BotWars API Documentation and How to Develop a Bot

How to start creating a Bot:

1. Create a Java project in the IDE of your choice.
2. Add the jar file "botwarlib.jar" to the build path (in case of Eclipse) or as a library (in case of Netbeans) of the project.
3. Create a public Java class, with the name same as your team name, in a new package (say *mypackage)* and inherit it from the base class *botwars.Bot.*
4. The base class must have a constructor and it should call the parameterized base class constructor with the name of the bot (viz. *super("MyBot")*).
5. The base class must implement the abstract method BotLogic(). This method contains the logic of your bot. It will be executed in a thread and generally has an infinite loop.
6. This class must also have a *public static void main(String args[]).* This method must have only one statement, A*rena.playGame("mypackage.TeamName", level)*. The first string in this method should contain the full name of the class (with package names) of your bot and the second parameter is the difficulty of the opponent bot (takes value 1 or 2 or 3).
7. Execute the project to start the war :-)

General Information:

- Objective of the game : To kill the opponent bot.
- The bots are pitted against each other in a rectangular arena of a fixed size (800 units x 500 units). Top-left corner is considered as origin (0,0).
- Bots have a size of 40 units.
- The basic unit of time is a tick and each tick is equal to 100 milli seconds.
- Each bot has an armour / health of 100 units. When a bot is damaged by a weapon, the armour / health is reduced.
- Bots have 3 functionalities - Movement, firing weapons and using a radar to sense the opponent.
- There is no acceleration in the bot, that is, if the bot has been assigned to be moving, it is immediately set into motion (50 units/sec). If the bot has been ordered to stop, it comes to an immediate halt.
- There are two types of weapons - bullets and missiles.
    - A bullet moves along the ground and will hit an opposite bot if it is in the path. If it does not encounter any bot (self or opponent), it will hit the arena's border and disappear. When a bullet hits a bot, a fixed amount of health is lost. Bullets are infinite ammunition.

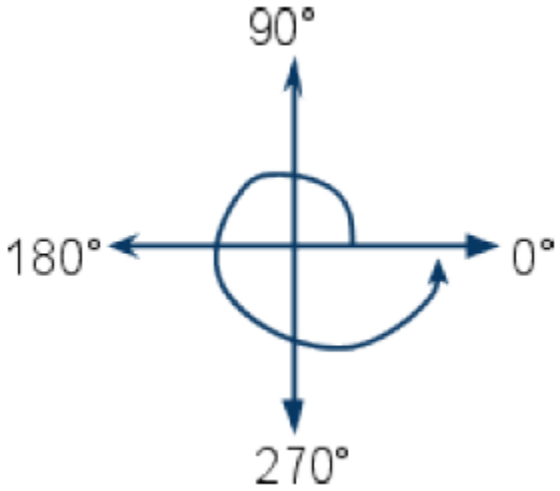
- A missile moves in the air and will not hit anything along the path of its motion (since it is in the air). It will land at a specified distance and explode. The explosion has a blast radius. If a bot is present within this radius, depending on the distance of the bot from the explosion, the amount of health will be reduced. Missiles have limited ammunition (50 missiles).

Note: A weapon can damage the person who initiated it. This is possible only while using a missile.

- Weapons can be fired in a particular direction.
- Weapons have a recoil time. Recoil time is the amount of time the bot has to wait before it can fire another bullet / missile. Each weapon has a different recoil time (Bullet - 5 ticks; Missile - 15 ticks).

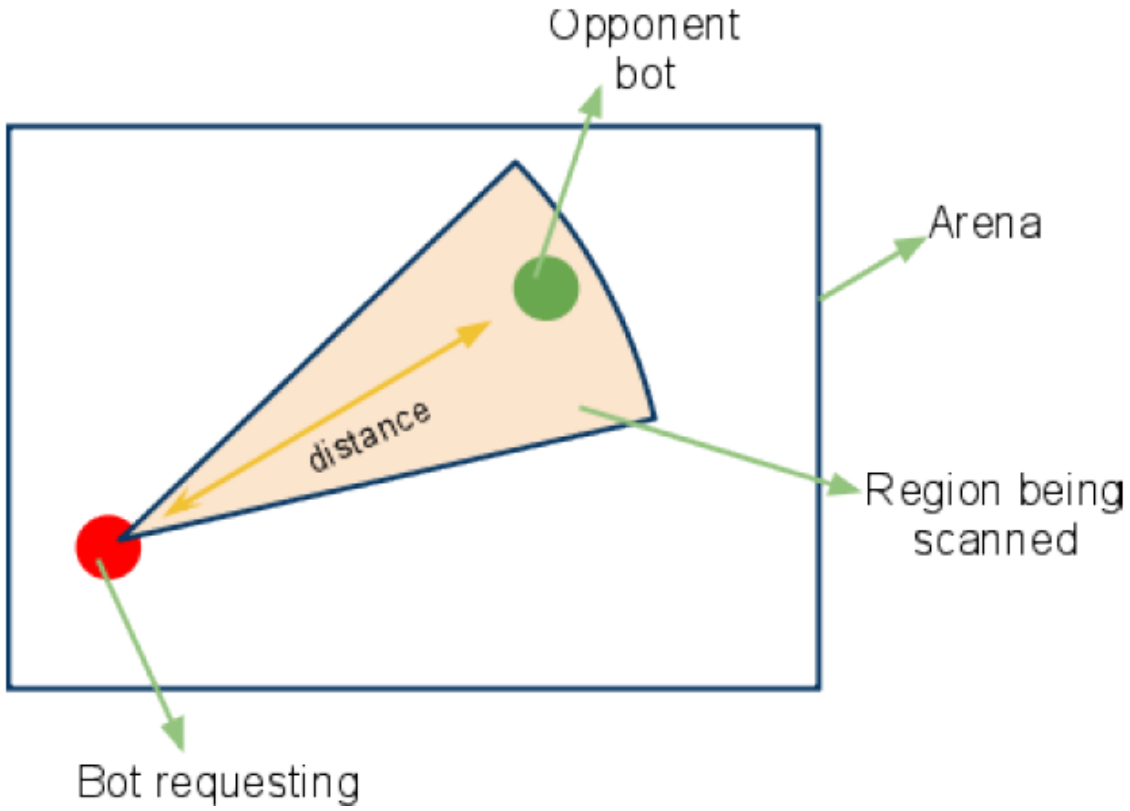
Note: When a weapon is fired, the bot has to wait for a time equal to the the recoil time to expire to fire any weapon, irrespective of which.

- Direction of firing of weapons is independent of direction or state of motion
- Directions are specified in terms of angle. Angles are measured anticlockwise and are absolute to the arena itself.



Note: Angles start from 0° to 359°

- The radar is used to scan an area to see if the opponent is present there. The radar scans a sector to check if an opponent bot is present, and if so, returns the distance between itself and the opponent. If the opponent bot is not present in the sector, then no data is regarding opponent's position is given to the bot.



- The radar scan can be performed for a maximum sector angle of 90°. The sector can start at any angle and end at any angle.
- When bots come in contact with the arena boundary, they will just stop irrespective of the angle in which they collide.
- When 2 bots collide, they stop moving.

API Documentation:

A blocking API is a method which will not return the control to the calling point until the action is completed. A non-blocking API is a method which will return immediately, and the intended action is performed in the background. Most of the APIs here are non-blocking but some are blocking. APIs have been done this way to improve the gameplay and to ensure equal chance for both bots.

- void moveBot(int angle)

  This method accepts an angle in which the bot should move. This method will not provide information if the bot is capable of moving in that direction. This method is non-blocking.

- void stopBot()

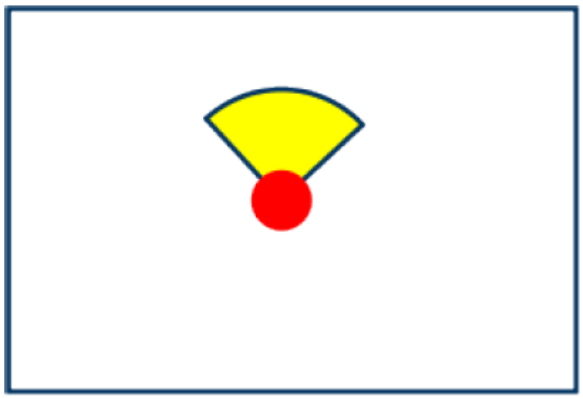
  This method stops the current motion of the bot and the bot stays at the current position. This method is non-blocking.

- int getCurrentDirection()

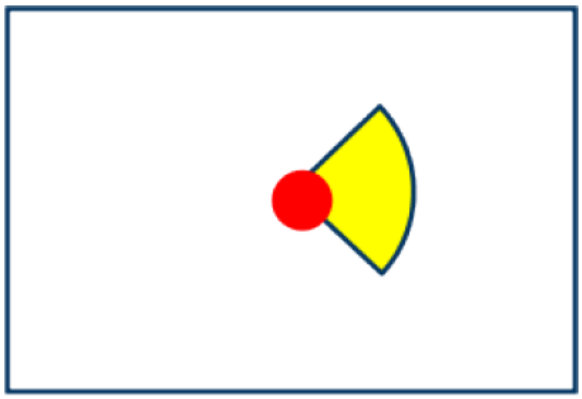
  This method returns the current direction in which the bot is moving. If the bot is stationary, it returns the last moving direction.

- int scanRadar(int start_angle, int end_angle)

  This method scans the region for an opponent bot. It returns the distance of the opponent bot from the requesting bot. If the bot is not present in the region, -1 is returned. If the scan sector angle is more than 90°, -1 is returned. Scan can be performed only when bot is stationary. If the scan request is sent when the bot is in motion, *-1* is returned from the method. Angles are measured counter-clockwise and so are the regions. This method is blocking. (5 ticks)



start_angle=45° end_angle=135°



start_angle=315° end_angle=45°

- boolean fireMissile(int fire_angle, int distance)

  This method fires a missile in the direction specified by *fire_angledistance* units. The distance can be a maximum of 500 units. A maximum of 50 missiles can be shot. The method returns *true*, if a missile has been fired successfully. The method returns *false* in the following cases :

  - If the distance specified is greater than the maximum distance.
  - If the missile ammo has been used up.
  - If the recoil time has not expired.

  This method is not blocking.

- boolean fireBullet(int fire_angle)

  This method fires a bullet in the direction specified by *fire_angle*. This weapon has infinite ammunition.

  - The method returns *true*, if a bullet has been fired successfully.
  - The method returns *false*, if the recoil time has not expired.

  This method is not blocking.

- Point getPosition()

  This method returns the position of the bot in the arena with the top-left point of the arena being (0,0).

- int getArmour()

  This method returns the armour / health of the bot. Armour values lie between 0 and 100, with 100 being full armour and 0 representing death.

- int getOpponentArmour()

  This method returns the armour / health of the opponent bot. Armour values lie between 0 and 100, with 100 being full armour and 0 representing death.

- int getRecoilTime()

  This method returns the amount of time left before the weapon can be fired again. If this value is 0, it means that the next weapon can be fired.

- int getMissileAmmo()

  This method returns the amount of missiles left remain to be fired.

- boolean isStationary()

  This method returns true if the bot is stationary, else returns false.

- int getTotalTicks()

  This method returns the total number of ticks for the entire game.

- int getTicksRemaining()

  This method returns the number of ticks remaining for the game to get over

Sample Code:

Assume,
Team Name : Survivor

Code:

```
package MyBot;

public class Survivor extends botwars.Bot {
        public Survivor(){
                super("Survivor Bot");
        }

        public void BotLogic() throws InterruptedException {
                while(true){
                        //The logic of your bot goes here.
                }
        }

        public static void main(String args[]){
                Arena.playGame("MyBot.Survivor",1);
        }
}
```

Note:

- The name of your public class must be the same as your Team Name(case sensitive).
- The name of the java file must be same as the name of your public class.
- You define any number of classes provided they are all in the same java file.

Contact us at izaazyunus@gmail.com, veeraaaa@gmail.com