**III/ Conception**

**3.1. Fichiers hpp**

**3.2. Test unitaires**

1. **Classe EcoAgentID**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Classe | Méthode | Test | Donnees |  | Résultat attendu | Principe |
| EcoAgentID | operator== | egalité | EcoAgentID(identifiant=5) | EcoAgentID(identifiant=5) | TRUE | - |
|  |  | inégalité | EcoAgentID(identifiant=5) | EcoAgentID(identifiant=2) | FALSE | - |
|  | getId | Obtention de l'ID correct | EcoAgentID(identifiant=1) | - | 1 | - |
|  | generateID | Bonne incrémentation du nombre de generations | - | - | 0 puis 1 puis 2 puis 3 etc. | On affiche l'attribut statique nombre de générations avant et après création d'EcoAgentID |

1. **Classe EcoAgent**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Classe | Méthode | Test | Donnees |  | Résultat attendu |
| EcoAgent | getEtat | état déterminé | EcoAgent e ( EcoAgentID(id=13), etat = RS, positionCourante : EcoAgentID(id=3), positionFinale : EcoAgentID(id=11)) | - | Etat = RS |
|  |  | état indéterminé | EcoAgent e ( EcoAgentID(id=13), etat = NULL, positionCourante : EcoAgentID(id=3), positionFinale : EcoAgentID(id=11)) | - | NULL |
|  | getId | ID déterminé | EcoAgent e ( EcoAgentID(id=13), etat = RS, positionCourante : EcoAgentID(id=3), positionFinale : EcoAgentID(id=11)) | - | EcoAgentID(id=13) |
|  |  | ID indéterminé | EcoAgent e ( EcoAgentID(id=NULL), etat = RS, positionCourante : EcoAgentID(id=3), positionFinale : EcoAgentID(id=11)) | - | NULL |
|  | setEtat | test sur un EcoAgent quelconque | EcoAgent e ( EcoAgentID(id=13), etat = RS, positionCourante : EcoAgentID(id=3), positionFinale : EcoAgentID(id=11)) | Etat = RF | Etat prend la valeur RF |
|  | setId | test sur un EcoAgent quelconque | EcoAgent e ( EcoAgentID(id=13), etat = RS, positionCourante : EcoAgentID(id=3), positionFinale : EcoAgentID(id=11)) | EcoAgentID(id=9) | l'identifiant de l'EcoAgentID prend la valeur 9 |
|  | getPositionCourante | positionCourante déterminée | EcoAgent e ( EcoAgentID(id=13), etat = RS, positionCourante : EcoAgentID(id=3), positionFinale : EcoAgentID(id=11)) | - | EcoAgentID(id=3) |
|  |  | positionCourante indéterminée | EcoAgent e ( EcoAgentID(id=13), etat = RS, positionCourante : NULL, positionFinale : EcoAgentID(id=11)) | - | NULL |
|  | setPositionCourante | test sur un EcoAgent quelconque | EcoAgent e ( EcoAgentID(id=13), etat = RS, positionCourante : EcoAgentID(id=3), positionFinale : EcoAgentID(id=11)) | positionCourante : EcoAgentID(id=7) | modification de positionCourante : l'identifiant de l'EcoAgentID prend la valeur 7 |
|  | getPositionFinale | positionFinale déterminée | EcoAgent e ( EcoAgentID(id=13), etat = RS, positionCourante : EcoAgentID(id=3), positionFinale : EcoAgentID(id=11)) | - | EcoAgentID(id=11) |
|  |  | positionFinale indéterminée | EcoAgent e ( EcoAgentID(id=13), etat = RS, positionCourante : EcoAgentID(id=3), positionFinale : NULL) | - | NULL |
|  | setPositionFinale | test sur un EcoAgent quelconque | EcoAgent e ( EcoAgentID(id=13), etat = RS, positionCourante : EcoAgentID(id=3), positionFinale : EcoAgentID(id=11)) | positionFinale : EcoAgentID(id=10) | modification de positionFinale : l'identifiant de l'EcoAgentID prend la valeur 10 |

1. **Classe Cube**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Classe | Méthode | Test | Donnees |  | Résultat attendu |
| Cube | rechercherFuite | test pour un cube qui est gêné | Cube a ( EcoAgentID(id=2),etat = RF, positionCourante : EcoAgentID(id=12), positionFinale : EcoAgentID(id=3) ) | Gêneur : Cube b ( EcoAgentID(id=5), etat = S, positionCourante : EcoAgentID(id=2), positionFinale : EcoAgentID(id=2) ) | lancement des méthodes : obtenirGeneur(a), qui retourne un pointeur sur le Cube b, puis agresser(b) |
|  |  | test pour un cube qui peut fuir | Cube a ( EcoAgentID(id=2),etat = RF, positionCourante : EcoAgentID(id=12), positionFinale : EcoAgentID(id=3) ) | aucun cube n'a pour position courante EcoAgentID(id=2) | lancement des méthodes : obtenirGeneur(a), qui retourne NULL, puis faireFuite() |
|  | rechercher Satisfaction | test pour un cube qui est gêné | Cube a ( EcoAgentID(id=2),etat = RS, positionCourante : EcoAgentID(id=12), positionFinale : EcoAgentID(id=3) ) | Gêneur : Cube b ( EcoAgentID(id=5), etat = S, positionCourante : EcoAgentID(id=2), positionFinale : EcoAgentID(id=2) ) | lancement des méthodes : obtenirGeneur(a), qui retourne un pointeur sur le Cube b, puis agresser(b) |
|  |  | test pour un cube qui peut être satisfait | Cube a ( EcoAgentID(id=2),etat = RS, positionCourante : EcoAgentID(id=12), positionFinale : EcoAgentID(id=3) ) | aucun cube n'a pour position courante EcoAgentID(id=2) ou EcoAgentID(id=3) | lancement des méthodes : obtenirGeneur(a), qui retourne NULL, puis faireSatisfaction |
|  | agresser | agression sur un cube quelconque | Agresseur : Cube( EcoAgentID(id=2),etat = RS, positionCourante : EcoAgentID(id=12), positionFinale : EcoAgentID(id=3) ) | Agressé : Cube( EcoAgentID(id=9), etat = RS, positionCourante : EcoAgentID(id=3), positionFinale : EcoAgentID(id=6) ) | le Cube agressé lance la méthode estAgressé() |
|  | estAgressé | Etat du Cube = S | Cube( EcoAgentID(id=15),etat = S, positionCourante : EcoAgentID(id=6), positionFinale : EcoAgentID(id=6) ) | - | Etat devient RF |
|  |  | Etat du Cube = RS | Cube( EcoAgentID(id=9),etat = RS, positionCourante : EcoAgentID(id=3), positionFinale : EcoAgentID(id=6) ) | - | Etat devient RF |
|  |  | Etat du Cube =RF | Cube EcoAgentID(id=2),etat = RF, positionCourante : EcoAgentID(id=12), positionFinale : EcoAgentID(id=3) ) | - | Etat devient RF |
|  | faireFuite | fuite d'un cube quelconque | Cube EcoAgentID(id=2),etat = RF, positionCourante : EcoAgentID(id=12), positionFinale : EcoAgentID(id=3) ) | - | positionCourante devient la Table |
|  |  |  |  |  | Etat devient RS |
|  | faireSatisfaction | satisfaction d'un cube quelconque | Cube a ( EcoAgentID(id=2),etat = RS, positionCourante : EcoAgentID(id=12), positionFinale : EcoAgentID(id=3) ) | - | positionCourante devient positionFinale |
|  |  |  |  |  | Etat devient S |
|  | initialiserEtat | position courante != position finale | Cube( EcoAgentID(id=10), positionCourante : EcoAgentID(id=24), positionFinale : EcoAgentID(id=1) ) | - | Etat devient RS |
|  |  | position courante = position finale | Cube( EcoAgentID(id=22), positionCourante : EcoAgentID(id=6), positionFinale : EcoAgentID(id=6) ) | - | Etat devient S |
|  | agir | test pour un cube en RF | Cube( EcoAgentID(id=2),etat = RF, positionCourante : EcoAgentID(id=17), positionFinale : EcoAgentID(id=8) ) | - | lancement de la méthode rechercheFuite() |
|  |  | test pour un cube en RS | Cube( EcoAgentID(id=9),etat = RS, positionCourante : EcoAgentID(id=3), positionFinale : EcoAgentID(id=6) ) | - | lancement de la méthode rechercheSatisfaction() |
|  |  | test pour un cube satisfait | Cube( EcoAgentID(id=15),etat = S, positionCourante : EcoAgentID(id=6), positionFinale : EcoAgentID(id=6) ) | - | Aucune action |

1. **Classe Table**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Classe | Méthode | Test | Donnees | Résultat attendu |
| Table | rechercheFuite | test trivial | Table | Aucune action |
|  | rechercheSatisfaction | test trivial | Table | Aucune action |
|  | agresser | test trivial | Table | Aucune action |
|  | estAgressé | test trivial | Table | Aucune action |
|  | faireFuite | test trivial | Table | Aucune action |
|  | faireSatisfaction | test trivial | Table | Aucune action |
|  | agir | test trivial | Table | Aucune action |

1. **Classe Singleton**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Classe | Méthode | Test | Donnees | Résultat attendu | Principe |
| template<typename> class Singleton | getInstance | obtention d'une instance unique | - | Les appels de plusieurs instances retournent la même instance unique | On appelle plusieurs fois getInstance() et on vérifie qu'on obtient bien la même adresse en mémoire |
|  | kill | destruction de l'instance NULL | - | Aucune destruction puisqu'aucune instance | On appelle kill sans avoir initialisé d'instance |
|  |  | destruction d'une instance existante | - | Destruction de l'instance | On crée une instance en faisant un getInstance puis on la détruit |

1. **Classe PlateformeEcoResolution**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Classe | Méthode | Test | Donnees | Résultat attendu | Principe |
| Plateforme EcoResolution | getEcoAgent | Recherche d'un ecoAgent inexistant | EcoAgentID (identifiant=70) | Pointeur NULL en retour | On essaye d'obtenir un ecoAgent à partir d'un ecoAgentID non referencé |
|  |  | Recherche d'un ecoAgent existant | EcoAgentID (identifiant=1) | Pointeur sur l'EcoAgent avec un EcoAgentID à 1 | - |
|  | addEcoAgent | Ajout d'un eco-agent déjà référencé | EcoAgent (EcoAgentID(identifiant=1)) | Exception "ExceptionEcoAgentDejaEnregistre" | - |
|  |  | Ajout d'un eco-agent non référencé | EcoAgent (EcoAgentID(identifiant=34)) | Ajout avec succès de l'agent | - |
|  | addRegle | Ajout d'une regle | Regle | La regle est ajoutée dans list<Regle\*> regles | - |
|  | verifierCoherence | Vérification de regles toutes vérifiées | 4 x Regles (verifier() -> TRUE) | TRUE | - |
|  |  | Verification de regles avec une regle non verifié | 2 x Regles (verifier() -> TRUE)  2 x Regles (verifier() -> FALSE) | FALSE |  |
|  |  | Verification de regles sans aucune regles | Aucune règle | TRUE | - |

1. **Classe PlateformeMondeDesCubes**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Classe | Méthode | Test | Donnees | | Résultat attendu | Principe |
| Plateforme MondeDesCubes | initialiser | aucun |  |  |  |  |
|  | resoudre |  |  |  |  |  |
|  | obtenirCubePrioritaire | obtention du cube prioritaire tous les cubes en RS | Cube a ( EcoAgentID(id=2),etat = RS, positionCourante : EcoAgentID (id=IDTABLE), positionFinale : EcoAgentID (id=3) ) | Cube b ( EcoAgentID(id=3), etat = RS, positionCourante : EcoAgentID (id=2), positionFinale : EcoAgentID (id=IDTABLE) ) | Cube b |  |
|  |  | obtention du cube prioritaire: mélange de RF et RS | Cube a ( EcoAgentID(id=2),etat = RS, positionCourante : EcoAgentID (id=IDTABLE), positionFinale : EcoAgentID (id=3) ) | Cube b ( EcoAgentID(id=3), etat = RS, positionCourante : EcoAgentID(id=2), positionFinale : EcoAgentID (id=IDTABLE) ) | Cube c |  |
|  |  |  | Cube c ( EcoAgentID(id=4),etat = RF, positionCourante : EcoAgentID(id=3), positionFinale : EcoAgentID(id=IDTABLE) ) |  |  |  |
|  |  | obtention du cube prioritaire alors que tous les cubes sont satisfaits | Cube a ( EcoAgentID(id=2),etat = S, positionCourante : EcoAgentID(id=IDTABLE), positionFinale : EcoAgentID(id=IDTABLE) ) | Cube b ( EcoAgentID(id=5), etat = S, positionCourante : EcoAgentID (id=12), positionFinale : EcoAgentID (id=12) ) | NULL |  |
|  | setNombreDeCubes |  |  |  |  |  |
|  | getNombreDeCubes |  |  |  |  |  |
|  | setTableIdentifiant |  |  |  |  |  |
|  | getTableID |  |  |  |  |  |
|  | verifierNombreDeCubes |  |  |  |  |  |
|  | obtenirGeneurs |  |  |  |  |  |
|  | addEcoAgent(Cube) |  |  |  |  |  |
|  | setPositionFinale (EcoAgentID,EcoAgentID) |  |  |  |  |  |
|  | setPositionCourante (EcoAgentID,EcoAgentID) |  |  |  |  |  |
|  | setCubeID (Cube,EcoAgentID) |  |  |  |  |  |