Le drone s'est écrasé 2 fois durant QGL 2
19ème/21 durant la rétrospective
Il y a une régression dans le classement durant le semestre 2.

Le code compile.
76% de couverture de tests et pas de duplication (d'après sonar). Aucun test ignoré. La classe gérant la stratégie terrestre est testé à 35% ce qui est dangereux étant donné les responsabilités qui reposent sur celle-ci.

Dans l'ensemble, un trop grand nombre de classes n'ont pas de comportement et sont utilisées à des fins de stockages d'informations. Un important couplage entre les classes a également été relevé, ceci rend les tests particulièrement difficiles. Des enums ont été créé mais elles ne sont pas utilisé, ce qui rend la maintenabilité du code très difficile car il faudra remplacer ces chaînes de caractères dans toutes les classes.

IAAB/

- Utilisation du français et de l'anglais, il faut faire un choix et utiliser une seule langue
- ActionDataProcess :
  privilégier l'utilisation d'une interface (List à la place ArrayList)
  Dans le but d'accroître la lisibilité du code, il pourrait être envisageable de séparer l'écriture de la lecture du JSON.
- Le nom des variables et des méthodes ne sont pas explicites. (Par exemple: getContrat ne renvoie pas le contrat mais la quantité de ressource à récolter)
- Certaines classes qui semblent appartenir à des packages ne sont pas incluent dans ces packages (ActionDataProcess, DataProcess -> data)
- L'interface List devrait être préférée à la classe ArrayList pour permettre
- Explorer
  Succession de if/else if ce qui rend le code peu lisible.
- C'est une bonne pratique de documenter le code bien qu'il soit préférable d'utiliser la javaDoc plutôt que de simples commentaires.

IAAB/positions:

- Position:
  Cette classe semble être une "copie" de la classe Point. DronePosition et CrewPosition hérite de cette classe. L'utilisation de la classe Point de java aurait permis une simplification du code. Un objet Drone et Crew aurait pu

instancier un objet de type Point et gérer lui même ses propres changement de coordonnées.

IAAB/command

- Chaque classe permettant d'effectuer un déplacement change les coordonnées du drone ou des marins, ce n'est pas leurs rôles. De plus, vous utilisez 3 méthodes pour la modification des coordonnées alors qu'une seule aurait été suffisante.

- Chaque commande pouvant être exécuter par les marins ou le drone sont séparés ce qui est une bonne idée. Cela permet une meilleur modularité du programme si chacune de ses classes implémente une interface ou hérite d'une classe.

IAAB/command

- L'enum ressource est une bonne idée mais il aurait peut être fallu séparer les ressources primaires des ressources secondaires

IAAB/

- C'est une bonne idée d'avoir créé des recettes.

IAAB/Strategies/

- StrategyGroundManager:
  Il faut faire attention à la portée des variables (une variable a la porté public ce qui est contraire au principe de la programmation orienté objet)

KANBAN :

Les priorités sont mal définies (il est plus important de trouver une crique que de permettre le retour du drone). Il y a une incohérence entre les tickets : [IAAB-57] **Pouvoir crafter** est terminé alors que IAAB-14 **Transformer les ressources** n'est pas terminé.

De nombreux tickets n'ont pas de description ce qui ne permet pas de savoir quelles sont les limites du ticket. Cependant, les tickets en possédant une sont clairs. De plus, plusieurs tickets ne sont pas assignés ce qui ne permet pas de savoir qui travaille sur le ticket.

Certain ticket comme IAAB-45 sont “en cours” alors qu’ils sont fini.

L’équipe utilise les sous-tâches ce qui est une bonne chose.

Peu de bug (seulement 5) mais les explications de la plupart des bugs (1 bug sans description) sont clairs.