“TBL_Dice.java”

```
package pieces;

public class TBL_Dice {
	private int result;

	public TBL_Dice () {result = 0;}

	public int RollDice () {
		result = ((int)(Math.random() * (double)6)) + 1;
		return result;
	}

	public int GetDiceResult () {return result;}
}
```

“TBL_Piece.java”

```
package pieces;

import board.TBL_Coord;

public class TBL_Piece {
	private boolean isFinished;
	private TBL_Coord pos;
	private int color;
	private int status; // 0-home - 7-finish - 1-running
	private int count; // 1 tour = 42
	public static final int HOME = 0;
	public static final int FINISH = 7;
	public static final int RUN = 1;

	public TBL_Piece () {
		Init();
		color = 0;
	}

	public TBL_Piece (TBL_Coord coord, int color_) {
		Init();
		color = color_;
		pos = new TBL_Coord(coord);
	}

	private void Init () {
		status = 0;
		count = 0;
		isFinished = false;
	}

	public boolean GetIsFinished () {return isFinished;}
	public void SetIsFinished (boolean b) {isFinished = b;}

	public TBL_Coord GetPos () {return pos;}
	public void SetPos (TBL_Coord coord) {pos = coord;}
	public void SetPos (int x_, int y_) {pos.SetXY(x_, y_);}

	public int GetColor () {return color;}
	public void SetColor (int col) {color = col;}

	public int GetStatus () {return status;}
	public void SetStatus (int status_) {status = status_;}

	public int GetCount () {return count;}
```

```
    public void SetCount (int n) {count = n;}
    public void IncrementCount () {count++;}
    public void ResetCount () {count = 0;}

    public boolean equals (TBL_Piece o) {
        if(this.GetPos().equals(o.GetPos()) && this.color == o.color &&
        this.status == o.status && this.count == o.count) return true;
        else return false;
    }
}
```

“TBL_Coord.java”

```
package board;

public class TBL_Coord {
	private int x, y;

	public TBL_Coord () {
		x = 0;
		y = 0;
	}

	public TBL_Coord (int a, int b) {
		x = a;
		y = b;
	}

	public TBL_Coord (TBL_Coord coord) {
		x = coord.x;
		y = coord.y;
	}

	public int GetX () {return x;}
	public int GetY () {return y;}
	public void SetX (int n) {x = n;}
	public void SetY (int n) {y = n;}
	public void SetXY (int a, int b) {x = a; y = b;}

	public boolean equals (TBL_Coord o) {
		if(this.x == o.x && this.y == o.y) return true;
		else return false;
	}
}
```

“TBL_Board.java”

```
package board;

import game.TBL_Player;
import pieces.TBL_Dice;
import pieces.TBL_Piece;

public class TBL_Board {
	private int[][] situation;
	private TBL_Dice dice;

	public TBL_Board () {
		Init();
	}

	private void Init () {
		dice = new TBL_Dice();
		situation = new int[15][15];
		// initialise le tableau à 0
		for(int i = 0 ; i < 15 ; i++)
			for(int j = 0 ; j < 15 ; j++) situation[i][j] = 0;
	}

	// affiche situation (pour les tests)
	@Deprecated
	public void DisplaySituation () {
		for(int i = 0 ; i < 15 ; i++) {
			for(int j = 0 ; j < 15 ; j++) {
				System.out.print(situation[i][j] + " | ");
			}
			System.out.println();
		}
	}

	// affiche le board et les pièces
	public void DisplayBoard (TBL_Player[] p, int turn) {
		// 0
		System.out.print("\t\t\t");
		Space(3); PrintR(1, 0, p);
		Space(3); PrintR(3, 0, p);
		Space(3); PrintR(5, 0, p);
		Space(3); PrintX(7, 0, p);
		Space(3); PrintR(9, 0, p);
		Space(3); PrintR(11, 0, p);
		Space(3); PrintR(13, 0, p);
		Space(3);
		System.out.println();
```

```
// 1
System.out.print("\t\t\t");
PrintR(0, 1, p); Space(29); PrintR(14, 1, p);
System.out.println();
// 2
System.out.print("\t\t\t");
Space(13);
PrintH(5, 2, p, 3); PrintH(6, 2, p, 3); PrintH(7, 2, p, 3); PrintH(8, 2, p, 3);
Space(13);
System.out.println();
// 3
System.out.print("\t\t\t");
PrintR(0, 3, p); Space(6);
PrintF(3, 3, p, 2); Space(15);
PrintF(11, 3, p, 3); Space(6);
PrintR(14, 3, p);
System.out.println();
// 4
System.out.print("\t\t\t");
Space(9); PrintF(4, 4, p, 2);
Space(11); PrintF(10, 4, p, 3);
Space(7);
System.out.println();
// 5
System.out.print("\t\t\t");
PrintR(0, 5, p); Space(10);
PrintF(5, 5, p, 2); Space(7);
PrintF(9, 5, p, 3); Space(10);
PrintR(14, 5, p);
System.out.println();
// 6
System.out.print("\t\t\t");
Space(5); PrintH(2, 6, p, 2);
Space(7); PrintF(6, 6, p, 2);
Space(3); PrintF(8, 6, p, 3);
Space(7); PrintH(12, 6, p, 4);
Space(3);
System.out.println();
// 7
System.out.print("\t\t\t");
PrintX(0, 7, p); Space(4); PrintH(2, 7, p, 2);
Space(9); PrintD(turn); Space(9);
PrintH(12, 7, p, 4); Space(4); PrintX(14, 7, p);
System.out.println();
// 8
System.out.print("\t\t\t");
Space(5); PrintH(2, 8, p, 2);
```

```
        Space(7); PrintF(6, 8, p, 1);
        Space(3); PrintF(8, 8, p, 4);
        Space(7); PrintH(12, 8, p, 4);
        Space(5);
        System.out.println();
        // 9
        System.out.print("\t\t\t");
        PrintR(0, 9, p); Space(4); PrintH(2, 9, p, 2); Space(5);
        PrintF(5, 9, p, 1); Space(7); PrintF(9, 9, p, 4);
        Space(5); PrintH(12, 9, p, 4); Space(4); PrintR(14, 9, p);
        System.out.println();
        // 10
        System.out.print("\t\t\t");
        Space(9); PrintF(4, 10, p, 1);
        Space(11); PrintF(10, 10, p, 4);
        Space(16);
        System.out.println();
        // 11
        System.out.print("\t\t\t");
        PrintR(0, 11, p); Space(6);
        PrintF(3, 11, p, 1); Space(15);
        PrintF(11, 11, p, 4); Space(6);
        PrintR(14, 11, p);
        System.out.println();
        // 12
        System.out.print("\t\t\t");
        Space(13);
        PrintH(5, 12, p, 1); PrintH(6, 12, p, 1); PrintH(7, 12, p, 1); PrintH(8, 12, p, 1);
        Space(13);
        System.out.println();
        // 13
        System.out.print("\t\t\t");
        PrintR(0, 13, p); Space(29); PrintR(14, 13, p);
        System.out.println();
        // 14
        System.out.print("\t\t\t");
        Space(3); PrintR(1, 14, p);
        Space(3); PrintR(3, 14, p);
        Space(3); PrintR(5, 14, p);
        Space(3); PrintX(7, 14, p);
        Space(3); PrintR(9, 14, p);
        Space(3); PrintR(11, 14, p);
        Space(3); PrintR(13, 14, p);
        Space(3);
        System.out.println();
    }
```

```
// utilitaire de l'affichage RUNNING
private void PrintR (int x, int y, TBL_Player[] p) {
	TBL_Coord coord = new TBL_Coord(x, y);
	int n = GetParams(coord);
	int col = ReadColor(n);
	int stat = ReadStatus(n);

	if(n == 0) System.out.print("\033[1;90m*\033[0m");
	else if(stat == 1) {
	switch(col) {
		// rouge
		case 1:	if(p[0].GetPieceByCoord(coord).GetPos().equals(coord)) {
				if(p[0].GetPieceByCoord(coord).equals(p[0].GetPieceByNumber(1)))
					System.out.print("\033[1;31m1\033[0m");
				else if(p[0].GetPieceByCoord(coord).equals(p[0].GetPieceByNumber(2)))
					System.out.print("\033[1;31m2\033[0m");
				else if(p[0].GetPieceByCoord(coord).equals(p[0].GetPieceByNumber(3)))
					System.out.print("\033[1;31m3\033[0m");
				else if(p[0].GetPieceByCoord(coord).equals(p[0].GetPieceByNumber(4)))
					System.out.print("\033[1;31m4\033[0m");
			}
			else System.out.print("\033[1;37m*\033[0m");
			break;
		// vert
		case 2:	if(p[1].GetPieceByCoord(coord).GetPos().equals(coord)) {
				if(p[1].GetPieceByCoord(coord).equals(p[1].GetPieceByNumber(1)))
					System.out.print("\033[1;32m1\033[0m");
				else if(p[1].GetPieceByCoord(coord).equals(p[1].GetPieceByNumber(2)))
					System.out.print("\033[1;32m2\033[0m");
				else if(p[1].GetPieceByCoord(coord).equals(p[1].GetPieceByNumber(3)))
					System.out.print("\033[1;32m3\033[0m");
				else if(p[1].GetPieceByCoord(coord).equals(p[1].GetPieceByNumber(4)))
					System.out.print("\033[1;32m4\033[0m");
			}
			else System.out.print("\033[1;37m*\033[0m");
			break;
		// jaune
		case 3:	if(p[2].GetPieceByCoord(coord).GetPos().equals(coord)) {
				if(p[2].GetPieceByCoord(coord).equals(p[2].GetPieceByNumber(1)))
					System.out.print("\033[1;33m1\033[0m");
				else if(p[2].GetPieceByCoord(coord).equals(p[2].GetPieceByNumber(2)))
					System.out.print("\033[1;33m2\033[0m");
				else if(p[2].GetPieceByCoord(coord).equals(p[2].GetPieceByNumber(3)))
					System.out.print("\033[1;33m3\033[0m");
				else if(p[2].GetPieceByCoord(coord).equals(p[2].GetPieceByNumber(4)))
					System.out.print("\033[1;33m4\033[0m");
			}
```

```
                    else System.out.print("\033[1;37m*\033[0m");
                    break;
                // bleu
                case 4:   if(p[3].GetPieceByCoord(coord).GetPos().equals(coord)) {
                            if(p[3].GetPieceByCoord(coord).equals(p[3].GetPieceByNumber(1)))
                                System.out.print("\033[1;34m1\033[0m");
                            else if(p[3].GetPieceByCoord(coord).equals(p[3].GetPieceByNumber(2)))
                                System.out.print("\033[1;34m2\033[0m");
                            else if(p[3].GetPieceByCoord(coord).equals(p[3].GetPieceByNumber(3)))
                                System.out.print("\033[1;34m3\033[0m");
                            else if(p[3].GetPieceByCoord(coord).equals(p[3].GetPieceByNumber(4)))
                                System.out.print("\033[1;34m4\033[0m");
                        }
                        else System.out.print("\033[1;37m*\033[0m");
                        break;
                default:  System.out.print("\033[1;37m*\033[0m");
            }
        }
    }

    // utilitaire de l'affichage HOME
    private void PrintH (int x, int y, TBL_Player[] p, int color_) {
        TBL_Coord coord = new TBL_Coord(x, y);
        int n = GetParams(coord);
        int col = ReadColor(n);
        int stat = ReadStatus(n);

        if(n == 0) {
            if(color_ == 1) System.out.print("\033[1;31m*\033[0m");
            else if(color_ == 2) System.out.print("\033[1;32m*\033[0m");
            else if(color_ == 3) System.out.print("\033[1;33m*\033[0m");
            else if(color_ == 4) System.out.print("\033[1;34m*\033[0m");
        }
        else if(stat == 0) {
            switch(col) {
                // rouge
                case 1:   if(p[0].GetPieceByCoord(coord).GetPos().equals(coord)) {
                            if(p[0].GetPieceByCoord(coord).equals(p[0].GetPieceByNumber(1)))
                                System.out.print("\033[1;31m1\033[0m");
                            else if(p[0].GetPieceByCoord(coord).equals(p[0].GetPieceByNumber(2)))
                                System.out.print("\033[1;31m2\033[0m");
                            else if(p[0].GetPieceByCoord(coord).equals(p[0].GetPieceByNumber(3)))
                                System.out.print("\033[1;31m3\033[0m");
                            else if(p[0].GetPieceByCoord(coord).equals(p[0].GetPieceByNumber(4)))
                                System.out.print("\033[1;31m4\033[0m");
                        }
                        else System.out.print("\033[1;31m*\033[0m");
```

```
                    break;
                // vert
                case 2:   if(p[1].GetPieceByCoord(coord).GetPos().equals(coord)) {
                            if(p[1].GetPieceByCoord(coord).equals(p[1].GetPieceByNumber(1)))
                                System.out.print("\033[1;32m1\033[0m");
                            else if(p[1].GetPieceByCoord(coord).equals(p[1].GetPieceByNumber(2)))
                                System.out.print("\033[1;32m2\033[0m");
                            else if(p[1].GetPieceByCoord(coord).equals(p[1].GetPieceByNumber(3)))
                                System.out.print("\033[1;32m3\033[0m");
                            else if(p[1].GetPieceByCoord(coord).equals(p[1].GetPieceByNumber(4)))
                                System.out.print("\033[1;32m4\033[0m");
                        }
                        else System.out.print("\033[1;32m*\033[0m");
                        break;
                // jaune
                case 3: I f(p[2].GetPieceByCoord(coord).GetPos().equals(coord)) {
                            if(p[2].GetPieceByCoord(coord).equals(p[2].GetPieceByNumber(1)))
                                System.out.print("\033[1;33m1\033[0m");
                            else if(p[2].GetPieceByCoord(coord).equals(p[2].GetPieceByNumber(2)))
                                System.out.print("\033[1;33m2\033[0m");
                            else if(p[2].GetPieceByCoord(coord).equals(p[2].GetPieceByNumber(3)))
                                System.out.print("\033[1;33m3\033[0m");
                            else if(p[2].GetPieceByCoord(coord).equals(p[2].GetPieceByNumber(4)))
                                System.out.print("\033[1;33m4\033[0m");
                        }
                        else System.out.print("\033[1;33m*\033[0m");
                        break;
                // bleu
                case 4:   if(p[3].GetPieceByCoord(coord).GetPos().equals(coord)) {
                            if(p[3].GetPieceByCoord(coord).equals(p[3].GetPieceByNumber(1)))
                                System.out.print("\033[1;34m1\033[0m");
                            else if(p[3].GetPieceByCoord(coord).equals(p[3].GetPieceByNumber(2)))
                                System.out.print("\033[1;34m2\033[0m");
                            else if(p[3].GetPieceByCoord(coord).equals(p[3].GetPieceByNumber(3)))
                                System.out.print("\033[1;34m3\033[0m");
                            else if(p[3].GetPieceByCoord(coord).equals(p[3].GetPieceByNumber(4)))
                                System.out.print("\033[1;34m4\033[0m");
                        }
                        else System.out.print("\033[1;34m*\033[0m");
                        break;
                default:  System.out.print("\033[1;37m*\033[0m");
            }
        }
    }
```

```
// utilitaire de l'affichage FINISH
private void PrintF (int x, int y, TBL_Player[] p, int color_) {
	TBL_Coord coord = new TBL_Coord(x, y);
	int n = GetParams(coord);
	int col = ReadColor(n);
	int stat = ReadStatus(n);

	if(n == 0) {
		if(color_ == 1) System.out.print("\033[1;31m*\033[0m");
		else if(color_ == 2) System.out.print("\033[1;32m*\033[0m");
		else if(color_ == 3) System.out.print("\033[1;33m*\033[0m");
		else if(color_ == 4) System.out.print("\033[1;34m*\033[0m");
	}
	else if(stat == 7) {
		switch(col) {
			// rouge
			case 1:	if(p[0].GetPieceByCoord(coord).GetPos().equals(coord)) {
					if(p[0].GetPieceByCoord(coord).equals(p[0].GetPieceByNumber(1)))
						System.out.print("\033[1;31m1\033[0m");
					else if(p[0].GetPieceByCoord(coord).equals(p[0].GetPieceByNumber(2)))
						System.out.print("\033[1;31m2\033[0m");
					else if(p[0].GetPieceByCoord(coord).equals(p[0].GetPieceByNumber(3)))
						System.out.print("\033[1;31m3\033[0m");
					else if(p[0].GetPieceByCoord(coord).equals(p[0].GetPieceByNumber(4)))
						System.out.print("\033[1;31m4\033[0m");
				}
				else System.out.print("\033[1;31m*\033[0m");
				break;
			// vert
			case 2:	if(p[1].GetPieceByCoord(coord).GetPos().equals(coord)) {
					if(p[1].GetPieceByCoord(coord).equals(p[1].GetPieceByNumber(1)))
						System.out.print("\033[1;32m1\033[0m");
					else if(p[1].GetPieceByCoord(coord).equals(p[1].GetPieceByNumber(2)))
						System.out.print("\033[1;32m2\033[0m");
					else if(p[1].GetPieceByCoord(coord).equals(p[1].GetPieceByNumber(3)))
						System.out.print("\033[1;32m3\033[0m");
					else if(p[1].GetPieceByCoord(coord).equals(p[1].GetPieceByNumber(4)))
						System.out.print("\033[1;32m4\033[0m");
				}
				else System.out.print("\033[1;32m*\033[0m");
				break;
			// jaune
			case 3:	if(p[2].GetPieceByCoord(coord).GetPos().equals(coord)) {
					if(p[2].GetPieceByCoord(coord).equals(p[2].GetPieceByNumber(1)))
						System.out.print("\033[1;33m1\033[0m");
					else if(p[2].GetPieceByCoord(coord).equals(p[2].GetPieceByNumber(2)))
						System.out.print("\033[1;33m2\033[0m");
```

```
                        else if(p[2].GetPieceByCoord(coord).equals(p[2].GetPieceByNumber(3)))
                            System.out.print("\033[1;33m3\033[0m");
                        else if(p[2].GetPieceByCoord(coord).equals(p[2].GetPieceByNumber(4)))
                            System.out.print("\033[1;33m4\033[0m");
                    }
                    else System.out.print("\033[1;33m*\033[0m");
                    break;
                // bleu
                case 4:  if(p[3].GetPieceByCoord(coord).GetPos().equals(coord)) {
                        if(p[3].GetPieceByCoord(coord).equals(p[3].GetPieceByNumber(1)))
                            System.out.print("\033[1;34m1\033[0m");
                        else if(p[3].GetPieceByCoord(coord).equals(p[3].GetPieceByNumber(2)))
                            System.out.print("\033[1;34m2\033[0m");
                        else if(p[3].GetPieceByCoord(coord).equals(p[3].GetPieceByNumber(3)))
                            System.out.print("\033[1;34m3\033[0m");
                        else if(p[3].GetPieceByCoord(coord).equals(p[3].GetPieceByNumber(4)))
                            System.out.print("\033[1;34m4\033[0m");
                    }
                    else System.out.print("\033[1;34m*\033[0m");
                    break;
                default:  System.out.print("\033[1;37m*\033[0m");
            }
        }
    }

    // utilitaire de l'affichage case XX
    private void PrintX (int x, int y, TBL_Player[] p) {
        TBL_Coord coord = new TBL_Coord(x, y);
        int n = GetParams(coord);
        int col = ReadColor(n);
        int stat = ReadStatus(n);

        if(n == 0) System.out.print("\033[1;95m*\033[0m");
        else if(stat == 1) {
            switch(col) {
                // rouge
                case 1:  if(p[0].GetPieceByCoord(coord).GetPos().equals(coord)) {
                        if(p[0].GetPieceByCoord(coord).equals(p[0].GetPieceByNumber(1)))
                            System.out.print("\033[1;31m1\033[0m");
                        else if(p[0].GetPieceByCoord(coord).equals(p[0].GetPieceByNumber(2)))
                            System.out.print("\033[1;31m2\033[0m");
                        else if(p[0].GetPieceByCoord(coord).equals(p[0].GetPieceByNumber(3)))
                            System.out.print("\033[1;31m3\033[0m");
                        else if(p[0].GetPieceByCoord(coord).equals(p[0].GetPieceByNumber(4)))
                            System.out.print("\033[1;31m4\033[0m");
                    }
                    else System.out.print("\033[1;95m*\033[0m");
```

```
                    break;
                // vert
                case 2:   if(p[1].GetPieceByCoord(coord).GetPos().equals(coord)) {
                            if(p[1].GetPieceByCoord(coord).equals(p[1].GetPieceByNumber(1)))
                                System.out.print("\033[1;32m1\033[0m");
                            else if(p[1].GetPieceByCoord(coord).equals(p[1].GetPieceByNumber(2)))
                                System.out.print("\033[1;32m2\033[0m");
                            else if(p[1].GetPieceByCoord(coord).equals(p[1].GetPieceByNumber(3)))
                                System.out.print("\033[1;32m3\033[0m");
                            else if(p[1].GetPieceByCoord(coord).equals(p[1].GetPieceByNumber(4)))
                                System.out.print("\033[1;32m4\033[0m");
                        }
                        else System.out.print("\033[1;95m*\033[0m");
                        break;
                // jaune
                case 3:   if(p[2].GetPieceByCoord(coord).GetPos().equals(coord)) {
                            if(p[2].GetPieceByCoord(coord).equals(p[2].GetPieceByNumber(1)))
                                System.out.print("\033[1;33m1\033[0m");
                            else if(p[2].GetPieceByCoord(coord).equals(p[2].GetPieceByNumber(2)))
                                System.out.print("\033[1;33m2\033[0m");
                            else if(p[2].GetPieceByCoord(coord).equals(p[2].GetPieceByNumber(3)))
                                System.out.print("\033[1;33m3\033[0m");
                            else if(p[2].GetPieceByCoord(coord).equals(p[2].GetPieceByNumber(4)))
                                System.out.print("\033[1;33m4\033[0m");
                        }
                        else System.out.print("\033[195m*\033[0m");
                        break;
                // bleu
                case 4:   if(p[3].GetPieceByCoord(coord).GetPos().equals(coord)) {
                            if(p[3].GetPieceByCoord(coord).equals(p[3].GetPieceByNumber(1)))
                                System.out.print("\033[1;34m1\033[0m");
                            else if(p[3].GetPieceByCoord(coord).equals(p[3].GetPieceByNumber(2)))
                                System.out.print("\033[1;34m2\033[0m");
                            else if(p[3].GetPieceByCoord(coord).equals(p[3].GetPieceByNumber(3)))
                                System.out.print("\033[1;34m3\033[0m");
                            else if(p[3].GetPieceByCoord(coord).equals(p[3].GetPieceByNumber(4)))
                                System.out.print("\033[1;34m4\033[0m");
                        }
                        else System.out.print("\033[1;95m*\033[0m");
                        break;
                default:  System.out.print("\033[1;37m*\033[0m");
            }
        }
    }
```

```
// utilitaire de l'affichage, insertion d'espaces
private void Space (int spaces) {for(int i = 1 ; i <= spaces ; i++) System.out.print(" ");}

// utilitaire de l'affichage DICE
private void PrintD (int turn) {
	switch(turn) {
		case 1:	System.out.print("\033[1;5;40;31m" + (situation[7][7] = dice.GetDiceResult()) + "\033[25;0m");
			break;
		case 2:	System.out.print("\033[1;5;40;32m" + (situation[7][7] = dice.GetDiceResult()) + "\033[25;0m");
			break;
		case 3:	System.out.print("\033[1;5;40;33m" + (situation[7][7] = dice.GetDiceResult()) + "\033[25;0m");
			break;
		case 4:	System.out.print("\033[1;5;40;34m" + (situation[7][7] = dice.GetDiceResult()) + "\033[25;0m");
			break;
		default:	System.out.println("Erreur dans TBL_Board.PrintD()");
	}
}

// retourne un nombre aléatoire entre 1 et 6 inclusivement
public int RollNGetDiceResult () {
	int n = dice.RollDice();
	UpdateDice(n);
	return n;
}

// update sitution pour la position du dé
public void UpdateDice (int n) {situation[7][7] = n;}

// update situation (cas général)
public void UpdateSituation (TBL_Coord coord, int player, int status) {
	situation[coord.GetY()][coord.GetX()] = ConvertParamsToInt(player, status);
}

// met à zéro une coordonnée
public void ZeroCoord (TBL_Coord coord) {situation[coord.GetY()][coord.GetX()] = 0;}
// retourne la conversion (joueur, statut) en int
private int ConvertParamsToInt (int player, int status) {return (player * 10) + status;}

// retourne le paramètre de situation à une coordonnée donnée
public int GetParams (TBL_Coord coord) {return situation[coord.GetY()][coord.GetX()];}

// retourne couleur à partir du paramètre de situation
public int ReadColor (int n) {return (n / 10);}

// retourne le statut à partir du paramètre de situation
public int ReadStatus (int n) {return (n % 10);}
```

```
    // retourne une pièce selon les coordonnées
    public TBL_Piece GetPieceByCoord (TBL_Coord coord, TBL_Player[] p) {
        int n = GetParams(coord);
        int col = ReadColor(n);

        switch(col) {
            // rouge
            case 1:   if(p[0].GetPieceByCoord(coord).GetPos().equals(coord)) return p[0].GetPieceByCoord(coord);
            // vert
            case 2:   if(p[1].GetPieceByCoord(coord).GetPos().equals(coord)) return p[1].GetPieceByCoord(coord);
            // jaune
            case 3:   if(p[2].GetPieceByCoord(coord).GetPos().equals(coord)) return p[2].GetPieceByCoord(coord);
            // bleu
            case 4:   if(p[3].GetPieceByCoord(coord).GetPos().equals(coord)) return p[3].GetPieceByCoord(coord);
            default:  return new TBL_Piece(new TBL_Coord(-1, -1), 0);
        }
    }
}
```

“TBL_Game.java”

```
package game;

import java.io.IOException;
import board.TBL_Board;
import board.TBL_Coord;
import pieces.TBL_Piece;

import java.util.Scanner;

public class TBL_Game {
	private TBL_Setup setup;
	private TBL_Player[] p;
	private TBL_Board b;
	private Scanner scan;
	private int turn;
	private int[] podium;

	public TBL_Game () {
I		nit();
	}

	private void Init () {
		scan = new Scanner(System.in);
		Clear();
		setup = new TBL_Setup();
		p = setup.GetPlayers();
		b = setup.GetBoard();
		turn = setup.GetFirstToPlay();
		podium = new int[4];
		for(int i = 0 ; i < 4 ; i++) podium[i] = 0;
		Clear();
	}

	public void PlayTurn () {
		TBL_Coord special = new TBL_Coord();
		TBL_Player now = GetPlayer();
		int opt = 0;
		if(now.GetIsActive() && GetPodiumCount() < 3) {
			Clear();
			b.DisplayBoard(p, turn); //b.DisplaySituation();
			// roulement de dé HUMAN / AUTO
			if(now.GetIsHuman()) now.SetLastDice(Roll(turn));
			else { // si AUTO
				now.SetLastDice(b.RollNGetDiceResult());
			}
			if(now.GetLastDice() == 6) { // dé == 6
```

```
                opt = 0;
                if(GetCountRUNNING(now) > 0) { // reste au moins une pièce RUNNING
                    opt = MenuHomeOrRunning(now);
                }
                else { // aucune pièce RUNNING
                    opt = 1;
                }
                if(opt == 1) {
                    if(GetCountHOME(now) > 0) PlaceOnSTART(now); //placer sur start
                }
                else { // avancer (demander pièce etc)
                    special = SubTurnGotRUNNING(now);
                }
            }
            else { // dé != 6
                if(GetCountRUNNING(now) > 0) { // reste au moins une pièce RUNNING
                    special = SubTurnGotRUNNING(now);
                }
                else { // aucune pièce RUNNING
                    if(GetCountHOME(now) == 0) { //GAME OVER - INSERER DANS PALMARES tant qu'il reste des joueurs
                        GameOver(now.GetColor()); // TEMPORAIRE!!!!!!!!!!!!!!!!!!!!
                    }
                }
            }

            // fin de tour sauf sur un 6
            if(!IsXX(special)) {
                if(GetPlayer().GetLastDice() != 6) EndTurn();
            }
        }
        else {
            if(GetPodiumCount() >= 3) {
                Clear();
                b.DisplayBoard(p, turn);
                DisplayPodium();
                System.exit(0); // exit
            }
            else EndTurn();
        }
    }
```

```
// sous-procédure commune si des pièces RUNNING existent
private TBL_Coord SubTurnGotRUNNING (TBL_Player p_) {
	TBL_Coord tempCoord = new TBL_Coord();
	TBL_Piece temp = new TBL_Piece();
	TBL_Player now = GetPlayer();
	TBL_Piece[] pRSet = GetPlayerPiecesRUNNING(now);
	int choice = 0;

	if(GetCountRUNNING(p_) > 1){ // plus d'une pièce
		do {
			choice = SubTurnChoice(p_);
			switch(choice) {
				case 1:	if(!p_.GetPieceByNumber(1).GetIsFinished()) temp = p_.GetPieceByNumber(1);
					break;
				case 2:	if(!p_.GetPieceByNumber(2).GetIsFinished()) temp = p_.GetPieceByNumber(2);
					break;
				case 3:	if(!p_.GetPieceByNumber(3).GetIsFinished()) temp = p_.GetPieceByNumber(3);
					break;
				case 4:	if(!p_.GetPieceByNumber(4).GetIsFinished()) temp = p_.GetPieceByNumber(4);
					break;
				default:	System.out.println("Erreur dans TBL_Game.SubTurnGotRUNNING");
			}
			tempCoord = ComputeDestination(temp, p_.GetLastDice());
		}
		while(IsFriendlyOnCoord(tempCoord));
		// ici temp est la pièce à jouer, contient ancienne position
		if(temp.GetCount() < 23) { // déplacement standard
			tempCoord = SubMoveUpdate(p_, temp);
		}
		else if(temp.GetCount() >= 23 && p_.GetLastDice() < (29 - temp.GetCount())) {
			tempCoord = SubMoveUpdate(p_, temp);
		}
		else { // mode "FINISH"
			FinishMode(p_, temp);
		}
	}
	else { // seulement une pièce RUNNING
		//set temp
		int i = 0;
		do {
			if(pRSet[i].equals(p_.GetPieceByNumber(1))) temp = pRSet[i];
			else if(pRSet[i].equals(p_.GetPieceByNumber(2))) temp = pRSet[i];
			else if(pRSet[i].equals(p_.GetPieceByNumber(3))) temp = pRSet[i];
			else if(pRSet[i].equals(p_.GetPieceByNumber(4))) temp = pRSet[i];
			i++;
		}
		while(temp != null && i < 4);
```

```
            tempCoord = ComputeDestination(temp, p_.GetLastDice()); // pour le tour de table seulement (test case XX)
            // ici temp est la pièce à jouer, contient ancienne position
            if(temp.GetCount() < 23) { // déplacement standard
                tempCoord = SubMoveUpdate(p_, temp);
            }
            else if(temp.GetCount() >= 23 && p_.GetLastDice() < (29 - temp.GetCount())) {
                tempCoord = SubMoveUpdate(p_, temp);
            }
            else { // mode "FINISH"
                FinishMode(p_, temp);
            }
        }
        return tempCoord;
    }

    // update position du déplacement
    private TBL_Coord SubMoveUpdate (TBL_Player p_, TBL_Piece o) {
        TBL_Coord coord = ComputeDestination(o, p_.GetLastDice()); // coord est la destination
        if(!IsEmptySlot(coord)) { // slot occupée par pièce (adversaire ou amie)
            int n = GetOccupied(coord).GetColor();

            if(n == 1) {if(n != turn) GoHome(p[0], GetOccupied(coord));}
            else if(n == 2) {if(n != turn) GoHome(p[1], GetOccupied(coord));}
            else if(n == 3) {if(n != turn) GoHome(p[2], GetOccupied(coord));}
            else if(n == 4) {if(n != turn) GoHome(p[3], GetOccupied(coord));}
            else {
                System.out.println("Erreur dans TBL_Game.SubMoveUpdate - conditions GoHome");
                scan.nextLine();
            }
        }
        // avancer (update)
        b.ZeroCoord(o.GetPos());
        o.SetPos(coord);
        b.UpdateSituation(coord, GetPlayerNumber(p_), 1);
        o.SetCount(o.GetCount() + p_.GetLastDice());
        return o.GetPos();
    }
```

```
// exécute le "FINISH" mode
private void FinishMode (TBL_Player p_, TBL_Piece o) {
	int n = 29 - o.GetCount();
	int m = p_.GetLastDice() - n;

	switch(p_.GetColor()) {
		case 1:	if(m == 0 && b.GetParams(new TBL_Coord(3, 11)) == 0) {
				UpdateFinishSpot(p_, 1, 0);
				SubFinishUpdate(p_, o, new TBL_Coord(3, 11));
			}
			else if(m == 1 && b.GetParams(new TBL_Coord(4, 10)) == 0) {
				UpdateFinishSpot(p_, 1, 1);
				SubFinishUpdate(p_, o, new TBL_Coord(4, 10));
			}
			else if(m == 2 && b.GetParams(new TBL_Coord(5, 9)) == 0) {
				UpdateFinishSpot(p_, 1, 2);
				SubFinishUpdate(p_, o, new TBL_Coord(5, 9));
			}
			else if(m == 3 && b.GetParams(new TBL_Coord(6, 8)) == 0) {
				UpdateFinishSpot(p_, 1, 3);
				SubFinishUpdate(p_, o, new TBL_Coord(6, 8));
			}
			break;
		case 2:	if(m == 0 && b.GetParams(new TBL_Coord(3, 3)) == 0) {
				UpdateFinishSpot(p_, 1, 0);
				SubFinishUpdate(p_, o, new TBL_Coord(3, 3));
			}
			else if(m == 1 && b.GetParams(new TBL_Coord(4, 4)) == 0) {
				UpdateFinishSpot(p_, 1, 1);
				SubFinishUpdate(p_, o, new TBL_Coord(4, 4));
			}
			else if(m == 2 && b.GetParams(new TBL_Coord(5, 5)) == 0) {
				UpdateFinishSpot(p_, 1, 2);
				SubFinishUpdate(p_, o, new TBL_Coord(5, 5));
			}
			else if(m == 3 && b.GetParams(new TBL_Coord(6, 6)) == 0) {
				UpdateFinishSpot(p_, 1, 3);
				SubFinishUpdate(p_, o, new TBL_Coord(6, 6));
			}
			break;
		case 3:	if(m == 0 && b.GetParams(new TBL_Coord(11, 3)) == 0) {
				UpdateFinishSpot(p_, 1, 0);
				SubFinishUpdate(p_, o, new TBL_Coord(11, 3));
			}
			else if(m == 1 && b.GetParams(new TBL_Coord(10, 4)) == 0) {
				UpdateFinishSpot(p_, 1, 1);
				SubFinishUpdate(p_, o, new TBL_Coord(10, 4));
```

```
                }
                else if(m == 2 && b.GetParams(new TBL_Coord(9, 5)) == 0) {
                    UpdateFinishSpot(p_, 1, 2);
                    SubFinishUpdate(p_, o, new TBL_Coord(9, 5));
                }
                else if(m == 3 && b.GetParams(new TBL_Coord(8, 6)) == 0) {
                    UpdateFinishSpot(p_, 1, 3);
                    SubFinishUpdate(p_, o, new TBL_Coord(8, 6));
                }
                break;
            case 4:  if(m == 0 && b.GetParams(new TBL_Coord(11, 11)) == 0) {
                    UpdateFinishSpot(p_, 1, 0);
                    SubFinishUpdate(p_, o, new TBL_Coord(11, 11));
                }
                else if(m == 1 && b.GetParams(new TBL_Coord(10, 10)) == 0) {
                    UpdateFinishSpot(p_, 1, 1);
                    SubFinishUpdate(p_, o, new TBL_Coord(10, 10));
                }
                else if(m == 2 && b.GetParams(new TBL_Coord(9, 9)) == 0) {
                    UpdateFinishSpot(p_, 1, 2);
                    SubFinishUpdate(p_, o, new TBL_Coord(9, 9));
                }
                else if(m == 3 && b.GetParams(new TBL_Coord(8, 8)) == 0) {
                    UpdateFinishSpot(p_, 1, 3);
                    SubFinishUpdate(p_, o, new TBL_Coord(8, 8));
                }
                break;
            default:  SubMoveUpdate(p_, o);
        }

        if(CountFINISH(p_) == 4) { // OVER - INSÉRER DANS PALMARÈS
            p_.SetIsActive(false);
            GameOver(p_.GetColor());
        }
    }

    // update position pour un "FINISH"
    private void SubFinishUpdate (TBL_Player p_, TBL_Piece o, TBL_Coord coord) {
        b.ZeroCoord(o.GetPos());
        o.SetPos(coord);
        o.SetIsFinished(true);
        o.SetStatus(7);
        b.UpdateSituation(coord, GetPlayerNumber(p_), 7);
        b.DisplayBoard(p, turn);
    }
```

```
// retourne le numéro de la pièce choisie à jouer
private int SubTurnChoice (TBL_Player p_) {
	int choice = 0;
	TBL_Piece temp = new TBL_Piece();

	if(p_.GetIsHuman()) {
		do {
			System.out.print("\n\t\t\t\033[1;37m # Pièce à jouer : \033[1;33m");
			choice = scan.nextInt();
			scan.nextLine();
			System.out.print("\033[0m");
			if(choice == 1) temp = p_.GetPieceByNumber(1);
			else if(choice ==2) temp = p_.GetPieceByNumber(2);
			else if(choice ==3) temp = p_.GetPieceByNumber(3);
			else if(choice ==4) temp = p_.GetPieceByNumber(4);
		}
		// vérifie si pièce choisie n'est pas "FINISHED", est RUNNING ou si elle peut "finir"
		while(temp.GetIsFinished() || temp.GetStatus() != 1 && !CanFinish(p_, choice));
	}
	else { // si AUTO
		int res = 1;
		int[] tab = new int[4];
		for(int i = 0 ; i < 4 ; i++) {
			if(p_.GetPieceByNumber(i+1).GetStatus() == 1) tab[i] = 1;
		}
		res = RandOneTo_N(4);
		while(tab[res-1] != 1 && !IsFriendlyOnCoord(ComputeDestination(p_.GetPieceByNumber(res),
		p_.GetLastDice()))) res = RandOneTo_N(4);
		return res;
	}
	return choice;
}

// place pièce sur START
private void PlaceOnSTART (TBL_Player p_) {
	TBL_Piece temp;

	switch(p_.GetColor()) {
		// rouge
		case 1:	temp = PickFromHOME(p_);
				// vérifie si adversaire sur coordonnées
				if(!IsEmptySlot(new TBL_Coord(1, 14))){ // slot occupée
					// retourner pièce, l'envoyer à sa base
					GoHome(p_, GetOccupied(new TBL_Coord(1, 14)));
				}
				// updater situation pour pièce courante
				temp.IncrementCount();
```

```
                temp.SetStatus(1);
                temp.SetPos(1, 14);
                b.UpdateSituation(temp.GetPos(), 1, 1);
                break;
            case 2:  temp = PickFromHOME(p_);
                // vérifie si adversaire sur coordonnées
                if(!IsEmptySlot(new TBL_Coord(0, 1))){ // slot occupée
                    // retourner pièce, l'envoyer à sa base
                    GoHome(p_, GetOccupied(new TBL_Coord(0, 1)));
                }
                // updater situation pour pièce courante
                temp.IncrementCount();
                temp.SetStatus(1);
                temp.SetPos(0, 1);
                b.UpdateSituation(temp.GetPos(), 2, 1);
                break;
            case 3:  temp = PickFromHOME(p_);
                // vérifie si adversaire sur coordonnées
                if(!IsEmptySlot(new TBL_Coord(13, 0))){ // slot occupée
                    // retourner pièce, l'envoyer à sa base
                    GoHome(p_, GetOccupied(new TBL_Coord(13, 0)));
                }
                // updater situation pour pièce courante
                temp.IncrementCount();
                temp.SetStatus(1);
                temp.SetPos(13, 0);
                b.UpdateSituation(temp.GetPos(), 3, 1);
                break;
            case 4:  temp = PickFromHOME(p_);
                // vérifie si adversaire sur coordonnées
                if(!IsEmptySlot(new TBL_Coord(14, 13))){ // slot occupée
                    // retourner pièce, l'envoyer à sa base
                    GoHome(p_, GetOccupied(new TBL_Coord(14, 13)));
                }
                // updater situation pour pièce courante
                temp.IncrementCount();
                temp.SetStatus(1);
                temp.SetPos(14, 13);
                b.UpdateSituation(temp.GetPos(), 4, 1);
                break;
            default: break;
        }
    }
```

```
// vérifie si une slot est libre
private boolean IsEmptySlot (TBL_Coord coord) {
	if(b.GetParams(coord) == 0) return true;
	else return false;
}

// retourne pièce qui occupe une slot convoitée
private TBL_Piece GetOccupied (TBL_Coord coord) {return b.GetPieceByCoord(coord, p);}

// retourne pièce de HOME
private TBL_Piece PickFromHOME (TBL_Player p_) {
	switch(p_.GetColor()) {
		// rouge
		case 1:	if(p_.GetPieceByNumber(1).GetPos().equals(new TBL_Coord(5, 12))) {
				UpdateHomeSpot(p_, 0, 0);
				b.ZeroCoord(new TBL_Coord(5, 12));
				return p_.GetPieceByNumber(1);
			}
			else if(p_.GetPieceByNumber(2).GetPos().equals(new TBL_Coord(6, 12))) {
				UpdateHomeSpot(p_, 0, 1);
				b.ZeroCoord(new TBL_Coord(6, 12));
				return p_.GetPieceByNumber(2);
			}
			else if(p_.GetPieceByNumber(3).GetPos().equals(new TBL_Coord(7, 12))) {
				UpdateHomeSpot(p_, 0, 2);
				b.ZeroCoord(new TBL_Coord(7, 12));
				return p_.GetPieceByNumber(3);
			}
			else if(p_.GetPieceByNumber(4).GetPos().equals(new TBL_Coord(8, 12))) {
				UpdateHomeSpot(p_, 0, 3);
				b.ZeroCoord(new TBL_Coord(8, 12));
				return p_.GetPieceByNumber(4);
			}
			else return new TBL_Piece(new TBL_Coord(-1, -1), 0);
		case 2:	if(p_.GetPieceByNumber(1).GetPos().equals(new TBL_Coord(2, 6))) {
				UpdateHomeSpot(p_, 0, 0);
				b.ZeroCoord(new TBL_Coord(2, 6));
				return p_.GetPieceByNumber(1);
			}
			else if(p_.GetPieceByNumber(2).GetPos().equals(new TBL_Coord(2, 7))) {
				UpdateHomeSpot(p_, 0, 1);
				b.ZeroCoord(new TBL_Coord(2, 7));
				return p_.GetPieceByNumber(2);
			}
			else if(p_.GetPieceByNumber(3).GetPos().equals(new TBL_Coord(2, 8))) {
				UpdateHomeSpot(p_, 0, 2);
				b.ZeroCoord(new TBL_Coord(2, 8));
```

```
                return p_.GetPieceByNumber(3);
            }
            else if(p_.GetPieceByNumber(4).GetPos().equals(new TBL_Coord(2, 9))) {
                UpdateHomeSpot(p_, 0, 3);
                b.ZeroCoord(new TBL_Coord(2, 9));
                return p_.GetPieceByNumber(4);
            }
            else return new TBL_Piece(new TBL_Coord(-1, -1), 0);
        case 3:  if(p_.GetPieceByNumber(1).GetPos().equals(new TBL_Coord(8, 2))) {
                UpdateHomeSpot(p_, 0, 0);
                b.ZeroCoord(new TBL_Coord(8, 2));
                return p_.GetPieceByNumber(1);
            }
            else if(p_.GetPieceByNumber(2).GetPos().equals(new TBL_Coord(7, 2))) {
                UpdateHomeSpot(p_, 0, 1);
                b.ZeroCoord(new TBL_Coord(7, 2));
                return p_.GetPieceByNumber(2);
            }
            else if(p_.GetPieceByNumber(3).GetPos().equals(new TBL_Coord(6, 2))) {
                UpdateHomeSpot(p_, 0, 2);
                b.ZeroCoord(new TBL_Coord(6, 2));
                return p_.GetPieceByNumber(3);
            }
            else if(p_.GetPieceByNumber(4).GetPos().equals(new TBL_Coord(5, 2))) {
                UpdateHomeSpot(p_, 0, 3);
                b.ZeroCoord(new TBL_Coord(5, 2));
                return p_.GetPieceByNumber(4);
            }
            else return new TBL_Piece(new TBL_Coord(-1, -1), 0);
        case 4:  if(p_.GetPieceByNumber(1).GetPos().equals(new TBL_Coord(12, 9))) {
                UpdateHomeSpot(p_, 0, 0);
                b.ZeroCoord(new TBL_Coord(12, 9));
                return p_.GetPieceByNumber(1);
            }
            else if(p_.GetPieceByNumber(2).GetPos().equals(new TBL_Coord(12, 8))) {
                UpdateHomeSpot(p_, 0, 1);
                b.ZeroCoord(new TBL_Coord(12, 8));
                return p_.GetPieceByNumber(2);
            }
            else if(p_.GetPieceByNumber(3).GetPos().equals(new TBL_Coord(12, 7))) {
                UpdateHomeSpot(p_, 0, 2);
                b.ZeroCoord(new TBL_Coord(12, 7));
                return p_.GetPieceByNumber(3);
            }
            else if(p_.GetPieceByNumber(4).GetPos().equals(new TBL_Coord(12, 6))) {
                UpdateHomeSpot(p_, 0, 3);
                b.ZeroCoord(new TBL_Coord(12, 6));
```

```
                    return p_.GetPieceByNumber(4);
                }
                else return new TBL_Piece(new TBL_Coord(-1, -1), 0);
            default:  return new TBL_Piece(new TBL_Coord(-1, -1), 0);
        }
    }

    // update homeSpot
    private void UpdateHomeSpot (TBL_Player p_, int value, int index) {p_.SetHSpot(value, index);}

    // update finishSpot
    private void UpdateFinishSpot (TBL_Player p_, int value, int index) {p_.SetFSpot(value, index);}

    // place une pièce à HOME (player p_ est la couleur de la pièce o "mangée")
    private void GoHome (TBL_Player p_, TBL_Piece o) {
        switch(p_.GetColor()) {
            // rouge
            case 1:  if(p_.GetPieceByNumber(1).equals(o)) {
                        b.ZeroCoord(o.GetPos());
                        o.SetPos(5, 12);
                        o.SetCount(0);
                        o.SetStatus(0);
                        b.UpdateSituation(o.GetPos(), 1, 0);
                        UpdateHomeSpot(p_, 1, 0);
                    }
                    else if(p_.GetPieceByNumber(2).equals(o)) {
                        b.ZeroCoord(o.GetPos());
                        o.SetPos(6, 12);
                        o.SetCount(0);
                        o.SetStatus(0);
                        b.UpdateSituation(o.GetPos(), 1, 0);
                        UpdateHomeSpot(p_, 1, 1);
                    }
                    else if(p_.GetPieceByNumber(3).equals(o)) {
                        b.ZeroCoord(o.GetPos());
                        o.SetPos(7, 12);
                        o.SetCount(0);
                        o.SetStatus(0);
                        b.UpdateSituation(o.GetPos(), 1, 0);
                        UpdateHomeSpot(p_, 1, 2);
                    }
                    else if(p_.GetPieceByNumber(4).equals(o)) {
                        b.ZeroCoord(o.GetPos());
                        o.SetPos(8, 12);
                        o.SetCount(0);
                        o.SetStatus(0);
                        b.UpdateSituation(o.GetPos(), 1, 0);
```

```
                UpdateHomeSpot(p_, 1, 3);
            }
            else System.out.println("Erreur : TBL_Game.GoHome : pas une pièce rouge");
            break;
// vert
case 2:     if(p_.GetPieceByNumber(1).equals(o)) {
                b.ZeroCoord(o.GetPos());
                o.SetPos(2, 6);
                o.SetCount(0);
                o.SetStatus(0);
                b.UpdateSituation(o.GetPos(), 2, 0);
                UpdateHomeSpot(p_, 1, 0);
            }
            else if(p_.GetPieceByNumber(2).equals(o)) {
                b.ZeroCoord(o.GetPos());
                o.SetPos(2, 7);
                o.SetCount(0);
                o.SetStatus(0);
                b.UpdateSituation(o.GetPos(), 2, 0);
                UpdateHomeSpot(p_, 1, 1);
            }
            else if(p_.GetPieceByNumber(3).equals(o)) {
                b.ZeroCoord(o.GetPos());
                o.SetPos(2, 8);
                o.SetCount(0);
                o.SetStatus(0);
                b.UpdateSituation(o.GetPos(), 2, 0);
                UpdateHomeSpot(p_, 1, 2);
            }
            else if(p_.GetPieceByNumber(4).equals(o)) {
                b.ZeroCoord(o.GetPos());
                o.SetPos(2, 9);
                o.SetCount(0);
                o.SetStatus(0);
                b.UpdateSituation(o.GetPos(), 2, 0);
                UpdateHomeSpot(p_, 1, 3);
            }
            else System.out.println("Erreur : TBL_Game.GoHome : pas une pièce verte");
            break;
// jaune
case 3:     if(p_.GetPieceByNumber(1).equals(o)) {
                b.ZeroCoord(o.GetPos());
                o.SetPos(8, 2);
                o.SetCount(0);
                o.SetStatus(0);
                b.UpdateSituation(o.GetPos(), 3, 0);
                UpdateHomeSpot(p_, 1, 0);
```

```
            }
            else if(p_.GetPieceByNumber(2).equals(o)) {
                b.ZeroCoord(o.GetPos());
                o.SetPos(7, 2);
                o.SetCount(0);
                o.SetStatus(0);
                b.UpdateSituation(o.GetPos(), 3, 0);
                UpdateHomeSpot(p_, 1, 1);
            }
            else if(p_.GetPieceByNumber(3).equals(o)) {
                b.ZeroCoord(o.GetPos());
                o.SetPos(6, 2);
                o.SetCount(0);
                o.SetStatus(0);
                b.UpdateSituation(o.GetPos(), 3, 0);
                UpdateHomeSpot(p_, 1, 2);
            }
            else if(p_.GetPieceByNumber(4).equals(o)) {
                b.ZeroCoord(o.GetPos());
                o.SetPos(5, 2);
                o.SetCount(0);
                o.SetStatus(0);
                b.UpdateSituation(o.GetPos(), 3, 0);
                UpdateHomeSpot(p_, 1, 3);
            }
            else System.out.println("Erreur : TBL_Game.GoHome : pas une pièce jaune");
            break;
        // bleu
        case 4:  if(p_.GetPieceByNumber(1).equals(o)) {
                b.ZeroCoord(o.GetPos());
                o.SetPos(12, 9);
                o.SetCount(0);
                o.SetStatus(0);
                b.UpdateSituation(o.GetPos(), 4, 0);
                UpdateHomeSpot(p_, 1, 0);
            }
            else if(p_.GetPieceByNumber(2).equals(o)) {
                b.ZeroCoord(o.GetPos());
                o.SetPos(12, 8);
                o.SetCount(0);
                o.SetStatus(0);
                b.UpdateSituation(o.GetPos(), 4, 0);
                UpdateHomeSpot(p_, 1, 1);
            }
            else if(p_.GetPieceByNumber(3).equals(o)) {
                b.ZeroCoord(o.GetPos());
                o.SetPos(12, 7);
```

```
                        o.SetCount(0);
                        o.SetStatus(0);
                        b.UpdateSituation(o.GetPos(), 4, 0);
                        UpdateHomeSpot(p_, 1, 2);
                    }
                    else if(p_.GetPieceByNumber(4).equals(o)) {
                        b.ZeroCoord(o.GetPos());
                        o.SetPos(12, 6);
                        o.SetCount(0);
                        o.SetStatus(0);
                        b.UpdateSituation(o.GetPos(), 4, 0);
                        UpdateHomeSpot(p_, 1, 3);
                    }
                    else System.out.println("Erreur : TBL_Game.GoHome : pas une pièce bleue");
                    break;
            default:  System.out.println("Erreur dans TBL_Game.GoHome");
            scan.nextLine();
        }
    }

    // 1-HOME-->START & 2-avancer
    private int MenuHomeOrRunning (TBL_Player p_) {
        int n = 0;
        if(p_.GetIsHuman()) {
            do {
                System.out.println("\t\t\t\033[1;100;37mOptions :\033[0m");
                if(!FriendlyOnStart() && GetCountHOME(p_) > 0) System.out.println("\t\t\t \033[1;32m1. Sortir pièce de
                sa base\033[0m");
                else System.out.println("\t\t\t \033[1;31m** Sortie non disponible **\033[0m");
                System.out.println("\t\t\t \033[1;32m2. Avancer pièce\033[0m");
                System.out.print("\t\t\t\t\033[1;37m--> \033[1;33m");
                n = scan.nextInt();
                System.out.print("\033[0m");
                scan.nextLine();
            }
            while(n == 0 || (FriendlyOnStart() && n == 1));
        }
        else { // si AUTO
            if(FriendlyOnStart()) return 2;
            else {
                if(GetCountHOME(p_) > 0) return RandOneTo_N(2);
                else return 2;
            }
        }
        return n;
    }
```

```
// calcule random entre 1 et n inclusivement
private int RandOneTo_N (int n) {
	int result = ((int)(Math.random() * (double)n)) + 1;
	return result;
}

// valide si pièce de même couleur déjà sur START
private boolean FriendlyOnStart () {
	int n;
	switch(turn) {
		case 1:	n = b.GetParams(new TBL_Coord(1, 14));
			if(n != 0 && (n / 10) == 1) return true;
			break;
		case 2:	n = b.GetParams(new TBL_Coord(0, 1));
			if(n != 0 && (n / 10) == 2) return true;
			break;
		case 3:	n = b.GetParams(new TBL_Coord(13, 0));
			if(n != 0 && (n / 10) == 3) return true;
			break;
		case 4:	n = b.GetParams(new TBL_Coord(14, 13));
			if(n != 0 && (n / 10) == 4) return true;
			break;
		default:	return false;
	}
	return false;
}

// valide si pièce de même couleur sur coord
private boolean IsFriendlyOnCoord (TBL_Coord coord) {
	int n;
	switch(turn) {
		case 1:	n = b.GetParams(coord);
			if(n != 0 && (n / 10) == 1) return true;
			break;
		case 2:	n = b.GetParams(coord);
			if(n != 0 && (n / 10) == 2) return true;
			break;
		case 3:	n = b.GetParams(coord);
			if(n != 0 && (n / 10) == 3) return true;
			break;
		case 4:	n = b.GetParams(coord);
			if(n != 0 && (n / 10) == 4) return true;
			break;
		default:	return false;
	}
	return false;
}
```

```
// retourne toutes les pièces RUNNING du joueur courant
private TBL_Piece[] GetPlayerPiecesRUNNING (TBL_Player p_) {
	TBL_Piece[] setR = new TBL_Piece[4];
	if(p_.GetPieceByNumber(1).GetStatus() == 1 && // enlever prochaine condition pour avancer pièces dans FINISH
	!p_.GetPieceByNumber(1).GetIsFinished()) setR[0] = p_.GetPieceByNumber(1);
	else setR[0] = new TBL_Piece(new TBL_Coord(-1, -1), 0);
	if(p_.GetPieceByNumber(2).GetStatus() == 1 &&
	!p_.GetPieceByNumber(2).GetIsFinished()) setR[1] = p_.GetPieceByNumber(2);
	else setR[1] = new TBL_Piece(new TBL_Coord(-1, -1), 0);
	if(p_.GetPieceByNumber(3).GetStatus() == 1 &&
	!p_.GetPieceByNumber(3).GetIsFinished()) setR[2] = p_.GetPieceByNumber(3);
	else setR[2] = new TBL_Piece(new TBL_Coord(-1, -1), 0);
	if(p_.GetPieceByNumber(4).GetStatus() == 1 &&
	!p_.GetPieceByNumber(4).GetIsFinished()) setR[3] = p_.GetPieceByNumber(4);
	else setR[3] = new TBL_Piece(new TBL_Coord(-1, -1), 0);
	return setR;
}

// retourne nombre de pièces RUNNING du joueur courant
private int GetCountRUNNING (TBL_Player p_) {
	int count = 0;
	if(p_.GetPieceByNumber(1).GetStatus() == 1) count++;
	if(p_.GetPieceByNumber(2).GetStatus() == 1) count++;
	if(p_.GetPieceByNumber(3).GetStatus() == 1) count++;
	if(p_.GetPieceByNumber(4).GetStatus() == 1) count++;
	return count;
}

// retourne nombre de pièces HOME du joueur courant
private int GetCountHOME (TBL_Player p_) {
	int count = 0;
	if(p_.GetPieceByNumber(1).GetStatus() == 0) count++;
	if(p_.GetPieceByNumber(2).GetStatus() == 0) count++;
	if(p_.GetPieceByNumber(3).GetStatus() == 0) count++;
	if(p_.GetPieceByNumber(4).GetStatus() == 0) count++;
	return count;
}

// retourne le joueur selon le tour courant
private TBL_Player GetPlayer () {
	if(turn == 1) return p[0];
	else if(turn == 2) return p[1];
	else if(turn == 3) return p[2];
	else return p[3];
}
```

```
// retourne le numéro de joueur
private int GetPlayerNumber (TBL_Player p_) {
	if(p_.equals(p[0])) return 1;
	else if(p_.equals(p[1])) return 2;
	else if(p_.equals(p[2])) return 3;
	else if(p_.equals(p[3])) return 4;
	else return 0;
}

// gère le déplacement autour du board
private TBL_Coord ComputeDestination (TBL_Piece o, int dice) {
	int posX = o.GetPos().GetX();
	int posY = o.GetPos().GetY();
	TBL_Coord destCoord;
	int x = 0, y = 0;

	// segment coin inférieur gauche - coin supérieur gauche (A-B)
	if(posX == 0) {
		if(dice <= (posY - 1) / 2) { // destination sur même axe
			x = 0;
			y = posY - 2 * dice;
		}
		else { // destination sur axe adjacent
			int n = (posY - 1) / 2;
			int m = dice - n;
			x = 2 * m - 1;
			y = 0;
		}
	}
	// segment coin supérieur gauche - coin supérieur droit (B-C)
	if(posY == 0) {
		if(dice <= (13 - posX) / 2) { // destination sur même axe
			x = posX + 2 * dice;
			y = 0;
		}
		else { // destination sur axe adjacent
			int n = (13 - posX) / 2;
			int m = dice - n;
			x = 14;
			y = 2 * m - 1;
		}
	}
	// segment coin supérieur droit - coin inférieur droit (C-D)
	if(posX == 14) {
		if(dice <= (13 - posY) / 2) { // destination sur même axe
			x = 14;
			y = posY + 2 * dice;
```

```
            }
            else { // destination sur axe adjacent
                int n = (13 - posY) / 2;
                int m = dice - n;
                x = 13 - 2 * (m - 1);
                y = 14;
            }
        }
        // segment coin inférieur droit - coin inférieur gauche (D-A)
        if(posY == 14) {
            if(dice <= (posX - 1) / 2) { // destination sur même axe
                x = posX - 2 * dice;
                y = 14;
            }
            else { // destination sur axe adjacent
                int n = (posX - 1) / 2;
                int m = dice - n;
                x = 0;
                y = 13 - 2 * (m - 1);
            }
        }
        destCoord = new TBL_Coord(x, y);
        return destCoord;
        }

    // détermine si la pièce choisie peut se rendre dans une case "FINISH" par numéro
    private boolean CanFinish (TBL_Player p_, int pieceNum) {
        switch(p_.GetColor()) {
            case 1:   if(ComputeDestination(p_.GetPieceByNumber(1), p_.GetLastDice()).equals(new TBL_Coord(3, 11)) ||
                      ComputeDestination(p_.GetPieceByNumber(2), p_.GetLastDice()).equals(new TBL_Coord(4, 10)) ||
                      ComputeDestination(p_.GetPieceByNumber(3), p_.GetLastDice()).equals(new TBL_Coord(5, 9)) ||
                      ComputeDestination(p_.GetPieceByNumber(4), p_.GetLastDice()).equals(new TBL_Coord(6, 8)))
                          return true;
            case 2:   if(ComputeDestination(p_.GetPieceByNumber(1), p_.GetLastDice()).equals(new TBL_Coord(3, 3)) ||
                      ComputeDestination(p_.GetPieceByNumber(2), p_.GetLastDice()).equals(new TBL_Coord(4, 4)) ||
                      ComputeDestination(p_.GetPieceByNumber(3), p_.GetLastDice()).equals(new TBL_Coord(5, 5)) ||
                      ComputeDestination(p_.GetPieceByNumber(4), p_.GetLastDice()).equals(new TBL_Coord(6, 6)))
                          return true;
            case 3:   if(ComputeDestination(p_.GetPieceByNumber(1), p_.GetLastDice()).equals(new TBL_Coord(11, 3)) ||
                      ComputeDestination(p_.GetPieceByNumber(2), p_.GetLastDice()).equals(new TBL_Coord(10, 4)) ||
                      ComputeDestination(p_.GetPieceByNumber(3), p_.GetLastDice()).equals(new TBL_Coord(9, 5)) ||
                      ComputeDestination(p_.GetPieceByNumber(4), p_.GetLastDice()).equals(new TBL_Coord(8, 6)))
                          return true;
            case 4:   if(ComputeDestination(p_.GetPieceByNumber(1), p_.GetLastDice()).equals(new TBL_Coord(11, 11)) ||
                      ComputeDestination(p_.GetPieceByNumber(2), p_.GetLastDice()).equals(new TBL_Coord(10, 10)) ||
                      ComputeDestination(p_.GetPieceByNumber(3), p_.GetLastDice()).equals(new TBL_Coord(9, 9)) ||
```

```
                    ComputeDestination(p_.GetPieceByNumber(4), p_.GetLastDice()).equals(new TBL_Coord(8, 8)))
                        return true;
            default:  return false;
        }
}

// détermine le nombre de pièce dans FINISH
private int CountFINISH (TBL_Player p_) {
    int count = 0;
    for(int i = 0 ; i < p_.GetFSpot().length ; i++) {
        if(p_.GetFSpot()[i] == 1) count++;
    }
    return count;
}

// retourne la valeur du roulement de dé (avec affichage)
private int Roll (int turn) {
    switch(turn) {
        case 1:   System.out.print("\n\t\t\033[1;37m Appuyer sur une \033[1;36mENTRÉE\033[0m pour \
                  033[1;31mrouler le dé ...\033[0m");
                  break;
        case 2:   System.out.print("\n\t\t\033[1;37m Appuyer sur une \033[1;36mENTRÉE\033[0m pour \
                  033[1;32mrouler le dé ...\033[0m");
                  break;
        case 3:   System.out.print("\n\t\t\033[1;37m Appuyer sur une \033[1;36mENTRÉE\033[0m pour \
                  033[1;33mrouler le dé ...\033[0m");
                  break;
        case 4:   System.out.print("\n\t\t\033[1;37m Appuyer sur une \033[1;36mENTRÉE\033[0m pour \
                  033[1;34mrouler le dé ...\033[0m");
                  break;
        default:  System.out.println("Erreur dans TBL_Game.Roll");
    }
    scan.nextLine();
    int n = b.RollNGetDiceResult();
    GetPlayer().SetLastDice(n);
    System.out.print("\t\t\t\033[1;37m Vous avez eu un \033[1;91m" + GetPlayer().GetLastDice() + " \033[1;37!!!\
    033[0m");
    scan.nextLine();
    b.UpdateDice(n);
    return n;
}
```

```
// valide positions XX (tour double)
private boolean IsXX (TBL_Coord coord) {
	if(coord.equals(new TBL_Coord(0, 7)) || coord.equals(new TBL_Coord(7, 0)) ||
	coord.equals(new TBL_Coord(14, 7)) || coord.equals(new TBL_Coord(7, 14))) return true;
	else return false;
}

// fin de tour + update pour tour suivant
private void EndTurn () {
	if(turn == 1) turn = 2;
	else if(turn == 2) turn = 3;
	else if(turn == 3) turn = 4;
	else if(turn == 4) turn = 1;
}

// GAME OVER
private void GameOver (int color_) {
	System.out.println("GAME OVER");
	UpdatePodium(color_);
	switch(color_) {
		case 1:	p[0].SetIsActive(false);
			break;
		case 2:	p[1].SetIsActive(false);
			break;
		case 3:	p[2].SetIsActive(false);
			break;
		case 4:	p[3].SetIsActive(false);
			break;
		default:	System.exit(0);
	}
}

// update le podium
private void UpdatePodium (int color_) {
	for(int i = 0 ; i < 4 ; i++) {
		if(podium[i] == 0) {
			podium[i] = color_;
			break;
		}
	}
}
```

```
    // retourne le compte de joueurs qui ont terminé
    private int GetPodiumCount () {
        int n = 0;
        for(int i = 0 ; i < 4 ; i++) {
            if(podium[i] != 0) n++;
        }
        return n;
    }

    // affiche le podium après la partie
    private void DisplayPodium () {
        System.out.println("\n\t\t\t\033[1;32m Palmarès des \033[1;5;31mTROUBLEMAKERS\033[25;0m");
        System.out.println("\t\t\t\t\033[1;96m 1. \033[0m" + setup.ColorIntToString(podium[0]));
        System.out.println("\t\t\t\t\033[1;96m 2. \033[0m" + setup.ColorIntToString(podium[1]));
        System.out.println("\t\t\t\t\033[1;96m 3. \033[0m" + setup.ColorIntToString(podium[2]));
        System.out.println("\t\t\t\t\033[1;96m 4. \033[0m" + setup.ColorIntToString(GetLastPodium()));
        System.out.println();
    }

    // calcule couleur en 4e position
    private int GetLastPodium () {
        int n = 0;
        for(int i = 0 ; i < 4 ; i++) {
            n += podium[i];
        }
        if(n == 6) return 4;
        else if(n == 7) return 3;
        else if(n == 8) return 2;
        else if(n == 9) return 1;
        return 0;
    }

    // efface l'affichage de la console
    public static void Clear () {
        try {
            if(System.getProperty("os.name").contains("Windows"))
                new ProcessBuilder("cmd", "/c", "cls").inheritIO().start().waitFor();
            else System.out.print("\033\143");
        }
        catch (IOException | InterruptedException ex) {System.out.println("Clear : " + ex.getMessage());}
    }
}
```

“TBL_Player.java”

```
package game;

import board.TBL_Coord;
import pieces.TBL_Piece;

public class TBL_Player {
	private int[] homeSpot, finishSpot;
	private int color, lastDice;
	private TBL_Piece o1, o2, o3, o4;
	private boolean isActive;
	private boolean isHuman;

	public TBL_Player () {
		Init(0);
	}

	public TBL_Player (int color_) {
		Init(color_);
	}

	private void Init (int color_) {
		isActive = false;
		isHuman = false;
		lastDice = 0;
		// initialise homeSpot à 1
		homeSpot = new int[4];
		for(int i = 0 ; i < 4 ; i++) homeSpot[i] = 1;
		// initialise finishSpot à 0
		finishSpot = new int[4];
		for(int i = 0 ; i < 4 ; i++) finishSpot[i] = 0;

		color = color_;
		switch(color) {
			case 1:	o1 = new TBL_Piece(new TBL_Coord(5, 12), color);
				o2 = new TBL_Piece(new TBL_Coord(6, 12), color);
				o3 = new TBL_Piece(new TBL_Coord(7, 12), color);
				o4 = new TBL_Piece(new TBL_Coord(8, 12), color);
				break;
			case 2:	o1 = new TBL_Piece(new TBL_Coord(2, 6), color);
				o2 = new TBL_Piece(new TBL_Coord(2, 7), color);
				o3 = new TBL_Piece(new TBL_Coord(2, 8), color);
				o4 = new TBL_Piece(new TBL_Coord(2, 9), color);
				break;
			case 3:	o1 = new TBL_Piece(new TBL_Coord(8, 2), color);
				o2 = new TBL_Piece(new TBL_Coord(7, 2), color);
				o3 = new TBL_Piece(new TBL_Coord(6, 2), color);
```

```
                o4 = new TBL_Piece(new TBL_Coord(5, 2), color);
                break;
        case 4:  o1 = new TBL_Piece(new TBL_Coord(12, 9), color);
                o2 = new TBL_Piece(new TBL_Coord(12, 8), color);
                o3 = new TBL_Piece(new TBL_Coord(12, 7), color);
                o4 = new TBL_Piece(new TBL_Coord(12, 6), color);
                break;
        default: System.out.println("Problème dans TBL_Player.Init()");
    }
}

// retourne la pièce selon coordonnées
public TBL_Piece GetPieceByCoord (TBL_Coord coord) {
    if(o1.GetPos().equals(coord)) return o1;
    else if(o2.GetPos().equals(coord)) return o2;
    else if(o3.GetPos().equals(coord)) return o3;
    else if(o4.GetPos().equals(coord)) return o4;
    else return new TBL_Piece(new TBL_Coord(-1, -1), 0); // couleur 0 ET coord(-1,-1) si erreur
}

// retourne la pièce selon son numéro
public TBL_Piece GetPieceByNumber (int n) {
    if(n == 1) return o1;
    else if(n == 2) return o2;
    else if(n == 3) return o3;
    else if(n == 4) return o4;
    else return new TBL_Piece(new TBL_Coord(-1, -1), 0);
}

public void RetrieveFromHOME (int index) {homeSpot[index] = 0;}
public void PlaceIntoHOME (int index) {homeSpot[index] = 1;}

public void PlaceIntoFINISH (int index) {finishSpot[index] = 1;}

public boolean GetIsActive () {return isActive;}
public void SetIsActive (boolean isActive_) {isActive = isActive_;}

public boolean GetIsHuman () {return isHuman;}
public void SetIsHuman (boolean isHuman_) {isHuman = isHuman_;}

public int GetColor () {return color;}
public void SetColor (int color_) { color = color_;}

public int GetLastDice () {return lastDice;}
public void SetLastDice (int n) {lastDice = n;}

public int[] GetFSpot () {return finishSpot;}
```

```
    public void SetFSpot (int value, int index) {finishSpot[index] = value;}
    public int[] GetHSpot () {return homeSpot;}
    public void SetHSpot (int value, int index) {homeSpot[index] = value;}

    // valide si des pièces RUNNING présentes pour joueur
    public boolean HasRUNNING (TBL_Player p_) {
        if(p_.o1.GetStatus() == 1) return true;
        else if(p_.o2.GetStatus() == 1) return true;
        else if(p_.o3.GetStatus() == 1) return true;
        else if(p_.o4.GetStatus() == 1) return true;
        else return false;
    }
}
```

“TBL_Setup.java”

```
package game;

import board.*;
import java.util.Scanner;

public class TBL_Setup {
	private TBL_Board b;
	private TBL_Player[] p;
	private Scanner scan;
	private int first;

	public TBL_Setup () {
		Init();
		Init2();
		Menu1();
	}

	private void Init () {
		scan = new Scanner(System.in);
		TBL_Game.Clear();
		Splash();
		scan.nextLine();
		first = 0;
		b = new TBL_Board();
		p = new TBL_Player[4];
		for(int i = 0; i < 4 ; i++) p[i] = new TBL_Player((i+1));
	}

	// update board avec les positions des pièces de tous les joueurs
	private void Init2 () {
		for(int i = 0 ; i < 4 ; i++) {
			for(int j = 1 ; j <=4 ; j++) {
				b.UpdateSituation(p[i].GetPieceByNumber(j).GetPos(), (i+1), 0);
			}
		}
	}

	// menu de sélection de joueurs
	private void Menu1 () {
		String choice = "";

		System.out.println("\033[1;37m** Sélectionner une couleur et/ou adversaires **\033[0m\n");
		// rouge
		do {
			System.out.print("\t\033[1;37mJoueur \033[1;31mROUGE\033[0m \033[35m(H[umain] / A[uto]) \033[1;37m: \
			033[1;36m");
```

```
        choice = scan.nextLine().toLowerCase();
    }
    while(!ValidateMenu1Choice(choice));
    if(choice.equals("h")) {
        p[0].SetIsHuman(true);
    }
    p[0].SetIsActive(true);
    p[0].SetColor(1);
    System.out.println("\t\033[1;37mDé roulé : \033[5;1;31m" + RollForTurn(p[0]) + "\033[25;0m");

    // vert
    choice = "";
    do {
        System.out.print("\t\033[1;37mJoueur \033[1;32mVERT\033[0m \033[35m(H[umain] / A[uto]) \033[1;37m: \
        033[1;36m");
        choice = scan.nextLine().toLowerCase();
    }
    while(!ValidateMenu1Choice(choice));
    if(choice.equals("h")) {
        p[1].SetIsHuman(true);
    }
    p[1].SetIsActive(true);
    p[1].SetColor(2);
    System.out.println("\t\033[1;37mDé roulé : \033[5;1;31m" + RollForTurn(p[1]) + "\033[25;0m");

    // jaune
    choice = "";
    do {
        System.out.print("\t\033[1;37mJoueur \033[1;33mJAUNE\033[0m \033[35m(H[umain] / A[uto]) \033[1;37m: \
        033[1;36m");
        choice = scan.nextLine().toLowerCase();
    }
    while(!ValidateMenu1Choice(choice));
    if(choice.equals("h")) {
        p[2].SetIsHuman(true);
    }
    p[2].SetIsActive(true);
    p[2].SetColor(3);
    System.out.println("\t\033[1;37mDé roulé : \033[5;1;31m" + RollForTurn(p[2]) + "\033[25;0m");

    // bleu
    choice = "";
    do {
        System.out.print("\t\033[1;37mJoueur \033[1;34mBLEU\033[0m \033[35m(H[umain] / A[uto]) \033[1;37m: \
        033[1;36m");
        choice = scan.nextLine().toLowerCase();
    }
```

```
        while(!ValidateMenu1Choice(choice));
        if(choice.equals("h")) {
            p[3].SetIsHuman(true);
        }
        p[3].SetIsActive(true);
        p[3].SetColor(4);
        System.out.println("\t\033[1;37mDé roulé : \033[5;1;31m" + RollForTurn(p[3]) + "\033[25;0m");

        System.out.print("\t\t\033[1;37mLe joueur " + ColorIntToString(WhoPlaysFirst(p)) + " \033[1;37mdébute ! ...\
        033[0m");
        scan.nextLine();
    }

    // retourne le nom de couleur (coloré) selon int
    public String ColorIntToString (int n) {
        String s = "";
        if(n == 1) {
            s = "\033[5;1;31mROUGE\033[25;1;0m";
            return s;
        }
        else if(n == 2) {
            s = "\033[5;1;32mVERT\033[25;0m";
            return s;
        }
        else if(n == 3) {
            s = "\033[5;1;33mJAUNE\033[25;0m";
            return s;
        }
        else if(n == 4) {
            s = "\033[5;1;34mBLEU\033[25;0m";
            return s;
        }
        else {
            s = "TROUBLE !!!";
            return s;
        }
    }

    // roule le dé
    private int RollForTurn (TBL_Player p_) {
        p_.SetLastDice(b.RollNGetDiceResult());
        return p_.GetLastDice();
    }
```

```
	// détermine quel joueur commence une partie
	private int WhoPlaysFirst (TBL_Player[] p) {
		int max1 = 0;
		int max2 = 0;
		if(p[0].GetLastDice() >= p[1].GetLastDice()) max1 = 1;
		else max1 = 2;
		if(p[2].GetLastDice() >= p[3].GetLastDice()) max2 = 1;
		else max2 = 2;
		if(max1 == 1 && max2 == 1) {
			if(p[0].GetLastDice() >= p[2].GetLastDice()) return first = 1;
			else return first = 3;
		}
		else if(max1 == 1 && max2 == 2) {
			if(p[0].GetLastDice() >= p[3].GetLastDice()) return first = 1;
			else return first = 4;
		}
		else if(max1 == 2 && max2 == 1) {
			if(p[1].GetLastDice() >= p[2].GetLastDice()) return first = 2;
			else return first = 3;
		}
		else if(max1 == 2 && max2 == 2) {
			if(p[1].GetLastDice() >= p[3].GetLastDice()) return first = 2;
			else return first = 4;
		}
		else return first = 0;
	}

	// retourne le numéro du premier joueur
	public int GetFirstToPlay () {return first;}

	// valide le choix humain vs auto
	private boolean ValidateMenu1Choice (String s) {
		if(s.length() > 1) return false;
		if(s.equals("h") || s.equals("a")) return true;
		else return false;
	}

	// retourne board
	public TBL_Board GetBoard () {return b;}

	// retourne les 4 joueurs
	public TBL_Player[] GetPlayers () {return p;}

	// splash
	public void Splash () {
	System.out.println("\033[1;31m################## \033[1;33m ######### \033[1;32m ##### \
033[1;34m#### #### \033[1;31m############ \033[1;32m############\033[0m");
```

```
        System.out.println("\033[1;31m################ \033[1;33m ############ \033[1;32m ####
#### \033[1;34m#### #### \033[1;31m############## \033[1;32m############\033[0m");
        System.out.println("\033[1;31m #### \033[1;33m##### ## \033[1;32m#### #### \033[1;34m####
#### \033[1;31m#### ### \033[1;32m### \033[0m");
        System.out.println("\033[1;31m #### \033[1;33m#### ### \033[1;32m### ### \033[1;34m#### #### \
033[1;31m#### ### \033[1;32m### \033[0m");
        System.out.println("\033[1;31m #### \033[1;33m#### ######## \033[1;32m### ### \033[1;34m####
#### \033[1;31m############ \033[1;32m########## \033[0m");
        System.out.println("\033[1;31m #### \033[1;33m#### #### \033[1;32m### ### \033[1;34m#### ####
\033[1;31m############## \033[1;32m### \033[0m");
        System.out.println("\033[1;31m #### \033[1;33m#### ### \033[1;32m#### ### \033[1;34m ####
#### \033[1;31m#### ### \033[1;32m### \033[0m");
        System.out.println("\033[1;31m #### \033[1;33m#### ### \033[1;32m #### #### \033[1;34m
########## \033[1;31m#### #### \033[1;32m############\033[0m");
        System.out.println("\033[1;31m #### \033[1;33m#### ### \033[1;32m ##### \033[1;34m ####### \
033[1;31m############## \033[1;32m############\033[0m");
        System.out.println("\033[5;101m \033[25;0m");
        System.out.println("\033[1;32m #### \033[1;34m#### ### \033[1;31m ##### \033[1;33m ####### \
033[1;34m############## \033[1;33m############\033[0m");
        System.out.println("\033[1;32m #### \033[1;34m#### ### \033[1;31m #### #### \033[1;33m
########## \033[1;34m#### #### \033[1;33m############\033[0m");
        System.out.println("\033[1;32m #### \033[1;34m#### ### \033[1;31m#### ### \033[1;33m ####
#### \033[1;34m#### ### \033[1;33m### \033[0m");
        System.out.println("\033[1;32m #### \033[1;34m#### #### \033[1;31m### ### \033[1;33m#### ####
\033[1;34m############## \033[1;33m### \033[0m");
        System.out.println("\033[1;32m #### \033[1;34m#### ######## \033[1;31m### ### \033[1;33m####
#### \033[1;34m############ \033[1;33m########## \033[0m");
        System.out.println("\033[1;32m #### \033[1;34m#### ### \033[1;31m### ### \033[1;33m#### #### \
033[1;34m#### ### \033[1;33m### \033[0m");
        System.out.println("\033[1;32m #### \033[1;34m#### ## \033[1;31m#### #### \033[1;33m#### ####
\033[1;34m#### ### \033[1;33m### \033[0m");
        System.out.println("\033[1;32m################ \033[1;34m############ \033[1;31m ####
#### \033[1;33m#### #### \033[1;34m############ \033[1;33m############\033[0m");
        System.out.println("\033[1;32m################ \033[1;34m######### \033[1;31m ##### \
033[1;33m#### #### \033[1;34m############ \033[1;33m############\033[0m");

    }
}
```

“Trouble.java”

```
import game.TBL_Game;

public class Trouble {
	public static void main (String[] args) {
		TBL_Game game = new TBL_Game();
		while(true) game.PlayTurn();
	}
}
```