Checklist:

Required function	Implement
Documentation 40%	completed successfully
Implementation 30%	completed successfully
Runtime evaluation 30%	completed successfully

0. Environment details and how to run the program

System:

- Windows 10 Professional 20H2

Software version:

- Robocode 1.9.4.1
- Java 15 SDK for 'Robots' module

How to run program: (Download the ZIP file and unzip it)

1. Robocode import robot(recommend): Robot -> Import robot or team -> Select .jar -> New Battle
2. Robot Editor: New Robot -> Copy the code in and compile -> New Battle

1. Control Model Describe : Behavior Tree

The behavior tree is a tree structure containing logical nodes and behavior nodes. Each time a behavior needs to be found, the root node of the tree is started, and each node is traversed to find the first behavior that matches the current data.

Advantages:

- Referencing logical nodes, fewer conversion bars(Clear logic)
- Easy to extend AI behavior
- Behavior can be reused

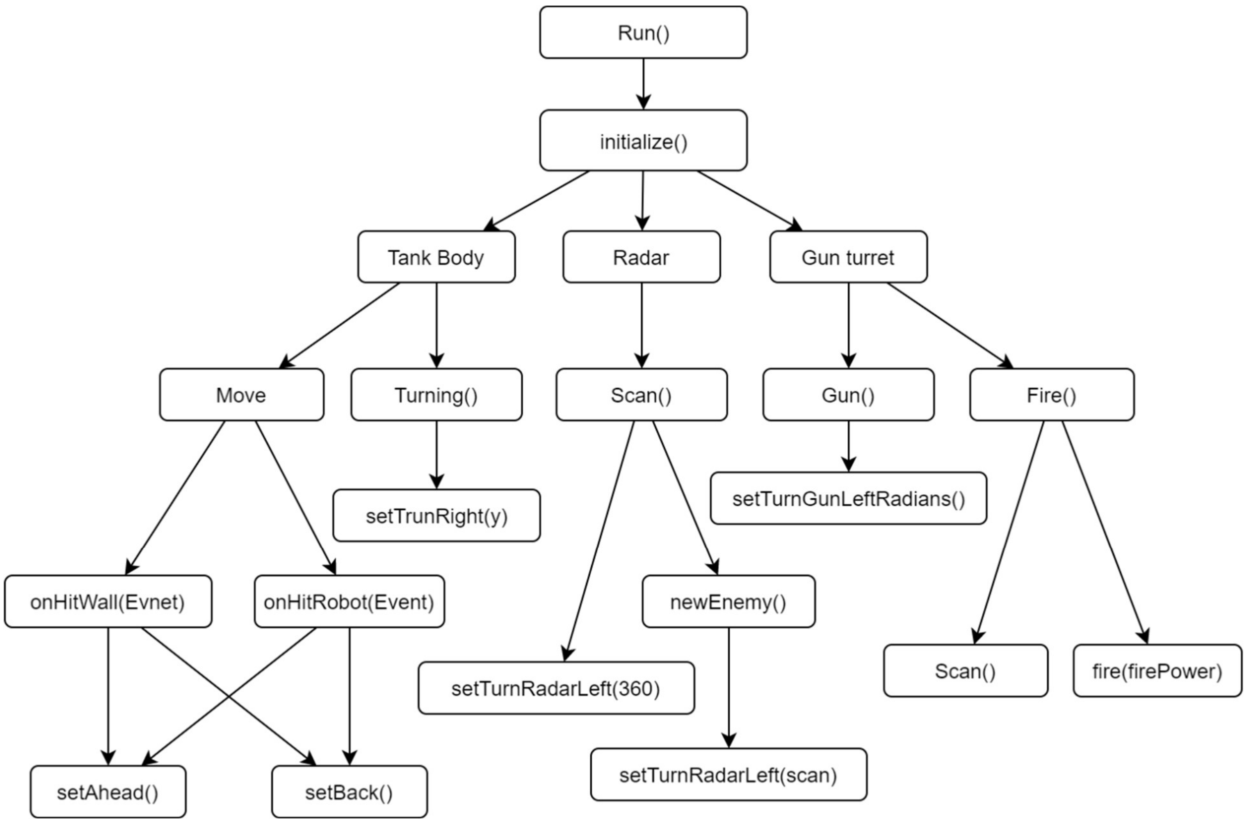
Structure:

1. Root Node
2. Logical Node
3. Behavior Node

2. Description AI design

The robot consists of three parts, namely the body, turret, and radar, all three of which can be rotated independently and can separately control some controls, such as forward distance, radar angle, firing, etc. Therefore, we roughly divide the logic into three logical.

Behavior Tree diagram:



Tank Body:

Responsible for movement, quote mathematical formula, make the tank move trajectory like a snake crawl sway, make the enemy difficult to predict the movement trajectory of this tank, and at the same time set the robot to detect the impact (hit the wall, hit the enemy) and immediately reverse the direction of movement. Greatly improve the survival rate of the tank in the battlefield

Radar:

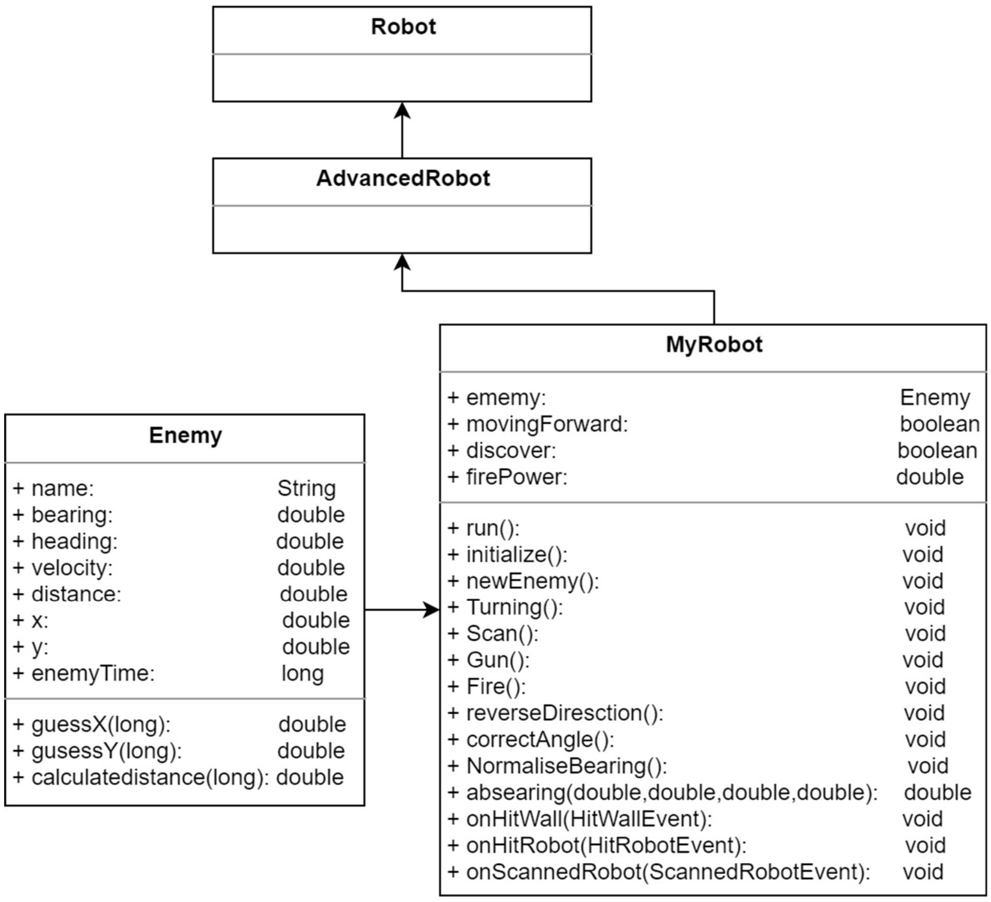
Responsible for scanning the battlefield for enemies, rotating 360 degrees if no enemy is scanned, until the enemy is scanned then locking the enemy direction, returning information about the enemy, position, direction of movement, speed, etc. Under the turret part of the behavior will call the enemy information.

Gun turret:

Responsible for rotating the muzzle and firing, the muzzle part calls radar to scan the information of the enemy, calculate and predict the trajectory of that enemy's movement through the formula, and set the angle of the muzzle to hit the advance. The firing part adjusts the fire size according to the local distance. Achieve the maximum use of firepower.

3. Implementation

UML diagram:



Class Description

MyRobot Class:

Each of the six different behaviors is controlled by multiple methods. `Turning()`, `Scan()`, `Gun()` and `Fire()` are called sequentially in `run()`, and each method has a different behavior pattern inside to implement the tank behavior transition.

Code part:

```
    public void run(){
        initialize();
        newEnemy();
        while(true){
            Turning();
            Scan();
            Gun();
            Fire();
            execute();
        }}
```

Enemy Class:

Record enemy information, three methods for calculating and predicting X,Y values and distance after enemy movement

Code part:

```
guessX(long time)
guessY(long time)
calculatedistance(long time)
```