# I – Projet Roomba

Afin de respecter le cahier des charges, nous avons modélisé notre programme selon l’UML fourni en annexe.

Les obstacles et les taches peuvent être circulaires, carrés ou rectangulaires. Ces paramètres passent par l’interface graphique ou console.

L’environnement comprend une arène qui peut être carré ou circulaire. Le paramétrage est fait dans l’interface graphique, mais pour qu’il soit effectif en console, il doit être modifié dans le code.

Le robot peut être carré, circulaire ou rectangulaire et contenir des capteurs à souhait. Pour simplifier l’exercice, nous avons créé une classe roomba qui crée un roomba aux caractéristiques du vrai roomba. Cependant il est possible de simuler un autre robot en créant une nouvelle classe.

Le robot contient des capteurs qui sont de trois types : capteur de contact type poussoir, capteur de tache et capteur de distance laser. Il peuvent être ajoutés au robot en modifiant le code, et peuvent servir dans la modélisation de comportement. Il est aussi facile d’ajouter un nouveau type de capteur à partir de la classe abstraite Capteur

# II – Tester un comportement

## 1 – Ajouter un nouveau comportement

Afin d'être correctement intégré au projet, tout nouveau comportement doit être placé dans le package "com.roomba.comportement.Comportement"

De plus, la nouvelle classe "NouveauComportement" doit étendre la classe "com.roomba.Comportement". Cette dernière implémentant Cloneable, tout nouveau comportement doit ré-implémenter la fonction "public Object clone()".

Ceci est obligatoire afin de permettre une réinitialisation des paramètres initiaux du nouveau comportement.

Pour finir, la classe "NouveauComportement" devra implémenter la méthode abstraite "public void calcul (ArrayList<Capteur>)", et devra pour cela importer le package "java.util.ArrayList;"

Fonctionnement d'une classe Comportement :

Le roomba commandé par un comportement appellera à chaque pas de temps la fonction "public void calcul (ArrayList<Capteur>)", celle-ci devra mettre à jour les attributs :

"protected double iL"

"protected double iR", qui sont présent dans la classe "com.roomba.comportement.Comportement". Ces deux valeurs représentent l'intensité du courant envoyé aux moteurs attribué aux roues, respectivement Left et Right.

La vitesse des roues étant directement proportionnelle à cette intensité. La vitesse maximale de chaque roue est de 1m/s.

## 2 – Configurer le Roomba

Un nouveau comportement pouvant s'appuyer sur l'ajout de nouveau capteur, il est possible, et même obligatoire pour le bon fonctionnement du comportement, d'ajouter ces capteurs au roomba avant de lancer la simulation.

Pour cela, l'utilisateur devra ajouter ces lignes dans la classe "com.roomba.main.RoombaMain", dans la partie "configuration du roomba" :

"roomba.addCapteur(new NouveauCapteur(new FormeDuCapteur(\*),\*);"

La classe "NouveauCapteur" devra répondre aux normes explicitées dans la partie III.3. La classe "FormeDuCapteur" peut être soit une forme pré-existente ("Carre","Rectangle","Cercle"), soit une nouvelle forme respectant les normes explicités dans la partie III.1 .

## 3 – Tester un comportement

Afin de tester un nouveau comportement respectant les règles prédéfinies, l'utilisateur doit l'instancier dans la classe "com.roomba.main.RoombaMain", dans la partie "Instance du comportement test". le nom de la variable "comportTest" ne doit pas être changé car utilisé dans la suite de la classe.

# III – Ajouter des fonctionnalités

## 1 – Ajouter une nouvelle forme

Afin d'être correctement intégrée au projet, toute nouvelle forme doit être placée dans le package "com.roomba.shape".

De plus, la nouvelle classe "NouvelleForme" doit étendre la classe "com.roomba.shape.Forme", cette dernière implémentant "Cloneable", la classe "NouvelleForme" devra ré-implémenter la fonction "public Object clone()".

Ceci est obligatoire afin de pouvoir enregistrer l'état initiale d'un environnement, utilisé dans l'interface graphique.

La classe NouvelleForme devra donc implémenter l'ensemble des méthodes abstraites de "com.roomba.shape.Forme". Voir la JavaDoc pour la fonction de chaque méthode.

## 2 – Ajouter une nouvelle interface

Afin d'être correctement intégrée au projet, toute nouvelle interface être placé dans le package "com.roomba.ui".

De plus, la nouvelle classe "NouvelleInterface" doit étendre la classe "com.roomba.ui.Interface".

La classe "NouvelleInterface" doit donc implémenter les méthodes abstraites de la classe "com.roomba.ui.Interface". Voir la JavaDoc pour la fonction de chaque méthode.

## 3 – Ajouter un nouveau capteur

Afin d'être correctement intégré au projet, tout nouveau capteur doit être placé dans le package "com.roomba.robot".

De plus, la nouvelle classe "NouveauCapteur" doit étendre la classe "com.roomba.robot.Capteur".

La classe "NouveauCapteur" doit donc implémenter l'ensemble des méthodes abstraites de "com.roomba.robot.Capteur". Voir la JavaDoc pour la fonction de chaque méthode.