

# Rapport des tests JUnit

Les méthodes testées dans notre projet sont les méthodes `deplacerRobot` et `extraire` de la classe `Monde`.

Nous avons choisi d'utiliser une couverture de conditions pour tester nos méthodes, ayant beaucoup de critères à vérifier dans nos méthodes, cette couverture nous paraissait la plus adaptée pour tester correctement notre code.

## Méthode extraire de la classe Monde

code :

```

268 @ public void extraire(Robot robot, Mine mine){ 1 usage  ± Remy
269     boolean MineraiSuffisant = mine.getNbMinerais() > robot.getCapaciteExtraction();
270     boolean StockageSuffisant = robot.getCapaciteStockage() > (robot.getNbMineraisExtraits()+robot.getCapaciteExtraction());
271     int minerai;
272
273     if (MineraiSuffisant && StockageSuffisant) {
274         minerai = robot.getCapaciteExtraction();
275     }
276     else if (MineraiSuffisant && !StockageSuffisant) {
277         minerai = robot.getCapaciteStockage()-robot.getNbMineraisExtraits();
278     }
279     else if (!MineraiSuffisant && !StockageSuffisant){
280         minerai = Math.min(robot.getCapaciteStockage() - robot.getNbMineraisExtraits(), mine.getNbMinerais());
281     }
282     else {
283         minerai = mine.getNbMinerais();
284     }
285
286     robot.setnbMineraisExtraits(robot.getNbMineraisExtraits() + minerai);
287     mine.setNbMinerais(mine.getNbMinerais() - minerai);
288 }

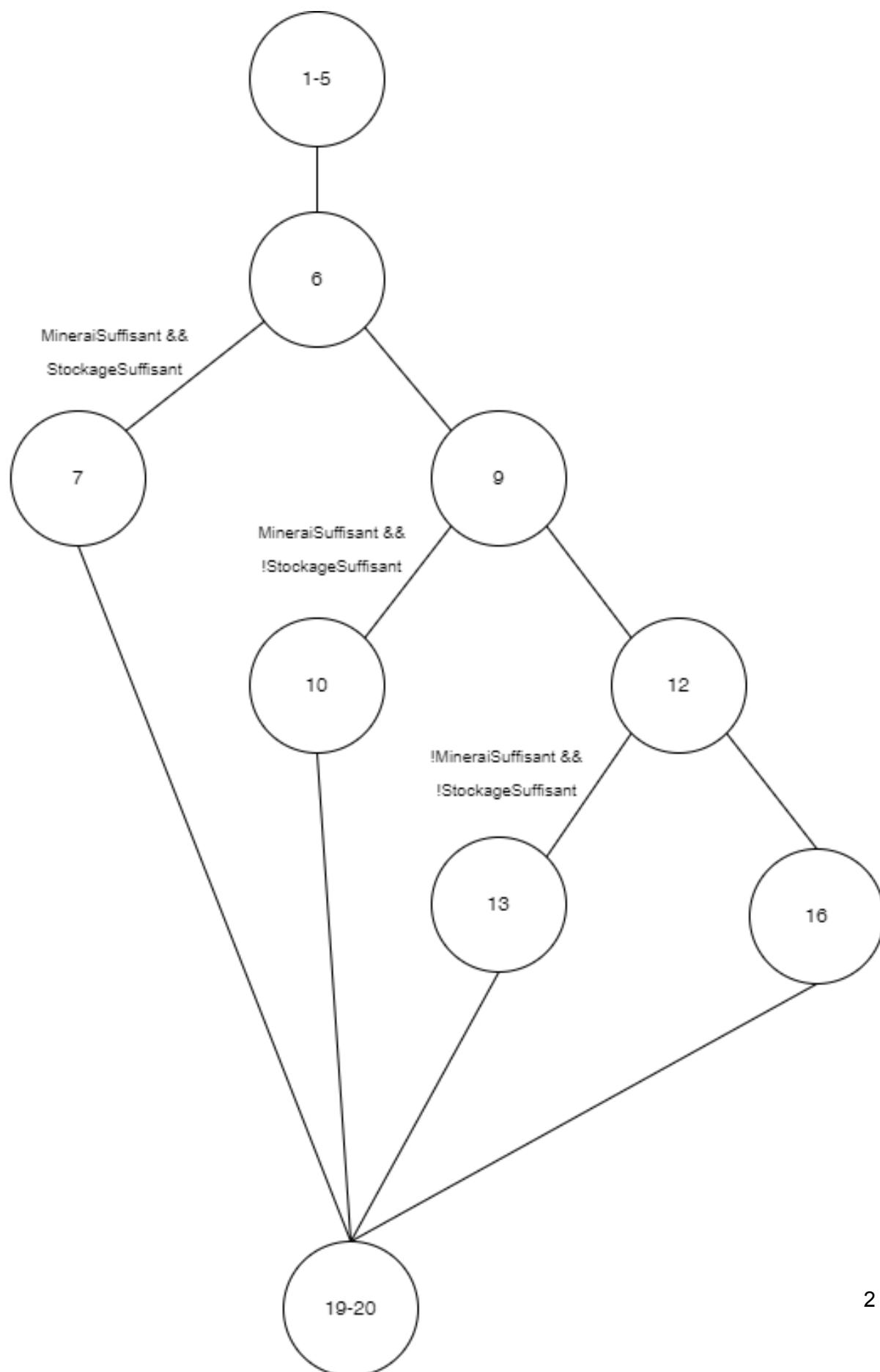
```

correspondance des lignes :

1- 5	public void extraire(Robot robot, Mine mine){ boolean MineraiSuffisant = mine.getNbMinerais()>robot.getCapaciteExtraction(); boolean StockageSuffisant = robot.getCapaciteStockage() > (robot.getNbMineraisExtraits()+robot.getCapaciteExtraction()); int minerai;
6	if (MineraiSuffisant && StockageSuffisant) {
7	minerai = robot.getCapaciteExtraction();
9	else if (MineraiSuffisant && !StockageSuffisant) {
10	minerai = robot.getCapaciteStockage()-robot.getNbMineraisExtraits();
12	else if (!MineraiSuffisant && !StockageSuffisant){
13	minerai = Math.min(robot.getCapaciteStockage() - robot.getNbMineraisExtraits(), mine.getNbMinerais());
16	minerai = mine.getNbMinerais();
19- 20	robot.setnbMineraisExtraits(robot.getNbMineraisExtraits() + minerai);

```
mine.setNbMinerais(mine.getNbMinerais() - minerais);
```

graphe de flot de contrôle :



### Tableau des cas de tests :

Il y a donc 4 tests à effectuer selon le graphe de flot de contrôle (nombre de régions du graphe + 1 = 4)

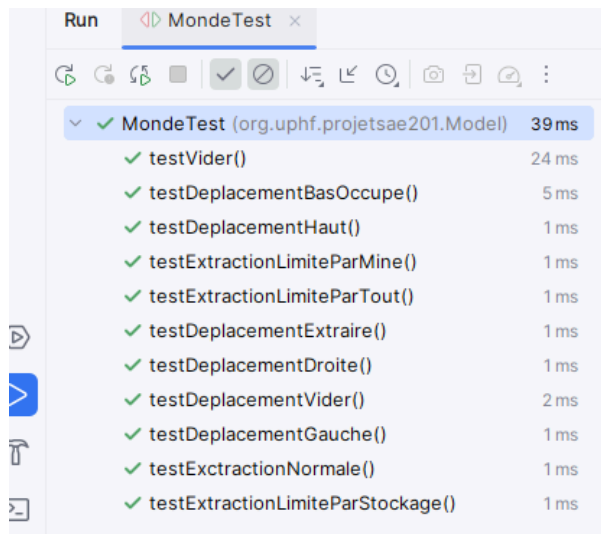
couverture de condition:

MineraiSuffisant (dans la mine)	StockageSuffisant (besace du robot)	cas de tests
V	V	minerais dans mine=50 capacité stockage robot=15 capacité extraction robot=5 nombre de minerai dans sac du robot=0
V	F	minerais dans mine=50 capacité stockage robot=15 capacité extraction robot=3 nombre de minerai dans sac du robot=13
F	V	minerais dans mine=2 capacité stockage robot=15 capacité extraction robot=3 nombre de minerai dans sac du robot=0
F	F	minerais dans mine=2 capacité stockage robot=15 capacité extraction robot=3 nombre de minerai dans sac du robot=14

## Code des tests :

```
56  /*TEST METHODE extraire*/
57  @Test  ± Margot
58  public void testExtractionNormale() {
59      /*Ici, le robot a assez de stockage disponible par rapport a sa capacité d'extraction et la mine a assez de minerai par rappo
60      robot.setCapaciteStockage(15);
61      robot.setCapaciteExtraction(5);
62      robot.setnbMineraisExtraits(0);
63      mine.setNbMinerais(50);
64
65      monde.extraire(robot, mine);
66      Assertions.assertEquals( expected: 5, robot.getNbMineraisExtraits(), message: "Le robot devrait avoir extrait 5 minerais");
67      Assertions.assertEquals( expected: 45, mine.getNbMinerais(), message: "Il devrait rester 45 minerais dans la mine");
68  }
69
70  @Test  ± Margot
71  public void testExtractionLimiteParStockage() {
72      /*Ici, le robot n'a pas assez de stockage par rapport a sa capacité d'extraction et la mine a assez de minerai par rappor
73      robot.setCapaciteStockage(15);
74      robot.setCapaciteExtraction(3);
75      robot.setnbMineraisExtraits(13);
76      mine.setNbMinerais(50);
77
78      monde.extraire(robot, mine);
79      Assertions.assertEquals( expected: 15, robot.getNbMineraisExtraits(), message: "Le robot devrait avoir extrait 5 minerais");
80      Assertions.assertEquals( expected: 48, mine.getNbMinerais(), message: "Il devrait rester 95 minerais dans la mine");
81  }
82
83  public void testExtractionLimiteParMine() {
84      /*Ici, le robot a assez de stockage par rapport a sa capacité d'extraction mais la mine n'a pas assez de minerai par rappo
85
86
87      robot.setCapaciteStockage(15);
88      robot.setCapaciteExtraction(3);
89      robot.setnbMineraisExtraits(0);
90      mine.setNbMinerais(2);
91
92      monde.extraire(robot, mine);
93      Assertions.assertEquals( expected: 2, robot.getNbMineraisExtraits(), message: "Le robot devrait avoir extrait 2 minerais");
94      Assertions.assertEquals( expected: 0, mine.getNbMinerais(), message: "La mine devrait être vide");
95  }
96
97  @Test  ± Margot
98  public void testExtractionLimiteParTout() {
99      /*Ici, le robot n'a pas assez de stockage par rapport a sa capacité d'extraction et la mine n'a assez de minerai par rappo
100
101
102      robot.setCapaciteStockage(15);
103      robot.setCapaciteExtraction(3);
104      robot.setnbMineraisExtraits(14);
105      mine.setNbMinerais(2);
106
107      monde.extraire(robot, mine);
108      Assertions.assertEquals( expected: 15, robot.getNbMineraisExtraits(), message: "Le robot devrait avoir extrait 1 minerais");
109      Assertions.assertEquals( expected: 1, mine.getNbMinerais(), message: "Il devrait rester 1 minerais dans la mine");
110  }
```

## Résultats des tests :



The screenshot shows the 'Run' window of an IDE. At the top, there's a tab labeled 'MondeTest' with a close button. Below the tab is a toolbar with various icons for running and debugging. The main area displays a list of test results. The first item is 'MondeTest (org.uphf.projetsae201.Model)' with a green checkmark and a duration of '39 ms'. Below it, there are ten individual test methods, each with a green checkmark and a duration. The tests are: testVider() (24 ms), testDeplacementBasOccupe() (5 ms), testDeplacementHaut() (1 ms), testExtractionLimiteParMine() (1 ms), testExtractionLimiteParTout() (1 ms), testDeplacementExtraire() (1 ms), testDeplacementDroite() (1 ms), testDeplacementVider() (2 ms), testDeplacementGauche() (1 ms), testExtractionNormale() (1 ms), and testExtractionLimiteParStockage() (1 ms). On the left side of the window, there are several icons: a play button, a blue arrow, a wrench, and a terminal icon.

Test Method	Duration
✓ MondeTest (org.uphf.projetsae201.Model)	39 ms
✓ testVider()	24 ms
✓ testDeplacementBasOccupe()	5 ms
✓ testDeplacementHaut()	1 ms
✓ testExtractionLimiteParMine()	1 ms
✓ testExtractionLimiteParTout()	1 ms
✓ testDeplacementExtraire()	1 ms
✓ testDeplacementDroite()	1 ms
✓ testDeplacementVider()	2 ms
✓ testDeplacementGauche()	1 ms
✓ testExtractionNormale()	1 ms
✓ testExtractionLimiteParStockage()	1 ms

## méthode déplacerRobot de la classe Monde

code :

```
241 @ public void déplacerRobot(String direction, Terrain T) { 15 usages  ± Remy +1 *
242     Robot r = T.getRobot();
243
244     if (r.verifDeplacement( m: this, direction)) {
245         int tmpY = r.getCoordonneesY();
246         int tmpX = r.getCoordonneesX();
247
248         switch (direction) {
249             case "Haut" -> tmpX -= 1;
250             case "Bas" -> tmpX += 1;
251             case "Gauche" -> tmpY -= 1;
252             case "Droit" -> tmpY += 1;
253             case "Extraire" -> {
254                 if (((Terrain) this.lstSecteur[tmpX][tmpY]).getDistrict() instanceof Mine) {
255                     this.extraire(r, (Mine) ((Terrain) this.lstSecteur[tmpX][tmpY]).getDistrict());
256                     T.setRobot(r);
257                 }
258                 return;
259             }
260             case "Vider" -> {
261                 if (((Terrain) this.lstSecteur[tmpX][tmpY]).getDistrict() instanceof Entrepot) {
262                     this.vider(r, (Entrepot) ((Terrain) this.lstSecteur[tmpX][tmpY]).getDistrict());
263                 }
264                 return;
265             }
266         }
267
268         if (((Terrain) this.lstSecteur[tmpX][tmpY]).getRobot() != null) {
269             Robot r2 = ((Terrain) this.lstSecteur[tmpX][tmpY]).getRobot();
270             ((Terrain) this.lstSecteur[r.getCoordonneesX()][r.getCoordonneesY()]).setRobot(r2);
271             ((Terrain) this.lstSecteur[tmpX][tmpY]).setRobot(r);
272             int originalX = r.getCoordonneesX();
273             int originalY = r.getCoordonneesY();
274             r.setCoordonneesX(tmpX);
275             r.setCoordonneesY(tmpY);
276             r2.setCoordonneesX(originalX);
277             r2.setCoordonneesY(originalY);
278         }
279         else {
280             ((Terrain) this.lstSecteur[tmpX][tmpY]).setRobot(r);
281             ((Terrain) this.lstSecteur[r.getCoordonneesX()][r.getCoordonneesY()]).setRobot(null);
282             r.setCoordonneesX(tmpX);
283             r.setCoordonneesY(tmpY);
284         }
285     }
286 }
```

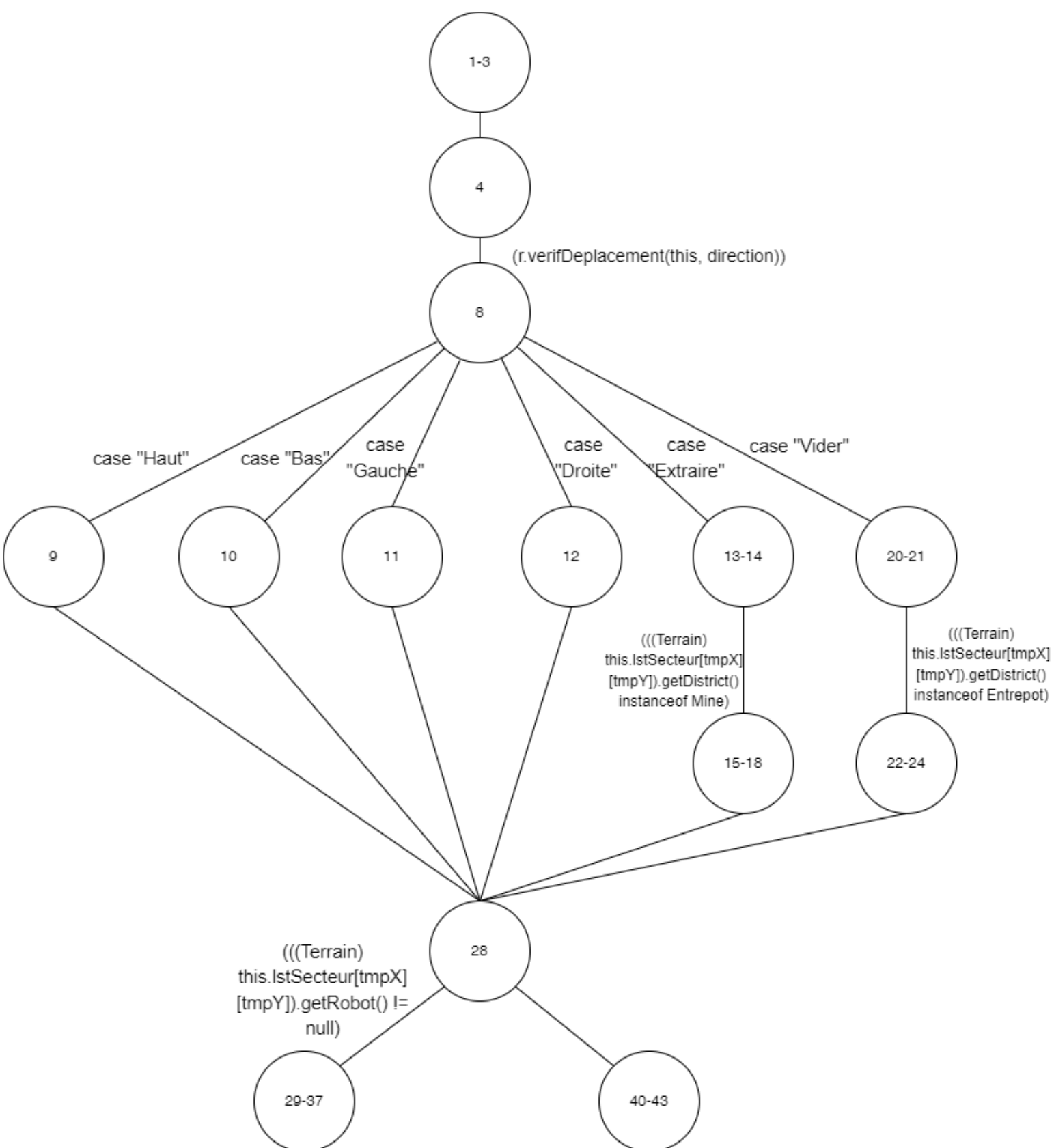
## correspondance des lignes :

1-3	<code>public void deplacerRobot(String direction, Terrain T) {     Robot r = T.getRobot();</code>
4	<code>if (r.verifDeplacement(this, direction)) {</code>
8	<code>switch (direction) {</code>
9	<code>case "Haut" -&gt; tmpX -= 1;</code>
10	<code>case "Bas" -&gt; tmpX += 1;</code>
11	<code>case "Gauche" -&gt; tmpY -= 1;</code>
12	<code>case "Droit" -&gt; tmpY += 1;</code>
13-14	<code>case "Extraire" -&gt; { if (((Terrain) this.lstSecteur[tmpX][tmpY]).getDistrict() instanceof Mine) {</code>
15-18	<code>this.extraire(r, (Mine) ((Terrain) this.lstSecteur[tmpX][tmpY]).getDistrict());     T.setRobot(r); } return;</code>
20-21	<code>case "Vider" -&gt; {     if (((Terrain) this.lstSecteur[tmpX][tmpY]).getDistrict() instanceof Entrepot) {</code>
22-24	<code>this.vider(r, (Entrepot) ((Terrain) this.lstSecteur[tmpX][tmpY]).getDistrict()); } return;</code>
28	<code>if (((Terrain) this.lstSecteur[tmpX][tmpY]).getRobot() != null) {</code>
29-37	<code>Robot r2 = ((Terrain) this.lstSecteur[tmpX][tmpY]).getRobot(); Robot temp = new Robot(r);  ((Terrain) this.lstSecteur[r.getCoordonneesX()][r.getCoordonneesY()]).setRobot(r2); ((Terrain) this.lstSecteur[tmpX][tmpY]).setRobot(temp);  r2.setCoordonneesX(temp.getCoordonneesX()); r2.setCoordonneesY(temp.getCoordonneesY());</code>
40-43	<code>((Terrain) this.lstSecteur[tmpX][tmpY]).setRobot(r); ((Terrain) this.lstSecteur[r.getCoordonneesX()][r.getCoordonneesY()]).setRobot(null); r.setCoordonneesX(tmpX);</code>



```
r.setCoordonneesY(tmpY);
```

graphe de flot de contrôle :



### Tableau des cas de tests :

Il y a donc 6 tests à effectuer selon le graphe de flot de contrôle (nombre de régions du graphe + 1 = 6)

couverture de conditions :

"Haut"	"Bas"	"Gauche"	"Droit"	"Extraire"	"Vider"	Si le secteur est une mine	Si le secteur est un entrepôt	Si le secteur est occupé	cas de test
V	F	F	F	F	F	F	F	F	direction = "Haut"
F	V	F	F	F	F	F	F	V	direction="Bas" secteur du bas occupé (!null)
F	F	V	F	F	F	F	F	F	direction="Gauche"
F	F	F	V	F	F	F	F	F	direction="Droit"
F	F	F	F	F	V	F	V	F	direction="Vider" le secteur est un entrepôt (true)
F	F	F	F	V	F	V	F	F	direction="Extraire" le secteur est une mine (true)

## code des tests :

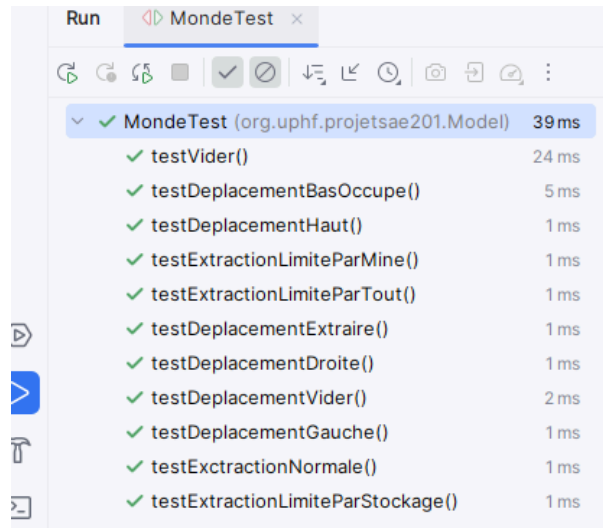
```
114  /*TEST METHODE deplacerRobot*/
115  @Test  ± Margot *
116  void testDeplacementHaut() {
117      monde.deplacerRobot( direction: "Haut", ((Terrain) monde.getLstSecteur()[2][2]));
118      assertNull(((Terrain) monde.getLstSecteur()[2][2]).getRobot()); //vérifie que le terrain initial ne cont
119      assertNotNull(((Terrain) monde.getLstSecteur()[1][2]).getRobot()); //vérifie que le terrain du haut cont
120      Assertions.assertEquals( expected: 1, ((Terrain) monde.getLstSecteur()[1][2]).getRobot().getCoordonneesX());
121  }
122
123  @Test  ± Margot *
124  void testDeplacementBasOccupe() {
125      monde.deplacerRobot( direction: "Bas", ((Terrain) monde.getLstSecteur()[2][2]));
126      assertNotNull(((Terrain) monde.getLstSecteur()[2][2]).getRobot()); //vérifie que le terrain initial cont
127      assertNotNull(((Terrain) monde.getLstSecteur()[3][2]).getRobot()); //vérifie que le terrain du bas conti
128
129      Assertions.assertEquals( expected: 3, robot.getCoordonneesX());
130      Assertions.assertEquals( expected: 2, robot.getCoordonneesY());
131      Assertions.assertEquals( expected: 2, autreRobot.getCoordonneesX());
132      Assertions.assertEquals( expected: 2, autreRobot.getCoordonneesY());
133  }
134
135  @Test  ± Margot *
136  void testDeplacementGauche() {
137      monde.deplacerRobot( direction: "Gauche", ((Terrain) monde.getLstSecteur()[2][2]) );
138      assertNull(((Terrain) monde.getLstSecteur()[2][2]).getRobot()); //vérifie que le terrain :
139      assertNotNull(((Terrain) monde.getLstSecteur()[2][1]).getRobot()); //vérifie que le terra:
140      Assertions.assertEquals( expected: 1, robot.getCoordonneesY()); //on vérifie les coordonnées
141  }
142
143  @Test  ± Margot *
144  void testDeplacementDroite() {
145      monde.deplacerRobot( direction: "Droit", ((Terrain) monde.getLstSecteur()[2][2]) );
146      assertNull(((Terrain) monde.getLstSecteur()[2][2]).getRobot()); //vérifie que le terrain :
147      assertNotNull(((Terrain) monde.getLstSecteur()[2][3]).getRobot()); //vérifie que le terra:
148      Assertions.assertEquals( expected: 3, robot.getCoordonneesY()); //on vérifie les coordonnées
149  }
150
151  @Test  ± Margot
152  void testDeplacementVider() {
153      robot.setCapaciteStockage(15); //on set les paramètres nécessaires au test
154      robot.setCapaciteExtraction(3);
155      robot.setnbMineraisExtraits(15);
156      entrepot.setnbMineraisStockes(0);
157
158
159      monde.deplacerRobot( direction: "Droit", ((Terrain) monde.getLstSecteur()[2][2]));
160      monde.deplacerRobot( direction: "Vider", ((Terrain) monde.getLstSecteur()[2][3]));
161
162      Assertions.assertEquals( expected: 15, entrepot.getnbMineraisStockes(), message: "L'entrepôt devrait maintenant
163      Assertions.assertEquals( expected: 0, robot.getNbMineraisExtraits(), message: "Le robot devrait avoir 0 minera
164  }
```

```

103
166 @Test  ± Margot
167 void testDeplacementExtraire() {
168     robotExtraire.setCapaciteStockage(15); //on set les paramètres nécessaires au test
169     robotExtraire.setCapaciteExtraction(3);
170     robotExtraire.setNbMineraisExtraits(0);
171     mine.setNbMinerais(20);
172
173     monde.deplacerRobot( direction: "Extraire", ((Terrain) monde.getLstSecteur()[1][3]));
174
175     Assertions.assertEquals( expected: 3, robotExtraire.getNbMineraisExtraits(), message: "Le robot devrait
176     Assertions.assertEquals( expected: 17, mine.getNbMinerais(), message: "La mine devrait contenir 17 mine
177 }

```

## Résultats des tests :



The screenshot shows the 'Run' window of an IDE. At the top, there is a tab labeled 'MondeTest' with a red 'x' icon. Below the tab is a toolbar with various icons for running and debugging. The main area displays a list of test results for 'MondeTest (org.uphf.projetsae201.Model)'. The total execution time is 39 ms. Each test is marked with a green checkmark and its duration in milliseconds.

Test Name	Duration
✓ MondeTest (org.uphf.projetsae201.Model)	39 ms
✓ testVider()	24 ms
✓ testDeplacementBasOccupe()	5 ms
✓ testDeplacementHaut()	1 ms
✓ testExtractionLimiteParMine()	1 ms
✓ testExtractionLimiteParTout()	1 ms
✓ testDeplacementExtraire()	1 ms
✓ testDeplacementDroite()	1 ms
✓ testDeplacementVider()	2 ms
✓ testDeplacementGauche()	1 ms
✓ testExtractionNormale()	1 ms
✓ testExtractionLimiteParStockage()	1 ms

### Conclusion :

Nos méthodes fonctionnent donc bien selon les résultats de chacun de nos tests. Des tests supplémentaires ont également été effectués pour la méthode `verifDeplacement` de la classe `Robot` (pour tester correctement la méthode `deplacerRobot` de la classe `Monde`) ainsi que pour d'autres méthodes telles que la méthode `vider` de la classe `Monde` et les méthodes `EstPlanEau` et `EstPasDansLeMonde` de la classe `Robot`, afin de s'assurer du bon fonctionnement de notre code.