La guerre des robots

On veut modéliser un jeu qui simule le comportement de plusieurs équipes de robots prospecteurs en concurrence féroce sur une planète riche en minerai. Les robots peuvent tous percevoir leur environnement (les autres robots, les bases et les obstacles) ; les obstacles, les bases et les foreuses sont fixes, les foreuses extraient du minerai sur place ; elles peuvent tirer (dans la limite de leur portée) par exemple sur des obstacles pour faciliter le passage des porteurs ou sur des robots d'équipes adverses ; seuls les porteurs peuvent se déplacer, ils font la navette entre les différentes foreuses de leur équipe et leur base (la base unique de l'équipe) pour y rapporter le plus possible de minerai.
L'équipe gagnante est celle qui, à la fin du jeu, a le plus de minerai dans sa base.

Modélisez ce simulateur par un diagramme de classe UML ; placez au moins les propriétés suivantes :

- x, y ; coordonnées spatiales
- energie ; niveau d'énergie courant du robot
- MineraiBase ; quantité de minerai actuellement dans la base
- chargeMax ; Capacité de chargement maximum pour un porteur (elle est de 50 pour tous)
- distancePercep ; Rayon de perception maxi du robot (il est de 10 pour les porteurs et de 20 pour les foreuses)
- equipe ; nom de l'équipe d'appartenance du robot
- numéro ; numéro d'identification du robot à l'intérieur de son équipe
- couleur ; tout objet du jeu possède une couleur, noire pour les obstacles, dépend de son équipe pour un robot et pour les bases
- icône ; image jpeg représentant les différentes fonctions des objets (obstacle, base, foreuses, porteurs) ; ex : les foreuses d'équipes différentes ont la même icône et se distinguent par la couleur
- solidité ; elle est initialisée à 100 pour tous les obstacles, elle est variable selon les robots (paramètre de leur fabrication initiale) ; dans tous les cas, elle peut être entamée par des tirs adverses (cf ci-dessous).
- portée ; distance de tir maximum
- puissanceTir ; énergie maximum portée par un tir, plus l'énergie est forte, plus il abîme la cible (plus il lui retire de la solidité) mais plus il consomme d'énergie
- forer() ; retire du sol une quantité Q de minerai variable selon les foreuses
- percevoir() ; examiner son environnement dans un rayon de « distancePercep » et retourne la liste des objets perçus
- seDéplacer(deltaX,deltaY) ; évident
- rechargerBatteries(e) ; Ajoute une quantité e à l'énergie courante du robot
- chargerMinerai(Q) ; prendre en charge une quantité Q de minerai (pour les porteurs)
- déchargerMinerai(Q) ; décharger sur sa base une quantité Q de minerai (pour les porteurs)
- tirer(x,y,p) ; tirer sur le point (x,y) avec la puissance p
- vivre() ; effectuer, à chaque itération, un pas d'exécution (ex : percevoir, puis forer ou tirer pour une foreuse)
- lancerSimulation() ;
- stopperSimulation() ;
- vainqueur() ; retourne l'équipe gagnante
- ajouterRobot() ; ajoute un robot à la simulation

NB : Sauf indication contraire (p.ex.charge max et rayon de perception), les attributs des robots sont spécifiques à chaque exemplaire et sont précisés lors de la construction de chacun.

Travail à faire : (2 phases)

- Analyse : Faites tout d'abord le diagramme de classes d'analyse
- Conception :
  - Explicitez et justifiez vos choix de conception (pour chaque classe nécessitant une gestion d'instances, quelle est la classe qui l'effectue et pourquoi ; justifiez aussi l'orientation des relations).
  - Faites le diagramme de conception, ajoutez une classe "IHM" qui servira d'interface entre l'utilisateur et le système
  - Faites un diagramme de séquences pour chacune des requêtes suivantes :
    - Recharger d'une quantité e les batteries du robot N°7 de l'équipe "rouge"
    - Quel est le type (foreuse ou porteur), l'équipe et le niveau d'énergie du robot situé en x,y ? (on supposera qu'il n'y en a qu'un)
    - Ajouter un nouveau robot à la simulation (avec ses paramètres de construction)
    - Afficher l'équipe gagnante : son nom, sa couleur, sa quantité de minerai et le nombre de foreuses qui y sont rattachées

Pensez aux précisions suivantes :

- Typez les attributs, les paramètres, les valeurs de retour des opérations
- Préfixez correctement les attributs et méthodes selon leur nature : individuel, collectif (_s) ou commun (_c)
- Pensez aussi aux éléments abstraits (classes et méthodes)

Remarques :

- Bien sûr, ne vous préoccupez pas du contenu des opérations mais seulement des paramètres,
- Si vous ajoutez des propriétés, expliquez à quoi elles sont utiles.