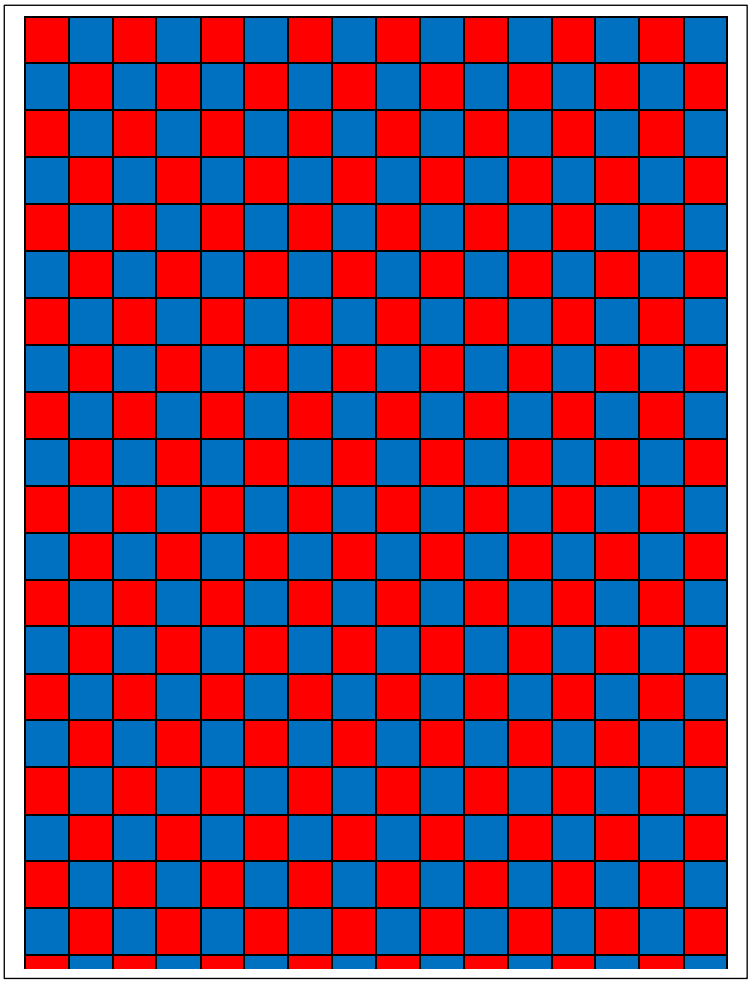
The Tiler Team, un jeu de pavage

- L'objet du dossier est de programmer un jeu de pavage.



10 MARS

Créé par :

- **Fangyuan Lisa YE (Groupe 109)**
- **Alysson RODRIGUEZ (Groupe 110)**

UNIVERSITÉ PARIS DESCARTES

Université de Paris

Table des matières

Introduction

Le projet Java que nous devons réaliser consiste un jeu collaboratif dans lequel une équipe de carreleurs participe pour paver un mur. Pour ce faire, nous devons réaliser un programme qui permet de jouer une partie dans sa totalité. En résultat, il faut que chaque carreau soit identifié par une lettre distincte en fonction de sa taille. Le carreau doit être déposé dans le mur en respectant les contraintes imposées.

Les différentes étapes afin de paver un mur :

- Un paquet de 33 cartes composé :
  - 9 cartes bleues
  - 9 cartes rouges
  - 5 cartes « Taille 1 »
  - 5 cartes « Taille 2 »
  - 5 cartes « Taille 3 »

Le paquet de cartes doit mélanger les cartes puis les faces imagées doivent être poser face au sol. Puis le carreleur doit piocher une carte située au sommet du paquet.

- Les carreaux que les carreleurs vont poser sur le mur ont une largeur et une hauteur comprises entre 1 et 3 unités. En totalité, nous avons 19 carreaux :
  - 9 carreaux bleus
  - 9 carreaux rouges
  - 1 carreau neutre

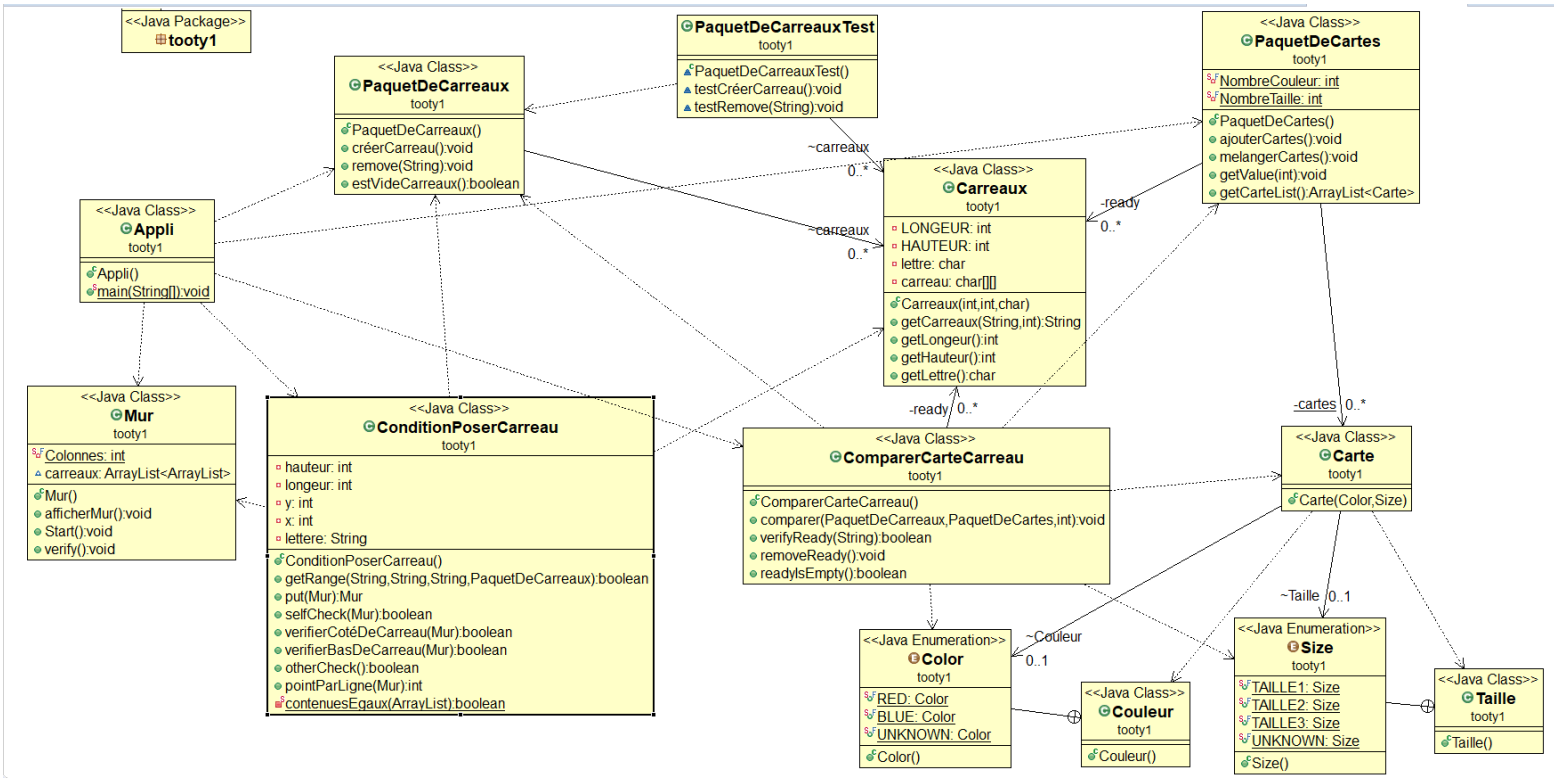
Le carreleur doit choisir un carreau en fonction des consignes de la carte qu'il a choisi puis le placer sur le mur s'il peut. En effet, il peut seulement placer les carreaux sur le mur s'il a respecté les six contraintes imposées dans le jeu.

- Le mur est d'une forme rectangulaire composant d'une largeur de 5 unités et d'une hauteur assez grande pour contenir tous les carreaux que les carreleurs vont poser.

La partie s'arrête seulement si tous les carreaux ont été posés, les cartes à piocher sont épuisées et que les joueurs décident de stopper la partie.
Des points sont attribués à l'équipe en fonctions des niveaux complétés ou sinon des pénalités.

Diagramme UML

- Ci-dessous nous vous avons représenté un diagramme UML de notre application sous forme de paquetage avec les attributs et méthodes.



Tests Unitaires

```
package tooty1;

import static org.junit.jupiter.api.Assertions.*;

import java.util.ArrayList;
import java.util.Iterator;

import org.junit.jupiter.api.Test;

class PaquetDeCarreauxTest {
	ArrayList<Carreaux> carreaux  = new ArrayList<Carreaux>();
	@Test
	void testCréerCarreau() {

		carreaux.add(new Carreaux(1,1,'a'));
		carreaux.add(new Carreaux(2,1,'b'));
		carreaux.add(new Carreaux(1,2,'c'));
		carreaux.add(new Carreaux(2,2,'d'));
		carreaux.add(new Carreaux(3,1,'e'));
		carreaux.add(new Carreaux(1,3,'f'));
		carreaux.add(new Carreaux(3,2,'g'));
		carreaux.add(new Carreaux(2,3,'h'));
		carreaux.add(new Carreaux(3,3,'i'));
		carreaux.add(new Carreaux(1,1,'A'));
		carreaux.add(new Carreaux(2,1,'B'));
		carreaux.add(new Carreaux(1,2,'C'));
		carreaux.add(new Carreaux(2,2,'D'));
		carreaux.add(new Carreaux(3,1,'E'));
		carreaux.add(new Carreaux(1,3,'F'));
		carreaux.add(new Carreaux(3,2,'G'));
		carreaux.add(new Carreaux(2,3,'H'));
		carreaux.add(new Carreaux(3,3,'I'));

		assertTrue(carreaux.size()==18);
	}

}
```

Il est beaucoup trop tard en moment qu'on sait qu'il faillait faire tests unitaires. En gros c'est un test qui nous permet de débogueur les problèmes de codes. Il nous permet de vérifie les problèmes de code mais pas logique de développeur.

Code Java

```java
package tooty1;

/**
 * Projet BPO
 * @author Fangyuan Lisa YE 109 and Alysson Rodriguez 110
 * @version 09/03/2020
 *Tailles possibles
 */
public class Taille {
	public enum Size{
		TAILLE1,
		TAILLE2,
		TAILLE3,
		UNKNOWN;
		}

}
```

```java
package tooty1;

/**
 *Couleurs possible
 */
public class Couleur {
	public enum Color {
		RED,
		BLUE,
		UNKNOWN;
	}
}
```

```java
package tooty1;

//Type données des cartes
public class Carte {
	Couleur.Color Couleur;
	Taille.Size Taille;

	/**
	 * @param c
	 * @param t
	 */
	public Carte (Couleur.Color c, Taille.Size t){
		this.Couleur = c;
		this.Taille = t;
	}
}
```

```java
package tooty1;

import java.util.ArrayList;
/**
 * Type de donée de carte
 */
public class PaquetDeCartes {
	private static final int NombreCouleur = 9;
	private static final int NombreTaille = 5;
	private static ArrayList<Carte> cartes = new ArrayList<Carte>();

    //ajouter les cartes dans un liste de carte
	public void ajouterCartes() {
		for(int i = 0; i < NombreTaille; i++) {
			cartes.add(new Carte(Color.UNKNOWN,Size.TAILLE1));
			cartes.add(new Carte(Color.UNKNOWN,Size.TAILLE2));
			cartes.add(new Carte(Color.UNKNOWN,Size.TAILLE3));
		}

		for(int j = 0; j < NombreCouleur; j++) {
			cartes.add(new Carte(Color.BLUE,Size.UNKNOWN));
			cartes.add(new Carte(Color.RED,Size.UNKNOWN));
		}
	}

	//Melanger le liste de carte
	public void melangerCartes() {
	  Collections.shuffle(cartes);
	}


	//Le carte a tire
	public void getValue(int i) {
		if(cartes.get(i).Taille == Size.UNKNOWN){
			System.out.println(cartes.get(i).Couleur + "--" + i);
		}
		else {
			System.out.println(cartes.get(i).Taille  + "--"+ i);
		}
	}

	//obtient les valeurs de liste de carte
	public ArrayList<Carte> getCarteList(){
		return this.cartes;
	}
}
```

```java
package tooty1;


public class Carreaux {
	private int LONGEUR;
	private int HAUTEUR;
	private char lettre;
	private char[][] carreau;

	//type de données d'un carreau
	public Carreaux(int hauteur, int longeur, char a){
```

```
		this.HAUTEUR = hauteur;
		this.LONGEUR = longeur;
		this.lettre = a;
		this.carreau = new char[this.HAUTEUR][this.LONGEUR];
		for(int i = 0; i< this.HAUTEUR;i++) {
			for(int j = 0; j<this.LONGEUR; j++) {
				this.carreau[i][j] = this.lettre;
			}
		}
	}

	//afficher le carreaux
	public String getCarreaux(String s, int n){
		if(n > this.HAUTEUR) {
			for(int i = 0; i < this.carreau[0].length; ++i){
				s += " ";
			}
		}else {
			for(int i = 0; i < this.carreau[0].length; ++i){
				s += this.carreau[n-1][i];
			}
		}
		s += " ";
		return s;
	}

	//obtient Longuer
	public int getLongeur() {
		return this.LONGEUR;
	}

	//obtient Hauteur
	public int getHauteur() {
		return this.HAUTEUR;
	}

	//obtient Lettre qui costruire les carreaux
	public char getLettre() {
		return this.lettre;
	}
}
```

```
package tooty1;

import java.util.ArrayList;
import java.util.Iterator;

/**
 *Le list dynamique de paquet de carreaux
 */
public class PaquetDeCarreaux {
	ArrayList<Carreaux> carreaux  = new ArrayList<Carreaux>();


	public void créerCarreau() { //rentrer tous les carreaux possibles
		carreaux.add(new Carreaux(1,1,'a'));
		carreaux.add(new Carreaux(2,1,'b'));
```

```
		carreaux.add(new Carreaux(1,2,'c'));
		carreaux.add(new Carreaux(2,2,'d'));
		carreaux.add(new Carreaux(3,1,'e'));
		carreaux.add(new Carreaux(1,3,'f'));
		carreaux.add(new Carreaux(3,2,'g'));
		carreaux.add(new Carreaux(2,3,'h'));
		carreaux.add(new Carreaux(3,3,'i'));
		carreaux.add(new Carreaux(1,1,'A'));
		carreaux.add(new Carreaux(2,1,'B'));
		carreaux.add(new Carreaux(1,2,'C'));
		carreaux.add(new Carreaux(2,2,'D'));
		carreaux.add(new Carreaux(3,1,'E'));
		carreaux.add(new Carreaux(1,3,'F'));
		carreaux.add(new Carreaux(3,2,'G'));
		carreaux.add(new Carreaux(2,3,'H'));
		carreaux.add(new Carreaux(3,3,'I'));
	}


	/**
	 * supprimer les carreaux
	 * @param lettere
	 */
	public void remove(String lettere) {
		for (Iterator<Carreaux> iter = carreaux.iterator(); iter.hasNext();) {
			Carreaux i = iter.next();
		if(String.valueOf(i.getLettre()).equals(lettere)){
		 carreaux.remove(i);
		 return;
		}
	  }
	}


	//verifier que les carte est epuisser
	public boolean estVideCarreaux() {
		return carreaux.size() < 1;
	}
}
```

```
package tooty1;

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

import tooty1.Couleur.Color;
import tooty1.Taille.Size;

public class ComparerCarteCarreau {

	private ArrayList<Carreaux> ready = new ArrayList<Carreaux>();
	//compare les carte tirée et trouver les carreaux correspondant
		public void comparer(PaquetDeCarreaux pdc, PaquetDeCartes p, int n) {
			ArrayList<Carte> cartes = p.getCarteList();
			Color c = cartes.get(n).Couleur;
			Size s = cartes.get(n).Taille;
```

```
                        String s1 = new String();
                        String s2 = new String();
                        String s3 = new String();
                        ArrayList<Character> tile1 = new ArrayList<Character>();
                        ArrayList<Character> tile2 = new ArrayList<Character>();
                        Collections.addAll(tile1,'a','b','c','d','e','f','g','h','i');
                        Collections.addAll(tile2,'A','B','C','D','E','F','G','H','I');
                        if(c == Color.UNKNOWN) {
                                switch(s){
                                        case TAILLE1:
                                                for(int i = 0; i <  pdc.carreaux.size(); ++i) {
                                                        if(pdc.carreaux.get(i).getHauteur() == 1
                                                                        ||
pdc.carreaux.get(i).getLongeur()== 1) {
                                                                s1 = pdc.carreaux.get(i).getCarreaux(s1,1);
                                                                s2 = pdc.carreaux.get(i).getCarreaux(s2,2);
                                                                s3 = pdc.carreaux.get(i).getCarreaux(s3,3);
                                                                ready.add(pdc.carreaux.get(i));
                                                        }
                                                }
                                                break;
                                        case TAILLE2:
                                                for(int i = 0; i <  pdc.carreaux.size(); ++i) {

                                                        if(pdc.carreaux.get(i).getHauteur() == 2
                                                                        || pdc.carreaux.get(i).getLongeur()
== 2){
                                                                s1 = pdc.carreaux.get(i).getCarreaux(s1,1);
                                                                s2 = pdc.carreaux.get(i).getCarreaux(s2,2);
                                                                s3 = pdc.carreaux.get(i).getCarreaux(s3,3);
                                                                ready.add(pdc.carreaux.get(i));
                                                        }
                                                }
                                                break;
                                        case TAILLE3:
                                                for(int i = 0; i <  pdc.carreaux.size(); ++i) {
                                                        if(pdc.carreaux.get(i).getHauteur() == 3
                                                                        || pdc.carreaux.get(i).getLongeur()
== 3) {
                                                                s1 = pdc.carreaux.get(i).getCarreaux(s1,1);
                                                                s2 = pdc.carreaux.get(i).getCarreaux(s2,2);
                                                                s3 = pdc.carreaux.get(i).getCarreaux(s3,3);
                                                                ready.add(pdc.carreaux.get(i));
                                                        }
                                                }
                                                break;
                                        default:
                                }
                        }
                        else {
                                if (c == Color.BLUE) {
                                        for(int i = 0; i <  pdc.carreaux.size(); ++i) {
                                                if(tile1.contains(pdc.carreaux.get(i).getLettre())) {
                                                        s1 = pdc.carreaux.get(i).getCarreaux(s1,1);
                                                        s2 = pdc.carreaux.get(i).getCarreaux(s2,2);
                                                        s3 = pdc.carreaux.get(i).getCarreaux(s3,3);
                                                        ready.add(pdc.carreaux.get(i));
                                                }
                                        }
                                }
                                else {
                                        for(int i = 0; i <  pdc.carreaux.size(); ++i) {
                                                if(tile2.contains(pdc.carreaux.get(i).getLettre())) {
                                                        s1 = pdc.carreaux.get(i).getCarreaux(s1,1);
                                                        s2 = pdc.carreaux.get(i).getCarreaux(s2,2);
```

```
                                                s3 = pdc.carreaux.get(i).getCarreaux(s3,3);
                                                ready.add(pdc.carreaux.get(i));
                                        }
                                }
                        }
                }
                System.out.println();
                System.out.println(s3);
                System.out.println(s2);
                System.out.println(s1);
        }

        //trouver les carresux qui pourra etre utiliser
        public boolean verifierCarreauxPossible(String lettere) {
                boolean pass = false;
                for(Carreaux i: ready)
                        if(String.valueOf(i.getLettre()).equals(lettere))
                                pass = true;
                return pass;
        }

        //supprimer les carresux qui pourra etre utiliser
        public void supprimerCarreauxPossibles() {
                ready.clear();
        }

        //Verifier les carreaux possible pour le carte precedent est bien supprimer
        public boolean supprimerCarreauxPossible() {
                if(ready.isEmpty())
                        return true;
                return false;
        }

}
```

```
package tooty1;

import java.util.ArrayList;
import java.util.Collections;
import java.util.Iterator;
import java.util.Random;

public class Mur {
        private static final int Colonnes = 6; //Largeur de mur est fixe en 5
        ArrayList<ArrayList> carreaux  = new ArrayList<ArrayList>();

        public Mur() { //Initialiser les bases de mur
                carreaux.add(new ArrayList<String>());
                carreaux.get(0).add("  ");
                for(int i = 1; i < Colonnes; ++i) {
                        carreaux.add(new ArrayList<String>());
                        carreaux.get(i).add(i+"");
                }
        }

        public void afficherMur() {  //afficher le mur
                for(int i = 0; i < Colonnes; ++i) {
                        Collections.reverse(carreaux.get(i)); // inverser les ArrayList
carreaux
                }
```

```
        Iterator<String> it0 = carreaux.get(0).iterator(); //créer Interator
        Iterator<String> it1 = carreaux.get(1).iterator();
        Iterator<String> it2 = carreaux.get(2).iterator();
        Iterator<String> it3 = carreaux.get(3).iterator();
        Iterator<String> it4 = carreaux.get(4).iterator();
        Iterator<String> it5 = carreaux.get(5).iterator();

        System.out.println("-----------");
        for(; it0.hasNext();) {
               System.out.print(it0.next()+" "); // afficher le mur
               System.out.print(it1.next()+" ");
               System.out.print(it2.next()+" ");
               System.out.print(it3.next()+" ");
               System.out.print(it4.next()+" ");
               System.out.print(it5.next()+" ");
               System.out.println();
        }
        System.out.println("-----------");
        for(int i = 0; i < Colonnes; ++i) {  //retourner le mur
               Collections.reverse(carreaux.get(i));
        }
}

//initialiser le mur avec un carreau neutre
public void Start() {
        Random r = new Random();
        int num = r.nextInt(4)+1; //choisi en hasart un chiffre entre 1 et 4
        switch(num) {
        case 1:
               carreaux.get(1).add("x");
               carreaux.get(2).add("x");
               carreaux.get(3).add("x");
               for(int i = 0; i < 2; ++i)
                      carreaux.get(0).add(" "+(i+1));
               break;
        case 2:
               carreaux.get(1).add("x");
               carreaux.get(1).add("x");
               carreaux.get(1).add("x");
               for(int i = 0; i < 4; ++i)
                      carreaux.get(0).add(" "+(i+1));
               break;
        case 3:
               carreaux.get(3).add("x");
               carreaux.get(4).add("x");
               carreaux.get(5).add("x");
               for(int i = 0; i < 2; ++i)
                      carreaux.get(0).add(" "+(i+1));
               break;
        case 4:
               carreaux.get(5).add("x");
               carreaux.get(5).add("x");
               carreaux.get(5).add("x");
               for(int i = 0; i < 4; ++i)
                      carreaux.get(0).add(" "+(i+1));
               break;
        default:;
        }
}
```

```
	public void verifierHauteurMur() { //Remplie le case vide de la mur et garder un
liste vide en plus
		ArrayList<Integer> sizeOfCarreaux = new ArrayList<Integer>();
		for(ArrayList i: carreaux) //prendres le grandeur de tous les lists qui
constitue le mur
			sizeOfCarreaux.add(i.size());
		Collections.sort(sizeOfCarreaux); // met en l'ordre ces chiffres

		//chercher les difference entre les plus haut list et le plus bas list
		int diff = sizeOfCarreaux.get(sizeOfCarreaux.size()-1) -
sizeOfCarreaux.get(0);
		// ajouter les chiffres de l'ordonnée s'il n'est pas plus long
		for(int i = carreaux.get(0).size(); i <
sizeOfCarreaux.get(sizeOfCarreaux.size()-1); ++i)
			if(i < 10)carreaux.get(0).add(" "+i);
			else carreaux.get(0).add(i+"");
		// remplir les espaces vide de mur par " "
		if(diff!=0)
			for(ArrayList i: carreaux)
				for(;i.size()<sizeOfCarreaux.get(sizeOfCarreaux.size()-1);)
					i.add(" ");
	}



}
```

```
package tooty1;

import java.util.ArrayList;
import java.util.HashSet;
import java.util.Iterator;

//cordonnées de mur + lettre
public class ConditionPoserCarreau {
	private int hauteur = 0;
	private int longeur = 0;
	private int y;
	private int x;
	private String lettere;

	//obtenir les donées comme longuer et largeur de carreau choisi
	public boolean getRange(String lettere, String y, String x, PaquetDeCarreaux pdc) {
		boolean pass = false;
		try {
			this.y = Integer.valueOf(y);
			this.x = Integer.valueOf(x);
		}catch(Exception e) {
			return pass;
		}
		this.lettere = lettere;
		for (Iterator<Carreaux> iter = pdc.carreaux.iterator(); iter.hasNext();) {
			Carreaux i = iter.next();
		if(String.valueOf(i.getLettre()).equals(lettere)){
		 hauteur = i.getHauteur();
		 longeur = i.getLongeur();
		}
```

```
		}
			return pass = true;
		}

		//insere le carreau choisi dans le mur
		public Mur put(Mur m) {
			if(this.y + this.hauteur >= m.carreaux.get(x).size()) {
				for(int i = 0; i <= (this.y + this.hauteur) -
m.carreaux.get(x).size();;) {
					m.carreaux.get(x).add(" ");
				}
			}
			m.verifierHauteurMur();
			for(int i = 0; i < this.longeur; ++i) {
				for(int j = 0; j < this.hauteur; ++j) {
					m.carreaux.get(x+i).set(y+j, this.lettere);
				}
			}
			return m;
		}


		//verfier est ce qu'il exist déjà des lettres presentes dans le cases que les
		// utilisateurs choisi et eviter le depassement de mur
		public boolean selfCheck(Mur m) {
			boolean pass = false;
			if(this.y < 1)return pass;
			if(this.x < 1 || this.x > 5)return pass;
			if(m.carreaux.size() < this.x + this.longeur) return pass;
			for(int i = 0; i < this.longeur; ++i) {
				int h = this.hauteur;
				if(this.y + this.hauteur >= m.carreaux.get(x).size()) h =
m.carreaux.get(x).size() - this.y;
				for(int j = 0; j < h; ++j) {
					if(m.carreaux.get(x+i).get(y+j).equals(" "));
					else return pass;
				}
			}
			return pass = true;
		}

		//Verifier les contenue des cases de deux côté
		public boolean verifierCotéDeCarreau(Mur m) {
			boolean pass = false;
			ArrayList<Object> arrayLeft = new ArrayList<Object>();
			ArrayList<Object> arrayRight = new ArrayList<Object>();
			int h = this.hauteur;
			if(this.y + h >= m.carreaux.get(x).size())
				h = m.carreaux.get(x).size() - this.y;
			for(int i = 0; i < h; ++i) {
				if(m.carreaux.size() <= this.x + this.longeur) {
					arrayLeft.add(m.carreaux.get(x-1).get(y+i));
				}
				else {
					if(this.x == 1)
						arrayRight.add(m.carreaux.get(x+this.longeur).get(y+i));
					else{
						arrayLeft.add(m.carreaux.get(x-1).get(y+i));
						arrayRight.add(m.carreaux.get(x+this.longeur).get(y+i));
```

```
                        }
                    }
                }

                if(this.y == 1) {
                    if(this.x == 1) {
                        if(arrayRight.get(0).equals(" "))return pass;
                        if(contenuesEgaux(arrayRight)) {
                            if(arrayRight.get(0) !=
m.carreaux.get(x+this.longeur).get(y-1)
                                    && arrayRight.get(0) !=
m.carreaux.get(x+this.longeur).get(y+this.hauteur))
                                return pass;
                        }
                    }else {
                        if(this.x + this.longeur >= m.carreaux.size()) {
                            if(arrayLeft.get(0).equals(" "))
                                return pass;
                            if(contenuesEgaux(arrayLeft)) {
                                if(arrayLeft.get(0) != m.carreaux.get(x-1).get(y-1)
                                        && arrayLeft.get(0) !=
m.carreaux.get(x-1).get(y+this.hauteur))
                                    return pass;
                            }
                        }
                        else {
                            if(arrayRight.get(0).equals(" ") &&
arrayLeft.get(0).equals(" "))
                                return pass;
                        }
                    }
                }
                else{
                    if(this.y + this.hauteur < m.carreaux.get(x).size()) {
                        if(contenuesEgaux(arrayLeft)) {
                            if(arrayLeft.get(0) != m.carreaux.get(x-1).get(y-1)
                                    && arrayLeft.get(0) != m.carreaux.get(x-
1).get(y+this.hauteur))
                                return pass;
                        }
                        if(contenuesEgaux(arrayRight)) {
                            if(arrayRight.get(0) !=
m.carreaux.get(x+this.longeur).get(y-1)
                                    && arrayRight.get(0) !=
m.carreaux.get(x+this.longeur).get(y+this.hauteur))
                                return pass;
                        }
                    }
                }
                return pass = true;
        }

        //verifier le cases en bas de carreau choisi
        public boolean verifierBasDeCarreau(Mur m) {
            boolean pass = false;
            ArrayList<Object> arrayBottom = new ArrayList<Object>();
            for(int i = 0; i < this.longeur; ++i)
                arrayBottom.add(m.carreaux.get(x+i).get(y-1));
            if(arrayBottom.contains(" "))return pass;
```

```
			if(y != 1) {
				if(contenuesEgaux(arrayBottom)) {
					if(this.x == 1) {
						if(arrayBottom.get(0) !=
m.carreaux.get(x+this.longeur).get(y-1))
							return pass;
					}
					else {
						if(m.carreaux.size() <= this.x + this.longeur)	{
							if(arrayBottom.get(0) != m.carreaux.get(x-1).get(y-
1))
								return pass;
						}
						else {
							if(arrayBottom.get(0) != m.carreaux.get(x-1).get(y-
1)
									&& arrayBottom.get(0) !=
m.carreaux.get(x+this.longeur).get(y-1))
								return pass;
						}
					}
				}
			}
			return pass = true;
		}

		// verifier autre choses qu'on peut ajouter après
		public boolean autreContraint() {
			boolean pass = false;
			return pass;
		}


		//compter le point obtient par le ligne remplie
		public int pointParLigne(Mur m) {
			int point = 0;
			for(int i = 1; i < m.carreaux.get(0).size(); ++i) {
				for(ArrayList j : m.carreaux) {
					if(j.get(i).equals(" "))
						return point = (i - 1);
				}
				point = i;
			}
			return point;
		}

		//Verifier que les contenues de cases en bas du carreau choisi sont de même choses
		private static boolean contenuesEgaux(ArrayList list){
			if(null == list)
				return false;
			return 1 == new HashSet<Object>(list).size();
		}
}
```

```
package tooty1;

import java.util.Scanner;

/**
 * Projet BPO
```

```
 * @author Fangyuan Lisa YE 109 and Alysson Rodriguez 110
 * @version 09/03/2020
 */
public class Appli {

	public static void main(String[] args) { //L'entrée de jeu
		boolean game = true; //commencement de joue
		PaquetDeCartes p = new PaquetDeCartes();
		ComparerCarteCarreau ccc = new ComparerCarteCarreau();
		p.ajouterCartes();
		p.melangerCartes();
		int calc = 0; // nombre d'indice de n ieme cartes
		int ecart = 0; // nombre de cartes que le joueur l'ecarter
		int point = 0; // initaliser le point
		PaquetDeCarreaux pdc = new PaquetDeCarreaux();
		pdc.créerCarreau();

		System.out.println("---------Game Start---------\r");
		System.out.println("Ecarter la carte : next");
		System.out.println("Arrêter la partie : stop\r");

		Mur m = new Mur();//initialiser le mur
		m.Start();
		m.verifierHauteurMur();
		m.afficherMur();
		p.getValue(calc);
		ccc.comparer(pdc,p,calc); //trouver les carreaux
		++calc;                                  //correspondance de la carte tirée
		ConditionPoserCarreau poser = new ConditionPoserCarreau();
		Scanner scan = new Scanner(System.in);
		for(;game;) {
			if(scan.hasNextLine()) {
				String in = scan.nextLine(); //Entrez le lettre et les coordonnes de carreau que
vous voulez poser
				String[] str = in.split(" ");

				switch(in) {
					case "next": //entrer next pour ecarter les cartes
						if(calc < 33) {
							ccc.supprimerCarreauxPossibles();
							for(;ccc.supprimerCarreauxPossible();) {
								p.getValue(calc);
								ccc.comparer(pdc,p,calc);
								++ecart;
								++calc;
								--point;
							}
						}
						else {
							game = false; //Fin de joue si les cartes sont epuisser
							System.out.println("Tous les carreaux sont posés");
						}
						continue;

					case "stop": //entrer stop pour arreter le jeu
						game = false;
						continue;
					default: //dans le cas normal entrer le lettre et les coordonées de
carreau
							//pour poser le carreau sur le mur
						if(!ccc.verifierCarreauxPossible(str[0])) {
							System.out.println("Les carreaux choisi n'est pas
disponible"
									+ ",veuillez ressaisir :");
							continue;
						}
						if(!poser.getRange(str[0], str[1], str[2], pdc)) {
							System.out.println("Entrez le lettre et les cordonnées "
									+ "de carreaux d'un façon correcte,
veuillez ressaisir :");
							continue;
						}
						if(!poser.selfCheck(m)){
							System.out.println("Le case est déjà occupe ou "
```

```
                                                                        + "depasse la zone, veuillez ressaisir
:");
                                                        continue;
                                                }
                                                if(!poser.verifierCotéDeCarreau(m)){
                                                        System.out.println("Le carreau ne doit pas cloner"
                                                                        + " un autre carreau, veuillez ressaisir
:");
                                                        continue;
                                                }
                                                if(!poser.verifierBasDeCarreau(m)) {
                                                        System.out.println("Le carreau n'est pas stable ou
cloner "
                                                                        + "un carreau déjà posé,veuillez
ressaisir :");
                                                        continue;
                                                }
                                        }

                                        m = poser.put(m); //Poser le carreau
                                        pdc.remove(str[0]); // Supprimer le carreau
                                        if(pdc.estVideCarreaux()) {
                                                game = false; // Si tous les carreaux sont utilise alors fin de joue
                                                continue;
                                        }
                                        ccc.supprimerCarreauxPossibles(); // Supprimer les carreaux possible en chaque
tour

                                        m.verifierHauteurMur();
                                        m.afficherMur();
                                        for(;ccc.supprimerCarreauxPossible();) {
                                                p.getValue(calc);
                                                ccc.comparer(pdc,p,calc);
                                                ++calc;
                                        }
                                }
                        }
                        scan.close();
                        point -= pdc.carreaux.size();
                        int line = poser.pointParLigne(m);
                        point += line*5; //5 point pour un ligne remplie
                        System.out.println(point + " points ( "
                        + line + " niveaux complets, " + pdc.carreaux.size()
                        + " non posées, " + ecart + " cartes écartées )");
                        System.out.println("---------E N D---------");
                }

}
```

Bilan

- Les difficultés rencontrées dans ce projet :

La prmière difficulté de ce projet était le langage de programmation Java. C'est un nouveau langage que nous venions de découvrir en Amphi, TD et TP. Il se montrait plus facile à apprendre que les autres languages que nous avions fait (C ou C++). Néanmoins, il restera dure à coder en fonction des règles demandées par le jeu. Dans « The Tiler Team », les règles ne sont pas nombreuses mais le jeu reste très difficile à programmer. L'organisation du programme était plus difficle que les projets précédents car nous n'avons pas su par où et comment commencer le projet. Les consignes sont détaillées mais nous n'avions aucune fonction donée afin de mieux oraganiser notre programme. Nous avions mis énormément de temps à coder chaque fonction car cela néccessite un code structuré. Parfois, nous nous demandions si notre fonction pouvait avoir une telle structure. La partie la plus dure dans le programme était d'efecctuer les six contraintes qui nous ont été donnés. En effet, nous avons essayé de faire les contraintes demandées avant de déposer le carreau.

Nous avons eu des difficultés à faire les tests unitaires car nous avons pas compris comment les coder.

- Les réussites engendrées par ce projet :

Grâce au TD et TP que nous avions réalisé, nous avons pu faire les différentes classes et fonctions. Nous avons réussi à intégrer et comprendre certaines notions telles que « les getters et setters » que nous n'avions pas compris durant les cours de Java. Nous avons atteint d'afficher les cartes et les carreaux. Pour finir nous nous sommes parvenues à faire certaines contraintes.

- Les améliorations qui peuvent être fait pour ce projet :

Les améliorations que nous pouvons réaliser seraient sur un niveau de programmation. Dans un premier temps, on aurait pu améliorer l'organisations des classes car cela nous a pris énormément du temps. Nous avons du refaire au moins 2 fois notre programme.

Dernièrement, nous avons structurer notre code au maximum mais on aurait pu mieux faire.