Documentation : Robot.java

Lignes 6 à 16 : énumération des actions (Mouvement)

Actions qui seront liées aux inputs de la future manette.

Lignes 19 à 22 : déclaration des attributs de la classe

(scanObjet initialisé à « false » pour la méthode « scan() »)

* this.name : nom du robot (String)
* this.type : type du robot (String)
* this.taille : sa taille en cm (int)
* this.android : si le robot est un androïde (boolean)
* this.mobile : si le robot peut se déplacer (boolean)
* this.powerOn : si le robot est activé (boolean)
* this.posX : sa position sur l’axe X (int)
* this.posY : sa position sur l’axe Y (int)
* this.direction : sa direction sur une base de 360° (int)

Lignes 24 à 35 : constructeur par défaut, créé un objet robot androïde standard, de 180cm, mobile et activé, en position X = 0, Y = 0 et avec une direction initialisée à 0 (regarde devant lui sur l’axe X).

Lignes 38 à 48 : constructeur avec paramètres, permet de créer un objet robot androïde ou non, de taille variable, mobile ou non et de déterminer sa position, sa direction étant initialisée à 0 (regarde devant lui sur l’axe X).

Lignes 50 à 70 : « getters » retournant le nom, le type de robot, sa position et s’il est activé ou non.

Lignes 74 à 80 : « setter » qui met en marche ou éteint le robot suivant son état (this.powerOn) à l’appel de la méthode « setPower().

Lignes 83 à 128 : méthode « setMouvement() » retournant un booléen à « true » si le robot est activé et peut donc effectuer une action, sinon à « false ». Contient un « switch-case » qui reprend l’énumération des actions du robot et les traite, en envoyant à une méthode spécifique pour chaque action.

Lignes 131 à 145 : méthode « setDirection() » détermine la direction du robot, sur une base de 360° découpée tous les 90°. Exemple : un robot est initialisé avec une direction à 0. S’il tourne une fois à droite, il sera à 90°. S’il avait tourné à gauche, il aurait été à 270°. S’il tourne deux fois dans la même direction, il sera à 180°.

Lignes 148 à 184 : méthode « setMove() », méthode principale concernant le mouvement du robot (s’il est mobile), appelée par la méthode « setMouvement() ». Elle contient un « switch-case » ayant la direction en paramètre, et applique le mouvement soit sur l’axe x soit sur l’axe y en fonction de celle-ci. Dans le cas où le robot n’est pas mobile (comme le mixeur), un message est affiché indiquant qu’il ne l’est pas.

Lignes 187 à 191 : méthode « scan », petite méthode simulant le scan de la zone par le robot (« this.scanZone »), préalable à la saisie d’un objet ou la réalisation d’une recette (par exemple : pour un robot mixeur).

Lignes 194 à 201 : méthode « takeObject() », simule la saisie d’un objet, au préalable scanné, par le robot. Retourne « true » si et seulement si le robot est mobile, a scanné la zone et ne porte pas déjà un objet (this.mobile, this.scanObject et this.objectTaken), sinon retourne « false ».

Lignes 204 à 211 : méthode « dropObject() », simule le fait que le robot lâche l’objet qu’il tient. Ne retourne « true » que si un objet est tenu (this.objectTaken)

Lignes 216 à 244 : méthode « agir() », contient un « switch-case » conditionné par « this.powerOn » et affiche un message selon le type de robot, simulant une action particulière. Si « this.powerOn » est false, affiche un message indiquant que le robot n’est pas activé.

Lignes 247 à 257 : méthode « destroyAllMankind() », méthode humoristique sur la peur de l’humanité de se voir annihilée par sa propre création artificielle. Le seul type de robot pouvant retourner « true » est le type « WarBot », sinon elle retourne false et laisse un répit à ses créateurs.