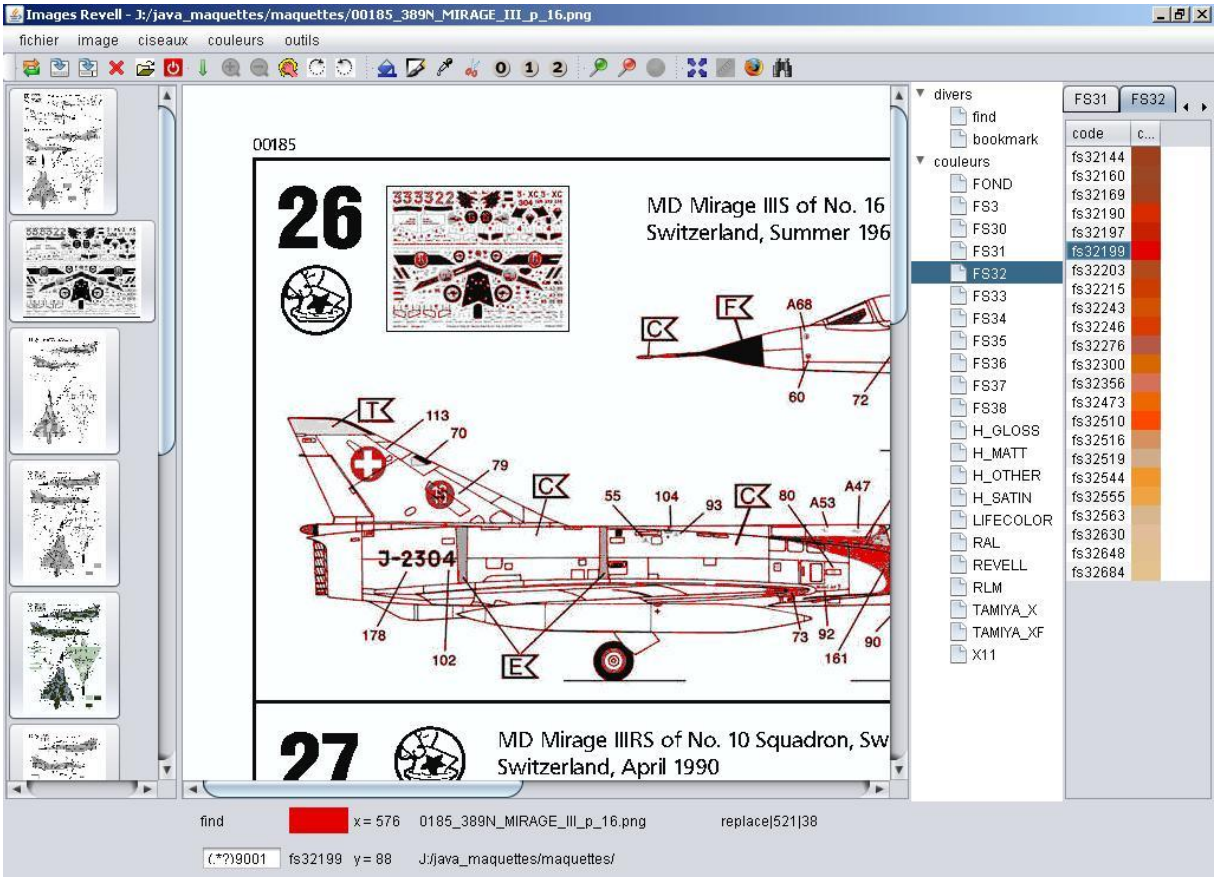
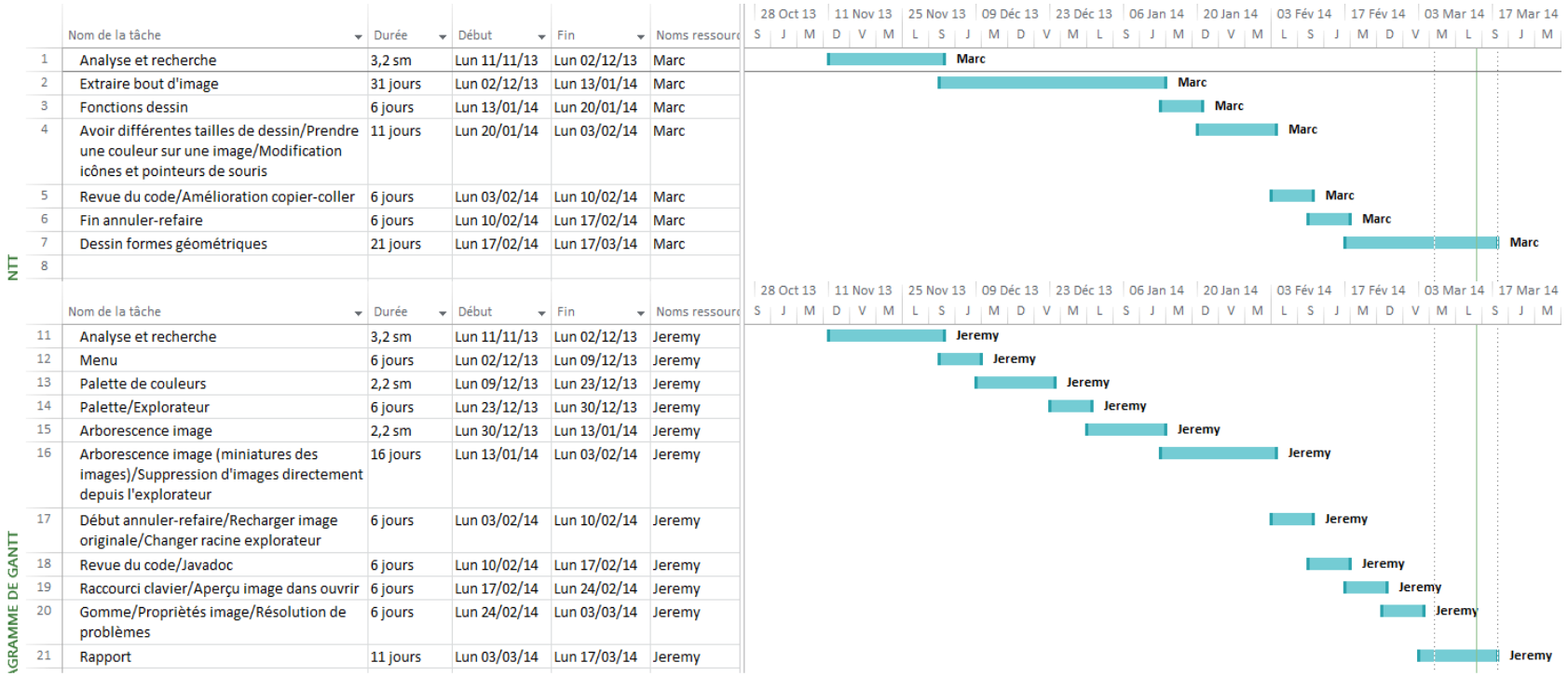


Figure 1 : application déjà réalisée par notre tuteur, la forme de la fenêtre de notre projet doit s'en approcher



	Nom de la tâche	Durée	Début	Fin	Noms ressourc
1	Analyse et recherche	3,2 sm	Lun 11/11/13	Lun 02/12/13	Marc
2	Extraire bout d'image	31 jours	Lun 02/12/13	Lun 13/01/14	Marc
3	Fonctions dessin	6 jours	Lun 13/01/14	Lun 20/01/14	Marc
4	Avoir différentes tailles de dessin/Prendre une couleur sur une image/Modification icônes et pointeurs de souris	11 jours	Lun 20/01/14	Lun 03/02/14	Marc
5	Revue du code/Amélioration copier-coller	6 jours	Lun 03/02/14	Lun 10/02/14	Marc
6	Fin annuler-refaire	6 jours	Lun 10/02/14	Lun 17/02/14	Marc
7	Dessin formes géométriques	21 jours	Lun 17/02/14	Lun 17/03/14	Marc
8					

	Nom de la tâche	Durée	Début	Fin	Noms ressourc
11	Analyse et recherche	3,2 sm	Lun 11/11/13	Lun 02/12/13	Jeremy
12	Menu	6 jours	Lun 02/12/13	Lun 09/12/13	Jeremy
13	Palette de couleurs	2,2 sm	Lun 09/12/13	Lun 23/12/13	Jeremy
14	Palette/Explorateur	6 jours	Lun 23/12/13	Lun 30/12/13	Jeremy
15	Arborescence image	2,2 sm	Lun 30/12/13	Lun 13/01/14	Jeremy
16	Arborescence image (miniatures des images)/Suppression d'images directement depuis l'explorateur	16 jours	Lun 13/01/14	Lun 03/02/14	Jeremy
17	Début annuler-refaire/Recharger image originale/Changer racine explorateur	6 jours	Lun 03/02/14	Lun 10/02/14	Jeremy
18	Revue du code/Javadoc	6 jours	Lun 10/02/14	Lun 17/02/14	Jeremy
19	Raccourci clavier/Aperçu image dans ouvrir	6 jours	Lun 17/02/14	Lun 24/02/14	Jeremy
20	Gomme/Propriétés image/Résolution de problèmes	6 jours	Lun 24/02/14	Lun 03/03/14	Jeremy
21	Rapport	11 jours	Lun 03/03/14	Lun 17/03/14	Jeremy

Figure 2 : Diagramme de Gantt réalisé de Marc et Jérémy

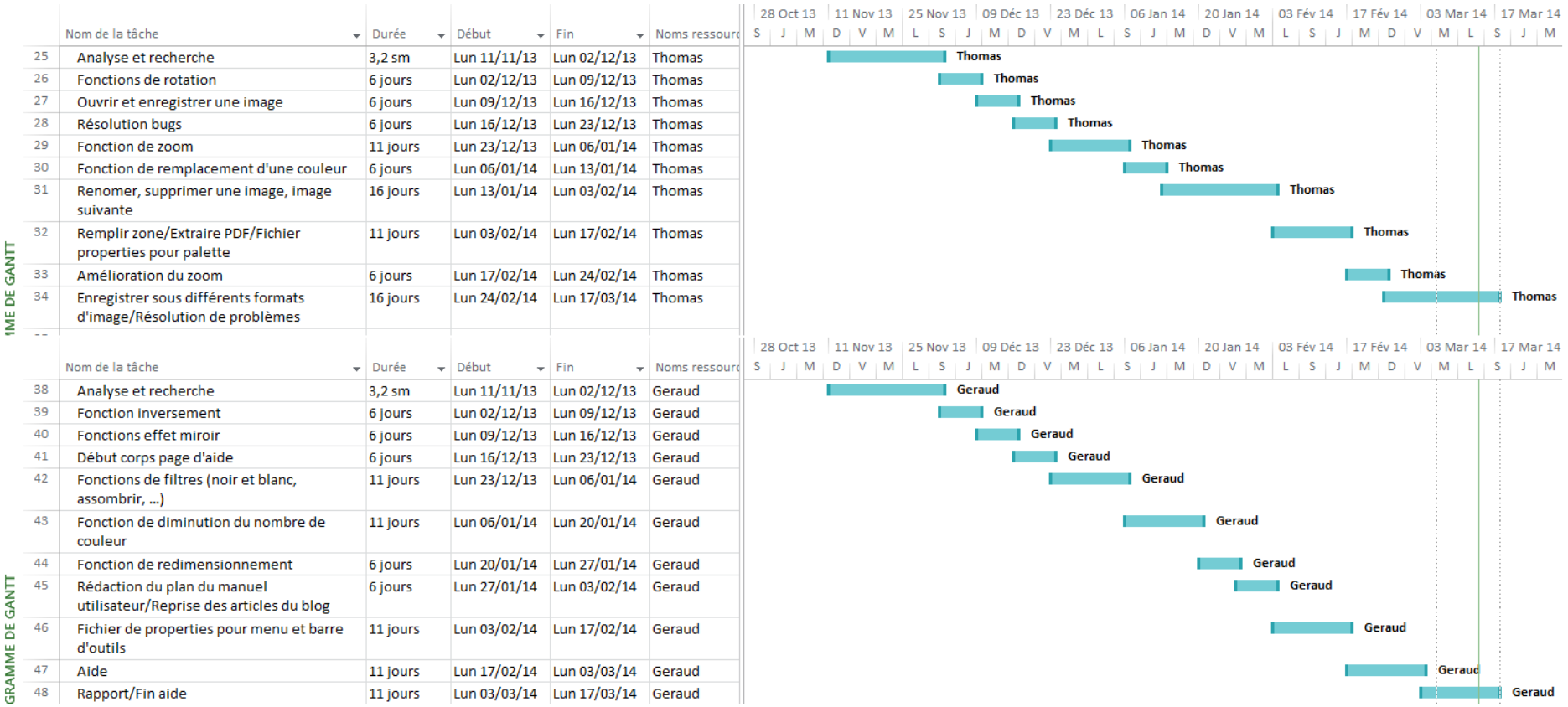


Figure 3 : Diagramme de Gantt réalisé de Thomas et Géraud

Figure 4 : Diagramme de classe de la Vue

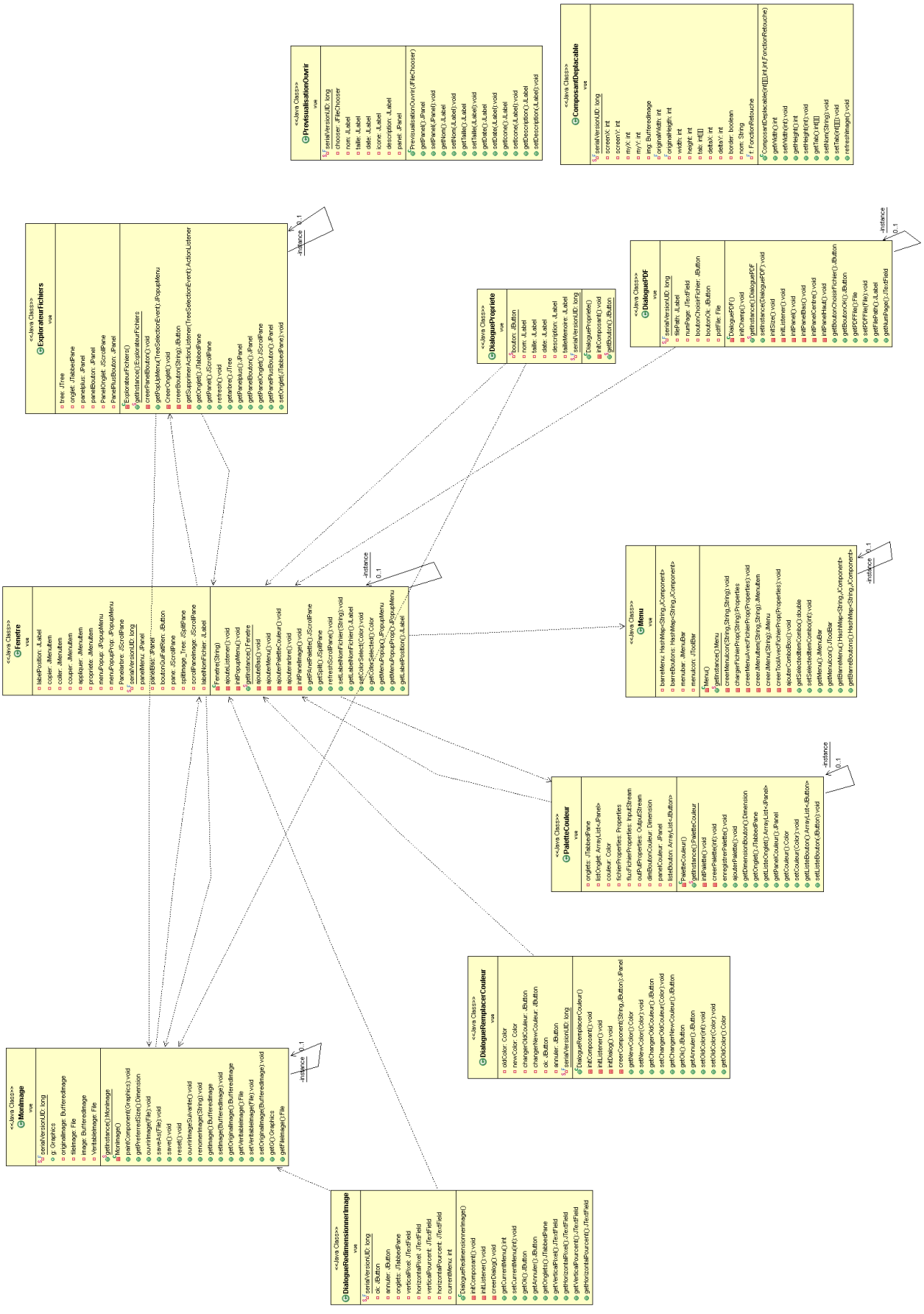
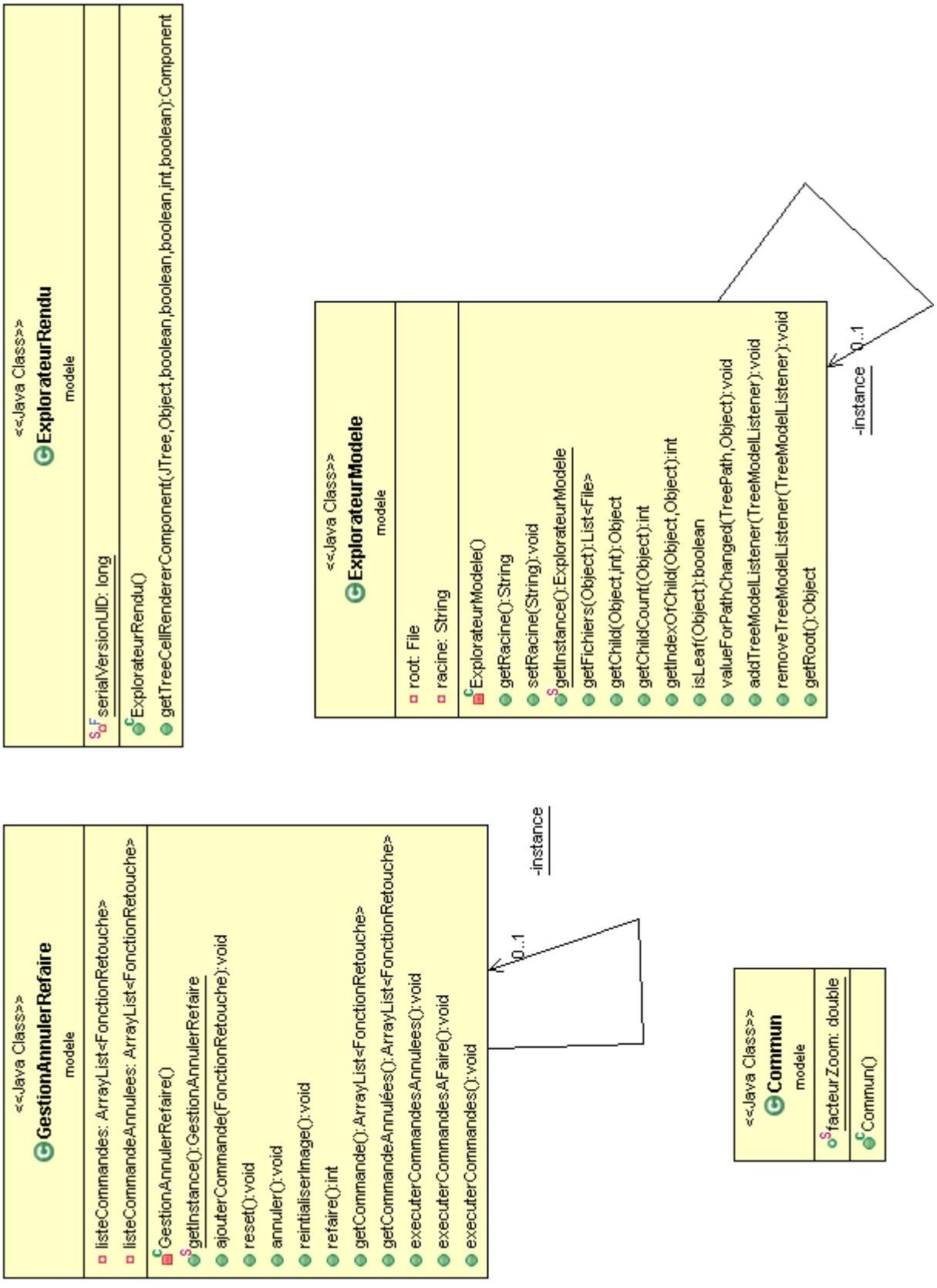


Figure 5 : Diagramme de classe du Modèle



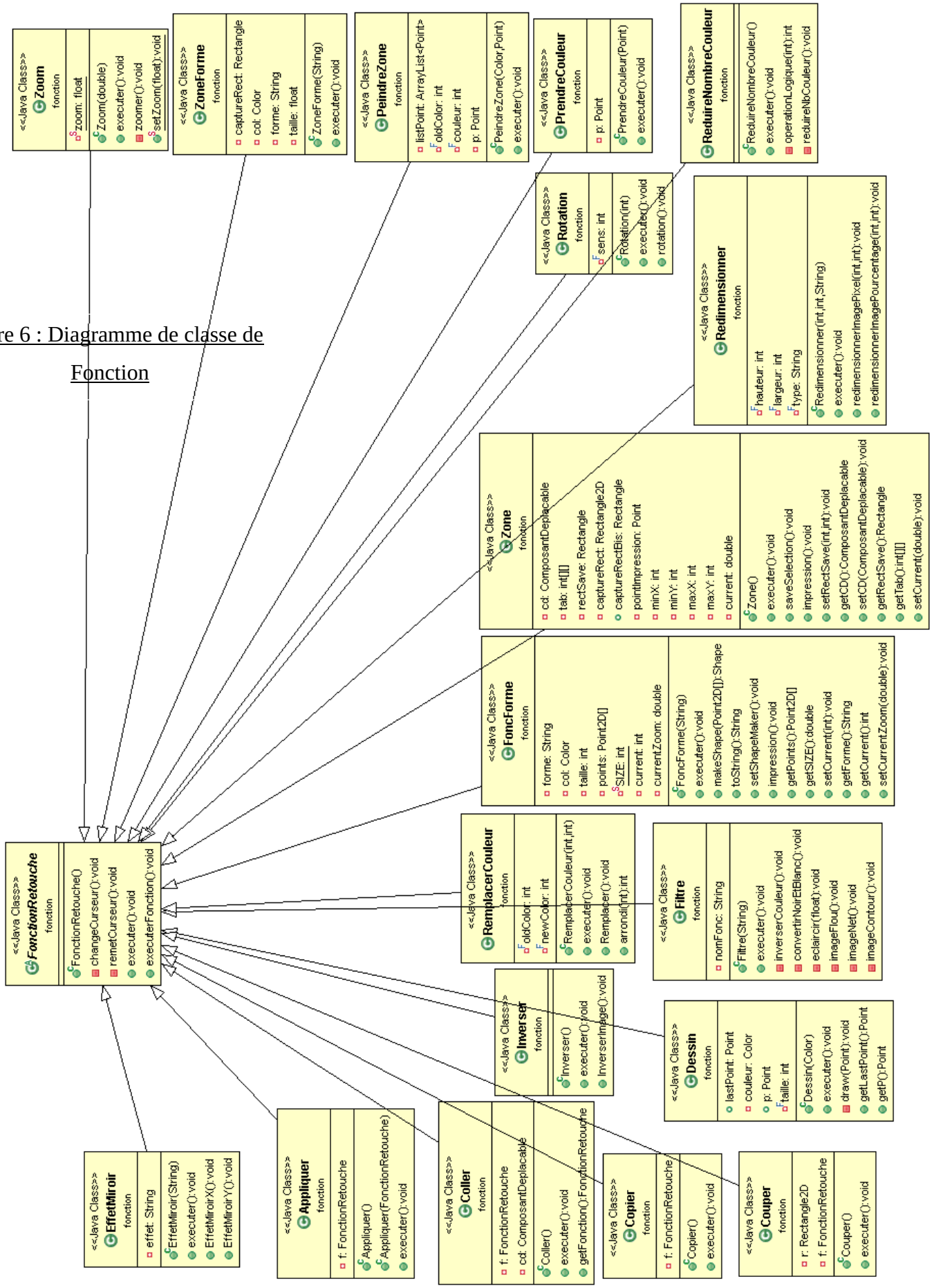


Figure 6 : Diagramme de classe de Fonction

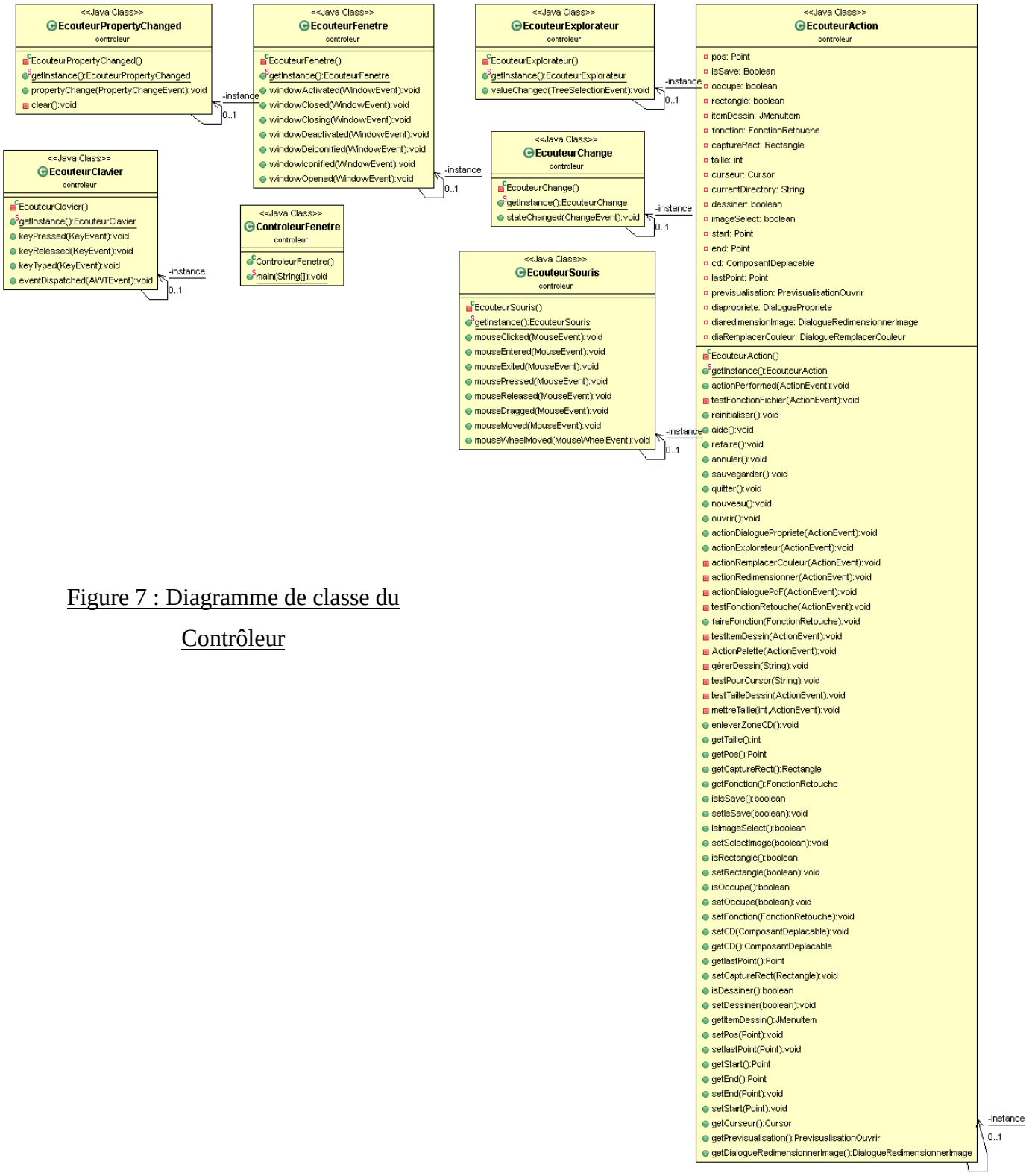


Figure 7 : Diagramme de classe du Contrôleur

```
#Clé=JMenu;JMenuItem;Chemin Image
#Menu fichier
109=Fichier;Nouveau;Icones/nouveau.png
108=Fichier;Ouvrir;Icones/icon_ouvrir.png
107=Fichier;Sauvegarder;Icones/save.png
106=Fichier;Sauvegarder-Sous;Icones/save.png
105=Fichier;Extraire PDF;Icones/pdf.png
104=Fichier;Ouvrir avec le logiciel par defaut;Icones/logiciel.png
103=Fichier;Image suivante;Icones/imageSuivante.png
102=Fichier;Renomer;Icones/renommerImage.png
101=Fichier;Supprimer Image;Icones/supprimerImage.png
100=Fichier;Quitter;Icones/quitter.png
#Menu image
18=Image;Effet miroir vertical;Icones/symétrie_verticale.png
17=Image;Effet miroir horizontal;Icones/symétrie_horizontale.png
16=Image;Inverser Image;Icones/inversion.png
15=Image;Rotation droite;Icones/rotation90-2.png
14=Image;Rotation gauche;Icones/rotation90-1.png
13=Image;Zoom +;Icones/zoom+.jpg
12=Image;Zoom -;Icones/zoom-.jpg
11=Image;Réduire nombre couleur;Icones/nbCouleur.png
10=Image;Redimensionner image;Icones/redimensionner.png
#Menu filtre
6=Filtre;Eclaircir image;Icones/luminosite.png
5=Filtre;Assombrir image;Icones/contraste.png
4=Filtre;Noir et blanc;Icones/noir-blanc-3.jpg
3=Filtre;Image en négatif;Icones/negatif-2.png
2=Filtre;Flouter une image;Icones/flouté.png
1=Filtre;Déflouter une image;Icones/déflouté.png
```

Figure 8 : Fichier Properties pour la barre de Menu

```
0=Filtre;Garder les contours d'une image;Icones/contour-2.png
#Menu forme
1008=Dessin;Dessiner un ovale;Icones/cercle.jpg
1007=Dessin;Dessiner un rectangle;Icones/rectangle.png
1006=Dessin;Dessiner une ligne;Icones/ligne.png
#Menu dessin
1005=Dessin;Remplacer Couleur;Icones/remplacerCouleur.png
1004=Dessin;Gommer l'image;Icones/gomme.png
1003=Dessin;Dessiner l'image;Icones/pinceau.jpg
1002=Dessin;Selectionner une zone;Icones/selection.jpg
1001=Dessin;Prendre une couleur;Icones/échantillon.png
1000=Dessin;Peindre une zone;Icones/potPeinture.png
#Menu taille
10007=Taille;Taille 1;Icones/pinceau.jpg
10006=Taille;Taille 2;Icones/pinceau.jpg
10005=Taille;Taille 3;Icones/pinceau.jpg
10004=Taille;Taille 4;Icones/pinceau.jpg
10003=Taille;Taille 5;Icones/pinceau.jpg
10002=Taille;Taille 6;Icones/pinceau.jpg
10001=Taille;Taille 7;Icones/pinceau.jpg
10000=Taille;Taille 8;Icones/pinceau.jpg
#Menu ?
100001=?;Aide;Icones/aide.jpg
100000=?;A Propos;Icones/EnSavoirPlus.jpg
#Edition
1000002=Edition;Annuler;Icones/annuler.png
1000001=Edition;Refaire;Icones/refaire.png
1000000=Edition;Reinitialiser;Icones/recharger.png
```

Figure 8 : Fichier Properties pour la barre de Menu

Figure 9 : Fichier Properties pour la barre d'icônes

```
#Clé=nomBouton;Chemin Image; texte ToolTip
37=separateur
36=bouton;boutonRefaire;Icones/refaire.png;Refaire
35=bouton;boutonAnnuler;Icones/annuler.png;Annuler
34=bouton;boutonReinitialiser;Icones/recharger.png;Reinitialiser
33=separateur
32=bouton;boutonOvale;Icones/cercle.jpg;Dessiner un ovale
31=bouton;boutonRectangle;Icones/rectangle.png;Dessiner un rectangle
30=bouton;boutonLigne;Icones/ligne.png;Dessiner une ligne
29=bouton;boutonPeindreZone;Icones/potPeinture.png;Peindre une zone
28=bouton;boutonPrendreCouleur;Icones/échantillon.png;Prendre une couleur
27=bouton;boutonGomme;Icones/gomme.png;Gommer l'image
26=bouton;boutonDessin;Icones/pinceau.jpg;Dessiner l'image
25=bouton;boutonRemplacerCouleur;Icones/remplacerCouleur.png;Remplacer Couleur
24=separateur
23=bouton;boutonRedimensionner;Icones/redimensionner.png;Redimensionner image
22=bouton;boutonSelection;Icones/selection.jpg;Selectionner une zone
21=separateur
20=bouton;boutonRotationG;Icones/rotation90-1.png;Rotation à gauche
19=bouton;boutonRotationD;Icones/rotation90-2.png;Rotation à droite
18=separateur
17=comboZoom;
16=bouton;boutonZoomM;Icones/zoom-.jpg;Effectuer un zoom en arrière
15=bouton;boutonZoomP;Icones/zoom+.jpg;Effectuer un zoom en avant
14=separateur
13=bouton;boutonAssombrir;Icones/contraste.png;Assombrir image
12=bouton;boutonEclaircir;Icones/luminosite.png;Eclaicir image
11=separateur
10=bouton;boutonReduireCouleur;Icones/nbCouleur.png;Réduire le nombre de couleur de
l'image
9=bouton;boutonNoirEtBlanc;Icones/noir-blanc-3.jpg;Niveau de gris
8=separateur
7=bouton;boutonInversion;Icones/inversion.png;Inverser une image
6=bouton;boutonMiroirV;Icones/symétrie_verticale.png;Symetrie verticale
```

5=bouton;boutonMiroirH;Icones/symétrie_horizontale.png;Symetrie horyzontale

4=separateur

3=bouton;boutonSauvegarder;Icones/save.png;Sauvegarder l'image

2=bouton;boutonSauvegarderSous;Icones/save.png;Sauvegarder l'image sous

1=bouton;boutonNouveau;Icones/nouveau.png;Créer nouvelle image

0=bouton;boutonOuvrir;Icones/icon_ouvrir.png;Ouvrir fichier

size=38

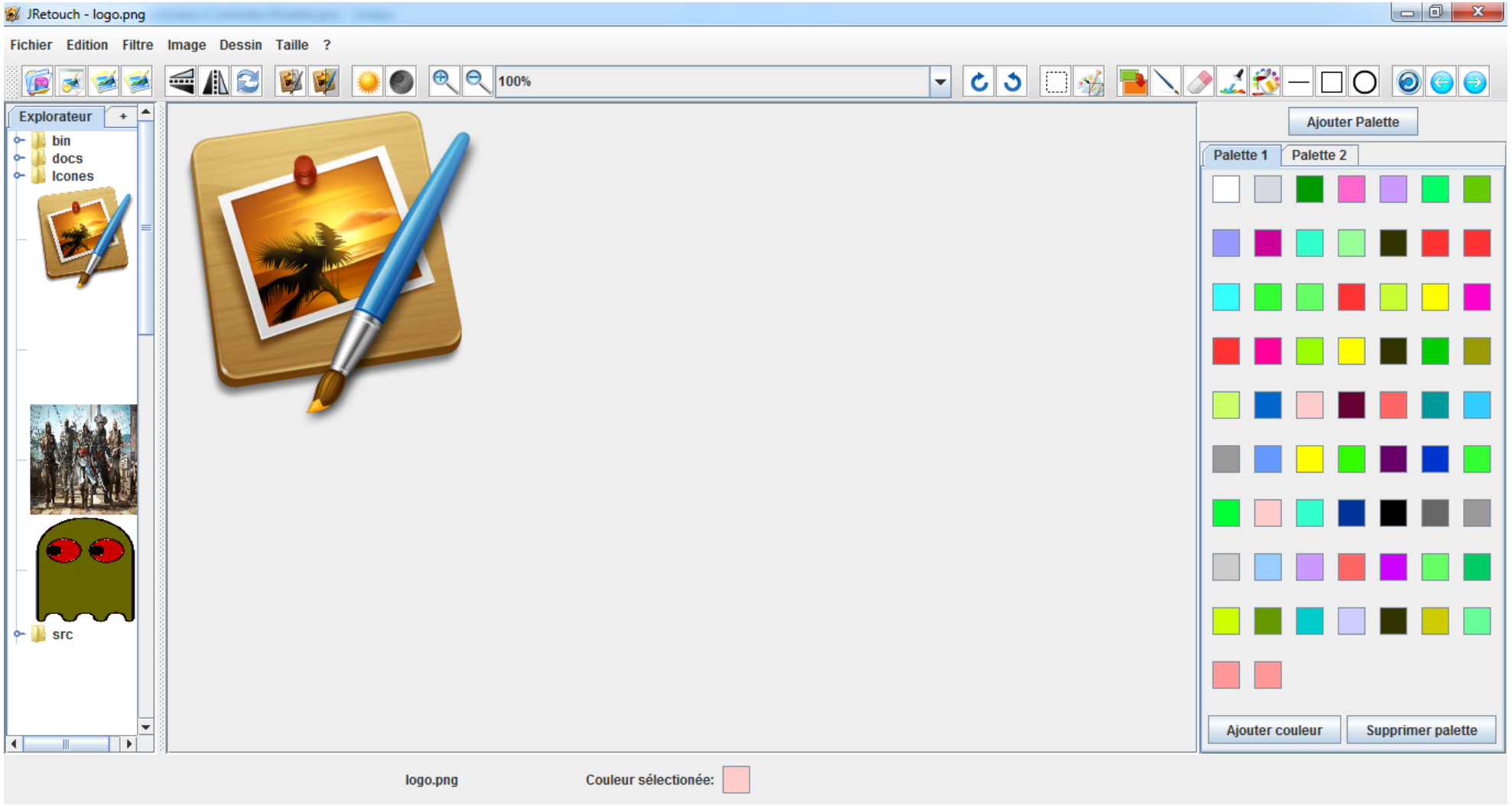


Figure 10: Apercu du Look And Feel

standard Java

1) Créer une nouvelle image

Figure 11: Plan Manuel Utilisateur

2) Ouvrir une image
    - 2.1) à partir du menu
    - 2.2) à partir de l'explorateur
3) Utilisation du zoom
4) Modifier une image
    - 4.1) effectuer une rotation dans le sens horaire
    - 4.2) effectuer une rotation dans le sens antihoraire
    - 4.3) effectuer un effet miroir vertical
    - 4.4) effectuer un effet miroir horizontal
    - 4.5) inverser une image
    - 4.6) réduire le nombre de couleur d'une image
    - 4.7) redimensionner une image
    - 4.8) assombrir une image
    - 4.9) éclaircir une image
    - 4.10) mettre une image en noir et blanc
    - 4.11) mettre une image sous forme de négatif
    - 4.12) flouter une image
    - 4.13) déflouter une image
    - 4.14) garder les contours d'une image
    - 4.15) dessiner sur une image
    - 4.16) dessiner des formes géométriques
    - 4.17) utiliser la gomme
    - 4.18) prendre une couleur
    - 4.19) remplacer une couleur par une autre
    - 4.20) peindre une zone
    - 4.21) copier/couper/coller une zone de l'image
5) Annuler/Refaire une action
6) Recharger l'image originale
7) Sauvegarder/Sauvegarder-sous une image
8) Extraire une page d'un document PDF

9) Passer à l'image suivante du répertoire courant

10) Renommer une image

11) Supprimer une image

    11.1) à partir du menu

    11.2) à partir de l'explorateur

12) Gérer les palettes de couleurs

13) Gérer l'explorateur

    13.1) changer la racine de l'explorateur

    13.2) changer la couleur de fond de l'explorateur

14) Ouvrir avec le logiciel par défaut

15) Fermer l'application

Page d’évaluation: