RAPPORT BPO

The Tiler team

KUMAR Aman
ROOBALO RODRIGUES Flavio **Groupe 106**

TABLE DES MATIERES

INTRODUCTION

Le jeu :

Le jeu the Tiler Team aussi connu sous le nom de TOOTY est un jeu collaboratif où une équipe de carreleurs coopèrent pour paver au mieux un mur. Pour cela nous disposons du mur à carreler ainsi que 9 carreaux de dimensions différentes de couleur rouge et les mêmes 9 carreaux de couleur bleu, une dernière pièce : la pièce neutre est également fournie. Ainsi qu'un paquet de 33 cartes qui viennent imposer les différentes contraintes du jeu tels que les couleurs des carreaux à jouer (bleu/rouge) ou encore les dimensions (taille 1, taille 2, taille 3) qui sont plus précisément définis dans l'énoncé.

Avant le début de chaque partie, le paquet de cartes doit être mélangé et tous les carreaux doivent être disposés. La pièce neutre est placée aléatoirement dans un des deux coins de la zone a carreler.

Les carreleurs jouent à tour de rôle. A chaque tour le carreleur doit choisir une carte dans le paquet, puis il doit choisir un carreau correspondant à la consigne de la carte et la placer dans le mur, uniquement si le carreau peut être placé tout en respectant certaines conditions (le carreau doit toucher un carreau et du basse différente ..etc.). Quand il a fini son tour sa carte est directement écartée.

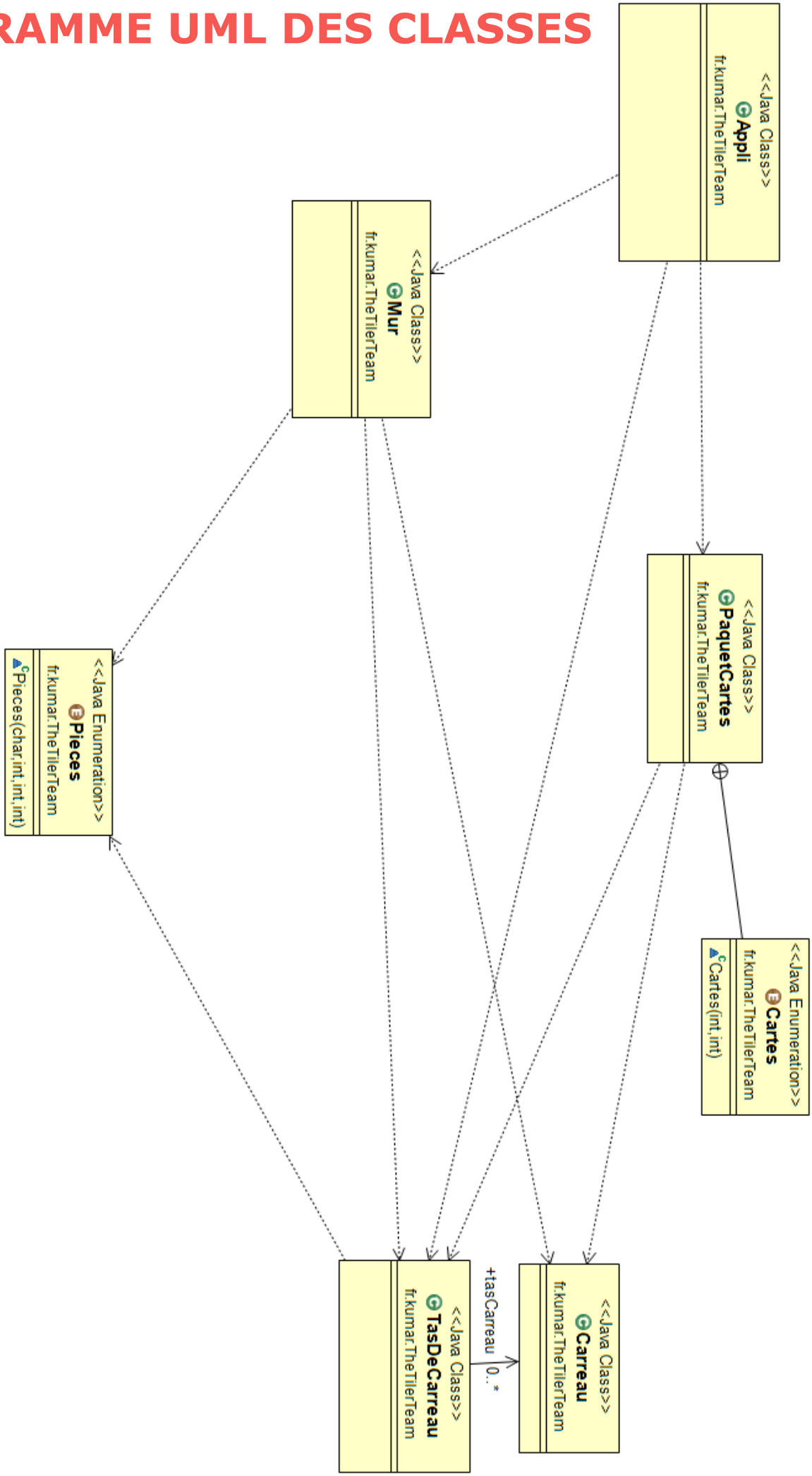
La partie se termine lorsque soit tous les carreaux ont été posés, soit les cartes à piocher sont épuisées soit les joueurs décident de stopper la partie car les risques de pénalités sont trop importants. L'équipe gagne des points en fonction du nombre de niveaux complétements carrelés, et sont pénalisés par le nombre de carreaux pas posés ainsi que le nombre de cartes dont la consigne n'a pas pu être réalisée.

Objectif :

L'objectif de ce projet est de réaliser un programme permettant de jouer une partie de Tiller Team dans sa totalité. Représentation Mur et Carreau dans le programme :

```
4
3 x       e                       e       g g         i i i
2 x   d d e           b       d d   e       g g   h h h   i i i
1 x A d d e       a   b   c c   d d   e   f f f   g g   h h h   i i i
  1 2 3 4 5
```

DIAGRAMME UML DES CLASSES



CLASSES

➢ Piece.java

```
/**
 *
 * @author KUMAR Aman Et ROBALO RODRIGUES Flavio GP-106
 *
 */

public enum Pieces {
    /*
     * Code couleur 4 --> Blue et 5--> Rouge
     */
    x('x', 1, 3,4),
    x2('x', 3, 1,4),
    a('a', 1, 1,4),
    b('b', 1, 2,4),
    c('c', 2, 1,4),
    d('d', 2, 2,4),
    e('e', 1, 3,4),
    f('f', 3, 1,4),
    g('g', 2, 3,4),
    h('h', 3, 2,4),
    i('i', 3, 3,4),
    A('A', 1, 1,5),
    B('B', 1, 2,5),
    C('C', 2, 1,5),
    D('D', 2, 2,5),
    E('E', 1, 3,5),
    F('F', 3, 1,5),
    G('G', 2, 3,5),
    H('H', 3, 2,5),
    I('I', 3, 3,5);


    private char letter;
    private int width;
    private int height;
    private int couleur;

    Pieces(char letter, int width, int height, int couleur) {
        this.letter = letter;
        this.width = width;
        this.height = height;
        this.couleur = couleur;
```

```
    }

    public char getLetter() {
        return this.letter;
    }

    public int getWidth() {
        return this.width;
    }

    public int getHeight() {
        return this.height;
    }

    public int getColor() {
        return this.couleur;
    }

}
```

➢ Carreau.java

```
package fr.kumar.TheTilerTeam;

/**
 *
 * @author KUMAR Aman Et ROBALO RODRIGUES Flavio GP-106
 *
 */

public class Carreau {

    private char idCarreau;
    private int coodx,coordy;
    private int hauteurCarreau;
    private int largeurCarreau;
    private int couleur;
    private boolean possé = false;

    public Carreau(char idCarreau, int hauteurCarreau, int largeurCarreau, int
couleur){
        this.idCarreau = idCarreau;
        this.hauteurCarreau = hauteurCarreau;
        this.largeurCarreau = largeurCarreau;
        this.couleur = couleur;
        this.coodx = this.coordy = -1;
    }

    /**
     * Methode permetant de initialiser les coordonnés d'un carreau
     * @param x int : la ligne
     * @param y int : la collone
     */
    public void postionCarreau(int x, int y){
        if(this.coodx != 0 && this.coordy !=0){
            this.coodx = x;
            this.coordy = y;
        }
    }
```

```
/**
 * Geter
 * @return int : retoune la ligne
 */
public int getCoodx(){
    return this.coodx;
}

/**
 * Geter
 * @return int : retoune la collone
 */
public int getCoordy(){
    return this.coordy;
}

/**
 * Geter
 * @return retourne la lettre qui correspond à carreau
 */
public char getIdCarreau() {
    return this.idCarreau;
}

/**
 * Getter
 * @return retourne l'hauteur qui correspond à carreau
 */
public int getHauteurCarreau() {
    return this.hauteurCarreau;
}

/**
 * Getter
 * @return int retourne la largeur qui correspond à carreau
 */
public int getLargeurCarreau() {
    return this.largeurCarreau;
```

```
    }

    /**
     * Getter
     * @return int retourne la couleur qui correspond à carreau
     */
    public int getCouleur() {
        return this.couleur;
    }

    /**
     * Méthode pemettant de savoir si le carreau a déja été possé da sle mour
     * @return boolean TRUE si carreau est deja possé
     */
    public boolean carreauEstDéjaPossé(){
        return this.possé;
    }

    /**
     * Méthode permet de posser le carreau. ²
     */
    public void pooserCarreau(){
        this.possé = true;
    }

}
```

✓ Test unitaire CarreauTest.java → s'exécute avec sucées

```
package fr.kumar.TheTilerTeam;
import org.junit.Test;
import static org.junit.Assert.*;
public class CarreauTest {
    @Test
    public void test(){
        Carreau c = new Carreau('a',1,1,4);
        assertEquals(c.getHauteurCarreau(),1);
        assertEquals(c.getLargeurCarreau(),1);
        assertEquals(c.getIdCarreau(),'a');
    }
}
```

➢ TasDeCarreau.java

```
package fr.kumar.TheTilerTeam;

import java.util.ArrayList;
import java.util.Arrays;

/**
 *
 * @author KUMAR Aman Et ROBALO RODRIGUES Flavio GP-106
 *
 */
public class TasDeCarreau {
    public ArrayList<Carreau> tasCarreau;

    public TasDeCarreau(){

      tasCarreau = new ArrayList<>(Arrays.asList(
        new Carreau(Pieces.a.getLetter(),Pieces.a.getHeight(),Pieces.a.getWidth(),Pieces.a.getColor()),
            new Carreau(Pieces.x.getLetter(),Pieces.x.getHeight(),Pieces.x.getWidth(),Pieces.x.getColor()),
            new Carreau(Pieces.x2.getLetter(),Pieces.x2.getHeight(),Pieces.x2.getWidth(),Pieces.x2.getColor()),
            new Carreau(Pieces.b.getLetter(),Pieces.b.getHeight(),Pieces.b.getWidth(),Pieces.b.getColor()),
            new Carreau(Pieces.c.getLetter(),Pieces.c.getHeight(),Pieces.c.getWidth(),Pieces.d.getColor()),
            new Carreau(Pieces.d.getLetter(),Pieces.d.getHeight(),Pieces.d.getWidth(),Pieces.d.getColor()),
            new Carreau(Pieces.e.getLetter(),Pieces.e.getHeight(),Pieces.e.getWidth(),Pieces.e.getColor()),
            new Carreau(Pieces.f.getLetter(),Pieces.f.getHeight(),Pieces.f.getWidth(),Pieces.f.getColor()),
            new Carreau(Pieces.g.getLetter(),Pieces.g.getHeight(),Pieces.g.getWidth(),Pieces.g.getColor()),
            new Carreau(Pieces.h.getLetter(),Pieces.h.getHeight(),Pieces.h.getWidth(),Pieces.h.getColor()),
            new Carreau(Pieces.i.getLetter(),Pieces.i.getHeight(),Pieces.i.getWidth(),Pieces.i.getColor()),
            new Carreau(Pieces.A.getLetter(),Pieces.A.getHeight(),Pieces.A.getWidth(),Pieces.A.getColor()),
            new Carreau(Pieces.B.getLetter(),Pieces.B.getHeight(),Pieces.B.getWidth(),Pieces.B.getColor()),
            new Carreau(Pieces.C.getLetter(),Pieces.C.getHeight(),Pieces.C.getWidth(),Pieces.C.getColor()),
            new Carreau(Pieces.D.getLetter(),Pieces.D.getHeight(),Pieces.D.getWidth(),Pieces.D.getColor()),
            new Carreau(Pieces.E.getLetter(),Pieces.E.getHeight(),Pieces.E.getWidth(),Pieces.E.getColor()),
            new Carreau(Pieces.F.getLetter(),Pieces.F.getHeight(),Pieces.F.getWidth(),Pieces.F.getColor()),
            new Carreau(Pieces.G.getLetter(),Pieces.G.getHeight(),Pieces.G.getWidth(),Pieces.G.getColor()),
            new Carreau(Pieces.H.getLetter(),Pieces.H.getHeight(),Pieces.H.getWidth(),Pieces.H.getColor()),
            new Carreau(Pieces.I.getLetter(),Pieces.I.getHeight(),Pieces.I.getWidth(),Pieces.I.getColor())



            ));
    }
```

```java
    /**
     * Mérhode permettant de savoir si tous les carreaux ont été possé.
     * @return bollean TRUE si tas de carreaux est vide.
     */
    public boolean estVide(){
        boolean vide = true;
        for (Carreau c:tasCarreau) {
            if(!c.carreauEstDéjaPossé()){
                vide = false;
                break;
            }
        }
        return vide;
    }

    /**
     * Methode permettant d'enlever un carreau du tasCarreau
     * @param c Carreau a supprimer
     */
    public void enleverCarreau(Carreau c){
        this.tasCarreau.remove(c);
    }

    /**
     * Méthode permettant de savoir le nombre de carreaux qui sont pas encore été possé
     * @return int nombre de carreau non possé
     */
    public int nbCarreauxNonPosés(){
        return this.tasCarreau.size();
    }
}
```

- ✓ Test unitaire TasDeCarreauTest.java → s'exécute avec sucées

```java
package fr.kumar.TheTilerTeam;

import org.junit.Test;
import static org.junit.Assert.*;
public class TasDeCarreauTest {
    @Test
    public void test(){
        TasDeCarreau tc = new TasDeCarreau();
        assert(tc.nbCarreauxNonPosés() == 20);
        assertTrue(tc.estVide());
    }
}
```

➢ PaquetCartes.java

```
package fr.kumar.TheTilerTeam;

import java.util.ArrayList;
import java.util.Collections;
import java.util.LinkedList;

/**
 *
 * @author KUMAR Aman Et ROBALO RODRIGUES Flavio GP-106
 *
 */

public class PaquetCartes {
    private final int nbCartes = 33;
    private ArrayList<Character> carreauTireParCarte;
    private boolean écarté = false;

    /*Enum cartes*/

    private enum Cartes{
        R(5,9),
        B(4,9),
        T1(1,5),
        T2(2,5),
        T3(3,5),
        ;

        private int code;
        private int totalCartes;

        Cartes(int code, int totalCartes) {
            this.code = code;
            this.totalCartes = totalCartes;
        }

        public int getNom() {
            return this.code;
        }

        public int getNbRep() {
            return this.totalCartes;
        }

    }
```

```
    //Piles des cartes
    private LinkedList<Integer> paquet ;

    public PaquetCartes(){
        paquet = new LinkedList<>();
        carreauTireParCarte = new ArrayList<Character>();
        for (Cartes c : Cartes.values()) {
            for (int i = 0; i < c.totalCartes; i++)
                paquet.add(c.code);
        }
        Collections.shuffle(paquet);
    }

    /**
     * Methose permettant de tirer une carte du paquer
     * @return int retoune le code associé a chaque carte
     */
    public int cartesTiré(){

        //Vérification si la pile est bien rempli
        assert(!this.paquet.isEmpty());
        int carteTiré2 = paquet.getLast();
        paquet.removeLast();
        if(!paquetEstVide() )
            return carteTiré2;
        else{
            System.out.println("Il n'existe aucun carreau qui coorespond à la carte tiré, donc la carte a été
écarté.");
            System.out.println("Voici la nouvelle carte:");
            carteTiré2 = paquet.getLast();
            paquet.removeLast();
            return carteTiré2;
        }

    }

    /**
     * Méthode permettant de vérifier si le paquet est vide
     * @return boolean retoune TRUE si paquet vide
     */
    public boolean paquetEstVide(){
        return this.paquet.isEmpty();
    }

    /**
     * Méthode permettant de comparer si le carreau sasie coorespond bien à la carte tiré
     * @param idCrreau lettre associé au carreau
```

```
 * @return boolean TRUE si le carreau vérifie la condition du carte
 */
public boolean comparerCarreauEtCarte(char idCrreau){
    boolean bonSaisie = false;
    for(int i = 0; i < carreauTireParCarte.size();i++){
        if(carreauTireParCarte.get(i) == idCrreau)
            bonSaisie = true;
    }
    return bonSaisie;
}

public boolean écarterCarte(){
    return this.écarté;
}

/**
 *Méthode renvoie la carte tiré
 * @return String renvoie la carte tiré sous forme d'une chaine de caractère
 */
public String getStringCartes(){
    String s = " ";
    int code = this.paquet.getLast();
    switch (code){
        case 5:
            s+= "La carte tiré est Rouge";
            break;
        case 4:
            s+= "La carte tiré est Blue";
            break;
        case 3:
            s+= "La carte tiré est de Taille 3";
            break;
        case 2:
            s+= "La carte tiré est de Taille 2";
            break;
        case 1:
            s+= "La carte tiré est de Taille 1";
            break;
        default:
            break;
    }
    return s;
}

/**
 * Méthode permettant d'afiicher toutes les carreaux  qui correspond à une carte
 * @param carte code de chaque carte (1 = taille 1, 2 = taille 2 ... 4 = blue...)
```

```
     * @param t TasDeCarreau contient toutes les carreuc non-posé
     */
    public String toString(int carte, TasDeCarreau t) {
        int maxHeight = -1;
        String s = "";

        for (Carreau p : t.tasCarreau) {
            if (p.getHauteurCarreau() > maxHeight) {
                maxHeight = p.getHauteurCarreau();
            }
        }

        for (int i = maxHeight; i > 0; i--) {
            for (Carreau p : t.tasCarreau) {
                if(p.getCouleur() == carte && !p.carreauEstDéjaPossé()){
                    this.carreauTireParCarte.add(p.getIdCarreau());
                    if (p.getHauteurCarreau() < i) {
                        s += "  ".repeat(p.getLargeurCarreau());
                    } else {
                        s += (" " + p.getIdCarreau()).repeat(p.getLargeurCarreau());
                    }
                    s += "  ";
                }
                else if((p.getLargeurCarreau() == carte || p.getHauteurCarreau() == carte) &&
!p.carreauEstDéjaPossé()){
                    this.carreauTireParCarte.add(p.getIdCarreau());
                    if (p.getHauteurCarreau() < i) {
                        s += "  ".repeat(p.getLargeurCarreau());
                    } else {
                        s += (" " + p.getIdCarreau()).repeat(p.getLargeurCarreau());
                    }
                    s += "  ";
                }
                else {
                    this.écarté = true;
                }
            }
            s += "\n";
        }

        return  s;
    }
}
```

- ✓ Test unitaire PaquetCartesTest.java → s'exécute avec sucées.

```
package fr.kumar.TheTilerTeam;

import org.junit.Test;
import static org.junit.Assert.*;
public class PaquetCartesTest {
    @Test
    public void test(){
        PaquetCartes pc = new PaquetCartes();
        assertTrue(pc.paquetEstVide());
    }
}
```

➢ Mur.java

```
package fr.kumar.TheTilerTeam;

import java.util.ArrayList;
import java.util.Random;

/**
 *
 * @author KUMAR Aman Et ROBALO RODRIGUES Flavio GP-106
 *
 */
public class Mur {
    private int nbLigneRemplis;
    private int score;
    private static final int LARGEUR = 5;
    private boolean ligneVide = true;

    private final String[] orient = new String[]{
            "Horiz - 1","Vertic - 1","Horiz - 5","Vertic - 5"
    };

    /*Grille de jeu*/
    private ArrayList<char []> grille = new ArrayList<char []>();

    public Mur(TasDeCarreau tc){
        grille.add(getLigneVide());
        int pN = this.positionCarreauNeutre();

        switch (pN){
            case 0:
                grille.add(getLigneVide());
                grille.get(1)[0] = 'x';
                grille.get(1)[1] = 'x';
                grille.get(1)[2] = 'x';
                for(Carreau c: tc.tasCarreau){
                    if(c.getIdCarreau() == 'x' && c.getLargeurCarreau() == 3){
                        c.pooserCarreau();
                        c.postionCarreau(1,0);
                    }
                }
                break;
            case 1:
                grille.add(getLigneVide());
                grille.add(getLigneVide());
                grille.add(getLigneVide());
                grille.get(1)[0] = 'x';
```

```
            grille.get(2)[0] = 'x';
            grille.get(3)[0] = 'x';
            for(Carreau c: tc.tasCarreau){
                if(c.getIdCarreau() == 'x' && c.getLargeurCarreau() == 1){
                    c.pooserCarreau();
                    c.postionCarreau(1,0);
                }
            }
            break;
        case 2:
            grille.add(getLigneVide());
            grille.get(1)[2] = 'x';
            grille.get(1)[3] = 'x';
            grille.get(1)[4] = 'x';
            for(Carreau c: tc.tasCarreau){
                if(c.getIdCarreau() == 'x' && c.getLargeurCarreau() == 3){
                    c.pooserCarreau();
                    c.postionCarreau(1,2);
                }
            }
            break;
        case 3:
            grille.add(getLigneVide());
            grille.add(getLigneVide());
            grille.add(getLigneVide());
            grille.get(1)[4] = 'x';
            grille.get(2)[4] = 'x';
            grille.get(3)[4] = 'x';
            for(Carreau c: tc.tasCarreau){
                if(c.getIdCarreau() == 'x' && c.getLargeurCarreau() == 3){
                    c.pooserCarreau();
                    c.postionCarreau(1,4);
                }
            }
            break;

    }
    for(Carreau c: tc.tasCarreau){
        if(c.getIdCarreau() == 'x' && !c.carreauEstDéjaPossé()){
            c.pooserCarreau();
            c.postionCarreau(10,10);
        }
    }
    this.nbLigneRemplis = this.grille.size();
    this.score = 0;

}
```

```
/**
 * Méthode prmettant d'obtenir la position du carreau neutre
 * @return int position du carreau
 */
private int positionCarreauNeutre(){
    Random r = new Random();
    int indice = r.nextInt(orient.length );
    return indice;
}

/**
 * Méthode permettant de créer une ligne vide
 * @return char[] retoune un tableau de type char qui
 *          contient comme charactere " ".
 */
private char[] getLigneVide() {
    char [] ligneVide = {' ',' ',' ',' ',' '};
    return ligneVide;
}

/**
 * Méthode permettant de compter nombre de lignes remplit
 * @return int ligne remplit du grille
 */
public int compteNbLigneCompléte(){
    boolean ligneVide = false;
    for(int i = 0; i < LARGEUR; i++){
        if(this.grille.get(this.nbLigneRemplis - 1)[i] == ' '){
            ligneVide = true;
            break;
        }
    }
    if(!ligneVide)
        this.score++;

    return 5*this.score;
}

/**
 * Calculer le scoore en fonction du ligne du grille remplit
 * @return int score
 */
public int getScore() {
    return score;
}
```

```
    /**
     * Méthode permettant de placer un carreau dans la grille.
     * @param ligne int ligne sur la quel il faut placer le carreau
     * @param col   int collone sur la quel il faut placer le carreau
     * @param c     char le carreau a placé, désignié par son lettre
     * @param tc    TasCarreau qui contient les carreaux non possé
     * @return boolean retoune TRUE si le carreau a bien été possé
     */
    public boolean placerCarreau(int ligne, int col, char c, TasDeCarreau tc){
        boolean carreauPossé = false;
        for (Carreau p : tc.tasCarreau) {
            if(p.getIdCarreau() == c && !p.carreauEstDéjaPossé()
                    && this.toucherUnCarreau(this.grille.size()
                    - ligne,col - 1,p.getLargeurCarreau()))
            {

                this.agrendirMur(p.getHauteurCarreau(),p.carreauEstDéjaPossé());
                if(this.vérifeBasseDuCarreau(p.getLargeurCarreau(),this.grille.size() - ligne,col,tc)
                        && !this.pasCloner(p,this.grille.size() - ligne ,col,tc)){


                    if(this.placePourCarreau(ligne,col,p.getHauteurCarreau(),p.getLargeurCarreau())){
                        p.postionCarreau(this.grille.size() - ligne,col -1);
                        for(int k = this.grille.size() - ligne; k > (this.grille.size() - ligne) -
p.getHauteurCarreau() ;k--){
                            for(int y = col - 1; y < (p.getLargeurCarreau() + col) - 1;y++ ){

                                this.grille.get(k)[y] = p.getIdCarreau();
                                p.pooserCarreau();
                                carreauPossé = true;

                            }
                        }

                        break;

                    }
                }

            }

        }

        return carreauPossé;
    }

    /**
```

```
    * Méthode permetatnt de savoir si les longeurs de deux carreaux est identique ou non
    * @param c       Carreau a posser
    * @param x       int coordonnés du carreau a pooser (ligne)
    * @param y       int coordonnés du carreau a pooser (collone)
    * @return        boolean TRUE si les deux ont la meme longueur sur des bords
    */
   private boolean pasCloner(Carreau c,int x ,int y,TasDeCarreau tc){
      boolean clone = false;

      for(Carreau d:tc.tasCarreau){
         if((d.carreauEstDéjaPossé() && d.getCoodx() == x)
               && (d.getCoordy() == (y+c.getLargeurCarreau()) - 1 || (d.getCoordy() +
d.getLargeurCarreau()) + 1 == y)){
            if(d.getHauteurCarreau() == c.getHauteurCarreau()){
               clone = true;
               break;
            }

         }
      }

      if(clone)
         System.out.println("Le carreau colone un autre carreau");
      return clone;
   }

   /**
    *Méthode permettant de savoir si la basse du carreau
    *n'est pas identique au largeur d'un carreau deja possé
    * @param largeur int largeur du carreau a posser
    * @param x       int coordonnés du carreau a pooser (ligne)
    * @param y       int coordonnés du carreau a pooser (collone)
    * @return        boolean FALSE si les deux ont la meme basse
    */
   private boolean vérifeBasseDuCarreau(int largeur, int x , int y,TasDeCarreau tc){
      boolean baseValide = true;
      if(x != this.grille.size() - 1){
         for (Carreau c: tc.tasCarreau){
            if(c.getCoordy() == y && c.getLargeurCarreau() == largeur && c.carreauEstDéjaPossé()){
               baseValide = true;
            }
         }
      }

      if(!baseValide)
         System.out.println("Le carreau est sur une meme base que sa largeur ");
```

```
    return baseValide;
}

/**
 * Méthode permettant de savoir si il y des casses vides pour un le carreau à placer
 * @param ligne     int coordonnés du carreau a pooser
 * @param col       int coordonnés du carreau a pooser
 * @param hauteur   int hauteur de carreau à posser
 * @param largeur   int largeur de carreau à posser
 * @return          boolean TRUE si les casses du grille sont vide
 */
private boolean placePourCarreau(int ligne, int col, int hauteur, int largeur){
    boolean placeCarreau = true;
    for(int k = this.grille.size() - ligne; k > (this.grille.size() - ligne) - hauteur ;k--){
        for(int y = col - 1; y < (largeur + col) - 1;y++ ){
            if(y < LARGEUR){
                if(this.grille.get(k)[y] != ' '){
                    placeCarreau = false;
                    break;
                }
            }
            else
                placeCarreau = false;
        }
    }

    if(!placeCarreau)
        System.out.println("Le careau ne peut pas être placé car il n'a pas de place!");
    return  placeCarreau;
}

/**
 * Méthode pour vérifier si le carreau a posser se situe bien à coté d'un carreu déja possé.
 * @param x         int coordonnés du carreau a pooser (lignes)
 * @param y         int coordonnés du carreau a pooser (collone)
 * @param largeur   int largeur de carreau a posser
 * @return          boolean TRUE si le carreau touche bien un autre carreau
 */
private boolean toucherUnCarreau(int x, int y, int largeur ){
    boolean toucher = false;
    if((y > 0 && y < LARGEUR ) && x != this.grille.size() - 1){
        if(this.grille.get(x)[y-1] != ' ' ){
            toucher = true;
        }
        else if(this.grille.get(x)[y+1] != ' ')
            toucher = true;
        else if(this.grille.get(x + 1)[y] != ' ')
```

```
            toucher = true;
        }
        if(y == 0 && x != this.grille.size() - 1){
            if(this.grille.get(x)[y+1] != ' ')
                toucher = true;
            else if(this.grille.get(x + 1)[y] != ' ')
                toucher = true;
        }
        if(y == LARGEUR - 1 && x != this.grille.size() - 1){
            if(this.grille.get(x)[y-1] != ' ')
                toucher = true;
            else if(this.grille.get(x + 1)[y] != ' ')
                toucher = true;
        }
        if(x == this.grille.size() - 1){
            if (y == 0){
              if(this.grille.get(x)[y+1] != ' ' || this.grille.get(x)[(y + largeur)] != ' ')
                  toucher = true;
            }
            else if(y == LARGEUR - 1){
                if(this.grille.get(x)[y-1] != ' '){
                    toucher = true;
                }
            }
            else{
                if(this.grille.get(x)[y-1] != ' ')
                    toucher = true;
                else if(this.grille.get(x)[y+1] != ' ')
                    toucher = true;
            }

            if(!toucher)
                System.out.println("Le carreau touche un autre carreau !");
        }
        return toucher;
    }

    /**
     * Méthode permettant de ajouter des lignes à grille en fonction d'hauteur de carreau à placer
     * @param h         int hauteur de carreau à placer
     * @param possé     boolean TRUE si le carreau a déja été possé
     */
    private void agrendirMur(int h,boolean possé){
        if (h >= this.nbLigneVide() ){
            if (!possé)
                for(int i = this.nbLigneVide() ; i < h ; i++){
                    this.grille.add(0, getLigneVide());
```

```
            }
        }
    }

    /**
     * Méthode permetant de savoir le nombre de ligne vide du grille
     * @return  int retourne nombre de lignes vide
     */
    private int nbLigneVide(){
        int cmpt = 0;
        for(int i = 1;i<this.grille.size();i++){
            for(int j =0; j<LARGEUR;j++){
                if(this.grille.get(i)[j] == ' '){
                    cmpt++;
                    j = LARGEUR + 1;
                }
            }
        }
        return cmpt;
    }

    /**
     * Envoyer la chaine de caractere du mur
     * @return  String chaine de caractere
     */
    public String  toString(){
        String s = "";
        for (int i = 0; i <grille.size(); i++){
            s += (grille.size() - i) +" ";
            for(int j = 0; j<LARGEUR;j++){
                s += (grille.get(i)[j] + "  ");
            }
            s += "\n";
        }

        s += " ";

        for(int j = 0; j<LARGEUR;j++){
            s += (" "+ (j+1)+" ");
        }
        return s;
    }
}
```

➢ Appli.java

```
package fr.kumar.TheTilerTeam;
import java.util.Scanner;

/**
 *
 * @author KUMAR Aman Et ROBALO RODRIGUES Flavio GP-106
 *
 */

public class Appli {

    public static void main(String[] arg) {
        boolean gameOver = false;
        boolean carreauPlacé = false;
        int nbCarteEcarté = 0;
        PaquetCartes c = new PaquetCartes();
        TasDeCarreau p = new TasDeCarreau();
        Mur m = new Mur(p);


        Scanner sc = new Scanner(System.in);
        char Careau = ' ';
        String choix = " ";
        int ligne = 0;
        int colonne = 0;
        do {
            System.out.println();
            System.out.println("| Vous pouvez soit stoper la partie en saisissant 'stop' |");
            System.out.println( "| Soit écarter la carte si vous ne trouvez pas de solurion: 'next' |");
            System.out.println("| Soit placer un carreau en saisissant: Lettre Ligne Colonne |");
            System.out.println();

            System.out.println(m);
            System.out.println();
            System.out.println(c.getStringCartes());
            int carte = c.cartesTiré();
            System.out.println();
            System.out.println(" Voici les carreaux corespondant au carte tiré:");
            System.out.print(c.toString(carte,p));

            System.out.print("Votre choix:");

            choix = sc.next();

            if(choix == "stop"){
```

```
            gameOver = true;
        }

        if(choix == "next"){
            c.cartesTiré();
            nbCarteEcarté++;

        }

        if(sasieCorrecte(choix)){
            Careau = choix.charAt(0);
            ligne = sc.nextInt();
            colonne = sc.nextInt();
            if(c.comparerCarreauEtCarte(Careau)){
                carreauPlacé = m.placerCarreau( ligne, colonne, Careau,p);

                if(!carreauPlacé){
                    System.out.println("Le carreau sasie n'a pas été palcé car il ne respecte pas les régle de jeu, veuillez sasir un autrre carreau ou choisri next ou stop!");
                    do {
                        choix = sc.next();
                        if(choix == "stop"){
                            gameOver = true;
                        }

                        if(choix == "next"){
                            c.cartesTiré();
                            nbCarteEcarté++;

                        }
                        if(sasieCorrecte(choix)){
                            Careau = choix.charAt(0);
                            ligne = sc.nextInt();
                            colonne = sc.nextInt();
                            carreauPlacé = m.placerCarreau( ligne, colonne, Careau,p);
                        }

                    }while(!carreauPlacé);

                    carreauPlacé = false;
                }
            }
            else{
                do{
                    System.out.println("Le carreau sasie ne correspond à la carte tiré, Sasiez a nouveau le carreau a placé et ses coordonnées:");
                    Careau = choix.charAt(0);
```

```
                    ligne = sc.nextInt();
                    colonne = sc.nextInt();
                    m.placerCarreau( ligne, colonne, Careau,p);
                }while (!c.comparerCarreauEtCarte(Careau));

            }
        }

        if(p.estVide() || c.paquetEstVide())
            gameOver = true;

        System.out.println();
        System.out.println(( m.compteNbLigneCompléte() - (p.nbCarreauxNonPosés() +
nbCarteEcarté) )+ " points ("+ m.getScore()
                + " niveaux complets, " + " " + p.nbCarreauxNonPosés() + " carreaux non posés, " +
nbCarteEcarté + " cartes écaartées");


    } while (!gameOver);


}

public static boolean sasieCorrecte(String s){
    boolean bonSasie = false;
    if(s.length() <=6 ){
        char ch = s.charAt(0);
        for(Pieces p:Pieces.values()){
            if(ch == p.getLetter())
                bonSasie = true;
        }
    }
    return bonSasie;
}


}
```

BILAN

Notre premier projet en langage JAVA, nous a permis du travailler dans un autre aspect de la programmation à savoir la programmation orienté objets. De plus, Le cahier des charges qui nous a été demandé est respecté. Notre application possède les fonctionnalités minimales demandées. Ce projet, nous a permis d'appliquer et d'approfondir nous connaissance en programmation et surtout en langage JAVA, malgré quelques difficultés rencontrées.

- Les difficultés rencontrées :

La principale difficulté que nous avons rencontrée, ce fut au début de projet. En effet, nous avons eu un peu de mal à bien structurer notre projet en différentes classes et donc de bien choisir quel classes /objet était vraiment utile et efficace. Nous avons, peut ainsi comprendre que à quel point la programmation objet est très efficace puisque à travers les classes et les méthodes bien structuré on peut créer des programmes très efficaces et bien optimisé.

La deuxième difficulté était d'imaginer et coder les algorithmes afin de respecter différents fonctionnements du jeux The Tiler Team. Les algorithmes utilisé ont une complexité très grande.

- ✓ Ce qui est réussi :

Le cahier de charge est respecté. Notre jeu fonctionne bien avec des classes et bien définit et ordonné. En effet, par exemple pour les carreaux nous avons une classe Carreau qui contient un carreau et une autre classe TasDeCarreau qui contient tous les carreaux. Ce qui donne une bonne structure pour les carreaux.

Ensuite, nous avons des méthodes dans la portée au niveau du paquetages est bien définit par exemple : pour la classe Mur.java, seule méthode publique sont placerCarreau et toString, puisque ce sont les seules méthodes qui sont nécessaire en dehors de la classe.

- Ce qui peut être amélioré :

La classe Appli.java peut-être amélioré, plus ordonné, avec une structure plus agréable. En créant par exemple une autre classe Jouer.java, afin de l'utiliser dans la classe Appli.java pour réduire une utilisation de « if » et aussi réduire la répétition du code.

On aura pu bien-sûr amélioré les algorithmes de vérification des différentes règles du jeux, de les rendre moins complexe.

On aura pu aussi ajouter dans la classe paquetCartes.java un autre constructeur qui permettrai d’ajouter plus de cartes au jeux The Tiler Team.