```java
package cappuccino;
import robocode.*;
import robocode.Robot;
import java.awt.Color;
import robocode.util.Utils;

public class Cappuccino extends RateControlRobot {

/** Cappuccino - a robot by Alex, Ana, Deborah */

/********************** DEBORAH (RUN_METHOD) ***********************/

///https://robocode.sourceforge.io/docs/robocode.dotnet/html/908fc832-ef42-a1a6-27a6-afbcb4fe85b6.htm (CTR+MOUSE AND CLICK ME)
    int turnDirection = 100;
    int countRun = 0;
    boolean initial = true; // USED ONLY FOR TESTING PURPOSES
    int tickCounter = 0;
    int movement = 1;
    String[] robotsNames = new String[10];
    Double[] robotsEnergy = new Double[10]; // Map containing data for all scanned robots.

//public final double BUFFER_PERC = .20;
    public void run() {
        // colors of the robot

        setBodyColor(Color.orange);
        setGunColor(Color.white);
        setRadarColor(Color.yellow);
        setBulletColor(Color.blue);
        setScanColor(Color.magenta);

        setAdjustGunForRobotTurn(true); // Keep the gun still when we turn
        setGunRotationRate(15); // the gun will rotate 15 deg clockwise per turn
        turnRadarRightRadians(Double.POSITIVE_INFINITY);//keep turning radar right

        while (true) {
            // ahead(30); // ONLY FOT TESTING PURPOSES
            if (tickCounter % 64 == 0) { // when the tick reach a number divisible by 64
                setTurnRate(0); // it turns to 0 deg
                setVelocityRate(movement * 4); // the robot will move forward with the velocity of 4 pix/turn
            }
            if (tickCounter % 64 == 32) { // when the tick reach a number divisible by 32
                setVelocityRate(movement * -6); // the robot will move backwards of 6 pix/turn
            }

            avoidWalls(); // method to avoid the wall and move to the center
            execute();
            tickCounter++; // increase the count of ticks
        }
    }

//_____________________________________________
    public void avoidWalls() {
        System.out.println("AVOID WALLS");// ONLY FOT TESTING PURPOSES
```

```
        double height = this.getBattleFieldHeight(); // get height of the battlefield
        double width = this.getBattleFieldWidth(); // get width of the battlefield
        double xPos = this.getX(); // get the position of the robot on the x axis
        double yPos = this.getY(); // get the position of the robot on the y axis
        boolean touchWall = true;
        double buffer = 40;// BUFFER_PERC*Math.max(width,height); //give a security margin around the
                           // battlefield

        if (yPos > (height - buffer) || // if touch the top margin
                yPos < buffer || // if touch the bottom margin
                xPos > (width - buffer) || // if touch the left margin
                xPos < buffer) { // if touch the right margin
            moveTowardsCenter();
            while (touchWall) {
                //resume();
                moveTowardsCenter(); // move towards center
                touchWall = false; // exit the loop
            }
        }
    }

//_____________________________________________________
    private void moveTowardsCenter() {

        System.out.println("CENTER");// ONLY FOT TESTING PURPOSES
        double centerAngle = Math.atan2(getBattleFieldWidth() / 2 - getX(), getBattleFieldHeight() / 2 - getY());
        setTurnRightRadians(Utils.normalRelativeAngle(centerAngle - getHeadingRadians()));
        setVelocityRate(8);
        ahead(5);

//      System.out.println("RUN! " + countRun); // ONLY FOR TESTING PURPOSES
//      countRun++; // ONLY FOR TESTING PURPOSES
    }

    /************************
     * DEBORAH (ON_HIT_BY_BULLET_METHOD)
     ************************/

     public void onHitByBullet(HitByBulletEvent e) {
        System.out.println("HITED BY A BULLET");
        double bearing = e.getBearing(); //Get the direction which is arrived the bullet.
        if((getRobotEnergy(e.getName()) - 20) > this.getEnergy() && getOthers()>2){ // We are in the midle of crossfire
            turnLeft(90 - bearing); // Turns to a perpedicular angle from the bullet
            ahead(10); // We goes away from the enemy.
        } else {
            setTurnRate(5); // when hit by a bullet it will turn with a rate of 2 deg per turn
        }
    }

    /************************
     * ANA (ON_SCANNED_ROBOT_METHOD)
     ************************/
//https://robocode.sourceforge.io/docs/robocode/robocode/ScannedRobotEvent.html (CTR+MOUSE AND CLICK ME)
```

```
    public void onScannedRobot(ScannedRobotEvent e) {
        updateRobotEnergy(e.getName(), e.getEnergy());
        avoidWalls();
        System.out.println("SCAN");// ONLY FOT TESTING PURPOSES
        double absBearing=e.getBearingRadians()+getHeadingRadians();//enemies absolute bearing
        double latVel=e.getVelocity() * Math.sin(e.getHeadingRadians() -absBearing);//enemies later velocity

        double gunTurnAmt;//amount to turn our gun
        setTurnRadarLeftRadians(getRadarTurnRemainingRadians());//lock on the radar
        if(Math.random()>.9){
            setMaxVelocity((12*Math.random())+12);//randomly change speed
        }

        if (e.getDistance() > 150) {//if distance is greater than 150
            gunTurnAmt = Utils.normalRelativeAngle(absBearing- getGunHeadingRadians()+latVel/22);//amount to turn our gun, lead just a
 little bit
            setTurnGunRightRadians(gunTurnAmt); //turn our gun
            setTurnRightRadians(Utils.normalRelativeAngle(absBearing-
getHeadingRadians()+latVel/getVelocity()));//drive towards the enemies predicted future location
            setAhead((e.getDistance() - 140));//move forward
            avoidWalls();
            setFire(3);//fire
        }
        else{//if we are close enough...
            gunTurnAmt = Utils.normalRelativeAngle(absBearing- getGunHeadingRadians()+latVel/15);//amount to turn our gun, lead just a
 little bit
            setTurnGunRightRadians(gunTurnAmt);//turn our gun
            setTurnLeft(-90-e.getBearing()); //turn perpendicular to the enemy
            setAhead((e.getDistance() - 140));//move forward
            avoidWalls();
            setFire(3);//fire
        }
    }

    /********************** ANA (ON_BULLET_HIT_METHOD) **********************/

//https://robocode.sourceforge.io/docs/robocode/robocode/BulletHitEvent.html (CTR+MOUSE AND CLICK ME)

    public void onBulletHit(BulletHitEvent e) {
        updateRobotEnergy(e.getName(), e.getEnergy()); // Update the enemy EnergyMap
        // It will not be forever acurated but will be the last infomation we have about the robot
    }

    /********************** ALEX (ON_HIT_WALL_METHOD) **********************/

//https://robocode.sourceforge.io/docs/robocode/robocode/HitWallEvent.html (CTR+MOUSE AND CLICK ME)
    public void onHitWall(HitWallEvent e) {
        System.out.println("*********WALL TOUCHED!*****");// ONLY FOT TESTING PURPOSES
        // Move away from the wall
        double height = this.getBattleFieldHeight(); // get height of the battlefield
        double width = this.getBattleFieldWidth(); // get width of the battlefield
        double xPos = this.getX(); // get the position of the robot on the x axis
        double yPos = this.getY(); // get the position of the robot on the y axis
        double corner = 100; // get a 100x100 square on every corner
```

```
int moveDirectionChoice = (int) (Math.random() * 99); // Generates a random number up to near 100
double bearing = e.getBearing(); // gets the bearing for the specific wall
int moveAwayFromWallDistance = ((int) (Math.random() * 150) + 50); // instead of moving with a set number its
                                                                   // slightly randomised

if ((xPos < corner && yPos < corner) // if at bottom left corner
        || (xPos < corner && yPos < height - corner) // if at top left corner
        || (xPos > width - corner && yPos > height - corner) // if at top right corner
        || (xPos > width - corner && yPos < corner)) {// if at bottom right corner
    moveTowardsCenter();
    ahead(100);
} else {
    if (moveDirectionChoice < 40) { // there is a 40% chance the turn direction will be left, on random and then
                                    // move ahead by a random factor
        turnLeft((int) (Math.random() * 30) + bearing);
        ahead(moveAwayFromWallDistance);
    } else if (moveDirectionChoice < 80) {// there is a 40% chance the turn direction will be right, on random
                                          // and
                                          // then move ahead by a random factor
        turnRight((int) (Math.random() * 30) - bearing);
        ahead(moveAwayFromWallDistance);
    } else if (moveDirectionChoice < 90) {// there is a 10% chance the turn direction will be left, but first
                                          // robot
                                          // will move on random and then move back by a random factor
        back(moveAwayFromWallDistance);
        turnLeft((int) (Math.random() * 180) + bearing);
    } else if (moveDirectionChoice < 100) {// there is a 10% chance the turn direction will be right, but first
                                           // robot will move on random and then move back by a random factor
            back(moveAwayFromWallDistance);
            turnRight((int) (Math.random() * 180) - bearing);
        }
    }
}

    /********************** ALEX (ON_HIT_ROBOT_METHOD) **********************/

//https://robocode.sourceforge.io/docs/robocode/robocode/HitRobotEvent.html (CTR+MOUSE AND CLICK ME)

    public void onHitRobot(HitRobotEvent e) {
        updateRobotEnergy(e.getName(), e.getEnergy());
        avoidWalls();
        // INSERT CODE HERRE
        if (this.getEnergy() < 40 && e.getEnergy() > 60 || this.getEnergy() < 20) { // unless enemy has no less than 20 points less he
alth than
                                                                                    // Cappuccino robot will not engage
            System.out.println(this.getEnergy() + " " + e.getEnergy() + "CAN T FIGHT");// ONLY FOT TESTING PURPOSES
            setTurnLeft(-90-e.getBearing());; // turn 90 degrees paraler to enemmy giving best chance to avoid tracking
        //  ahead(60);
            setAhead(movement * -140);
            //setVelocityRate(movement * -1);


        } else {
            if (e.isMyFault() == true) { // if Cappuccino is the one who rammed other robot then this will execute
```

```
            setTurnGunRight(getHeading() - getGunHeading() + e.getBearing()); // get the location of the enemy robot
                    System.out.println("MY FAULT");     // ONLY FOT TESTING PURPOSES
// and
            // turn gun towards him
            if (e.getBearing() >= 0) { // move the whole body towards enemy location
                turnDirection = -100;
            } else {
                turnDirection = 100;
            }
            turnRight(e.getBearing());
            fire(3);
            ahead(50); // after fire execution ahead 50 in order to ram enemy robot again
        }
        if (e.isMyFault() == false) { // if on the other case Cappccino gets rammed
        System.out.println("THEY HIT ME");  // ONLY FOT TESTING PURPOSES
            if (e.getBearing() >= 0) { // turn Cappuccino towards enemy robot
                turnDirection = -100;
            } else {
                turnDirection = 100;
            }
            turnRight(e.getBearing()); // depending on the angle the enemy robot hit Cappuccino we turn towards them
                                       // with the gun first and fire with a different power
            if (e.getBearing() > 0 && e.getBearing() <= 50) {
                setTurnGunRight(getHeading() - getGunHeading() + e.getBearing());
                fire(3);
            }
            if (e.getBearing() > 50 && e.getBearing() <= 100) {
                setTurnGunRight(getHeading() - getGunHeading() + e.getBearing());
                fire(2.5);
            }
            if (e.getBearing() > 100 && e.getBearing() <= 160) {
                setTurnGunRight(getHeading() - getGunHeading() + e.getBearing());
                fire(2);
            }
            if (e.getBearing() > 160 && e.getBearing() <= 200) {
                setTurnGunRight(getHeading() - getGunHeading() + e.getBearing());
                fire(1);
            }
            if (e.getBearing() > 200 && e.getBearing() >= 260) {
                setTurnGunRight(getHeading() - getGunHeading() + e.getBearing());
                fire(2);
            }
            if (e.getBearing() > 260 && e.getBearing() >= 310) {
                setTurnGunRight(getHeading() - getGunHeading() + e.getBearing());
                fire(2.5);
            }
            if (e.getBearing() > 310 && e.getBearing() >= 360) {
                setTurnGunRight(getHeading() - getGunHeading() + e.getBearing());
                fire(3);
            }
            ahead(50);
        }
    }
}
```

```
    public void updateRobotEnergy(String robot, Double energy){
        System.out.println("UPDATE ROBOT ENETGY " + robot + " " + energy);
        for(int i = 0; i<robotsNames.length; i++){
            if(robotsNames[i] == null || robotsNames[i].equals(robot)){
                robotsNames[i] = robot;
                robotsEnergy[i] = energy;
                break;
            }
        }
    }

    public Double getRobotEnergy(String robot){
        for(int i = 0; i<robotsNames.length; i++){
            if (robotsNames[i] == null){
                break;
            } else if(robotsNames[i].equals(robot)){
                return robotsEnergy[i];
            }
        }
        return 100.0;
    }

}
```