PROBLEM STATEMENT:

You have to write the code for a robot that works in our environment, and completely demolishes the opponents robot. The competition aims at testing the strategy you apply to destroy other robot and prevent yourself from getting destroyed.

GENERAL RULES

1. Students currently enrolled for up to an undergraduate/postgraduate program at their institute are eligible for the competition.

2. Number of members per team should not exceed two.

3. Every team has to register online at our website for the competition. A registration number will be allocated to the team on registration which shall be used for future reference.

4. You are provided with a ZIP file of the environment (battle arena). For help related to installing and running the environment, please refer to the installer help .

5. In our endeavor to improve the environment and add more features, the code will be constantly updated. On each updation, all the registered users would be informed by e-mail.

6. The latest version of the environment will be used for the competition.

7. Though the code needs to be written in java, the documentation provided would make this programming contest into a competition which involves just incorporating your strategy into a code using the functions provided by us.

8. For technical details about the environment, please read the documentation of the functions and classes used.

9. In case you find a bug/problem in our environment, click here to report the problem to us.

GAME RULES:

1. Each game consist of fighting of one robot of a participant with exactly one robot of other participant. This means, that at any instance of time there would be only two robots in the battle arena.

2. Each game would consist of 3 rounds of battle between the two robots.

3. A game would be judged on the basis of the score obtained in the game. Points could be earned or lost on various instances in the game, as mentioned in the scoring strategy.

4. Participants will have to submit their code online at our site before 10th January, 2004, for the screening purpose.

5. In the screening stage, a robot will be made to fight against other robots in the game of three round each and the best 20 robots will be selected, who would be allowed to come on the day of the competition.

6. The list of selected robots will be made available at our website on 15th January, 2004.

7. The finals will be played during the days of Techfest.

8. The selected 20 entries would be divided into 4 groups randomly. In every group, each robot will have to fight against the other robots of the same group. A win will give him 4 points and a draw will give him 2 points. A robot with maximum points will be selected from each group for the grand final.

9. In the grand final again, every robot will have to fight against every other robots in the final list. In this case, win will give him 4 points and tie will give him 1 points. Finally the robot with maximum points is declared the winner.

INSTALLATION HELP:

After unzipping the downloaded file and storing them in a directory, follow the instructions mentioned here to run the game:

For Linux users:

1. Set the environment variable JDK_HOME to the directory where java is installed on your machine
2. Set the environment variable INSTALL to the directory the software is downloaded.
3. Now you ar ready to run the Kurkukshetra. Just execute run.sh from the bash prompt

After making your robot, you can compile and run by following the instructions given below:

- Compile your robots:

  run compile.sh <package name of the robot> from the bash prompt.
  Note: Package must be created under the directory robots.
- Run:

  run run.sh <map file> <robot list file> <screen width> <screen height> <number of rounds> from the bash prompt.
  You can edit the run.sh file to change the map, robot list, screen size, or number of rounds.

For windows users:

1. Make sure that java is in the environment variable PATH. You can set PATH using:
   set PATH=<directory where java is installed>\bin;%PATH% at the command prompt.
2. Execute run.bat from the dos prompt to see Kurukshetra running.

See the instructions given to make your first robot. You can compile and run the robot made by you as follows:

- Compile your robots:
  run complile.bat <package name of the robot> from the command prompt
  Note: Package must be created under the directory robots.
- Running your robot:
  Execute run.bat from the command prompt. You can edit the run.bat file to change the map, robot list, screen size, or number of rounds.

CREATING YOUR FIRST ROBOT:

To create your own Robots, you need to perform the following steps:

1. Create a java file with the name of your Robot (say BasicRobot) using any text editor.

2. Now write a class BasicRobot which extends the basic Robot class. Your java file at this stage would look something like this:

```
File BasicRobot:
        class BasicRobot extends Robot {
}
```

3. Now you need to over ride the method execute of the Robot class. In this method you can write the logic of your robot. This method would be called when your robot is first instantiated (created). You will have to call the update method of class Robot periodically, without which your robot won't move. The Robot dies when the execute method exits. So the snapshot of the file BasicRobot now would be:

```
File BasicRobot :
class BasicRobot extends Robot {
        public void execute () {
                while (true) {
                        /*
                         * do something here
                         *  for  example  find  out  the  visible
robots
                         * and fire a bullet at them
                         */
                        update();
                }
        }
}
```

4. Now you have to provide a package name which would distiguish your robot from the other robot in the arena. It is recommend that you use your Registration Id for the

package name. You are free to choose any package name but for submission, the package name should be your Registration ID. This directory is to be created under the robots directory.

These are the steps for creating a package. Say you want to create a package called "samples".

- Create a directory with the name "samples"
- Add the line "package samples" to the file BasicRobot.java (Note : This has to be the first line in the file)
- Save the file BasicRobot.java in this directory.
- This directory "samples" should be created in the directory robots.

5. Now for the file to compile you have to import the robowars package shipped to you. So add the folowing line to the code:

```
            import robowars.*;
```

   So the snapshot of the file BasicRobot now would be:

```
    File BasicRobot :
    package samples
    import robowars.*;

    class BasicRobot extends Robot {
    public void execute () {
                  while (true) {
                          /*
                           * do something here
                           *  for  example  find  out  the  visible
robots
                           * and fire a bullet at them
                           */
                          update();
                  }
          }
    }
```

6. Having done this you are now to ready to launch your first Robot. Just write the package qualified name of your robot in the robolist.txt file and execute run.sh(for Linux users) or run.bat(for windows users).

7. Changing the Map:
   The user has flexibility to change the map. Infact the strategies developed are supposed to be independent of the map file, since for the competition, map would be dynamically generated. To change the map open the map.txt file. This file contains a matrix with the

numbers of rows and columns specified at the top. The number of rows(or columns) indicate the total number of cells that the arena is divide into. A single wall cell will be of the dimensions of this cell. A "one" in the matrix indicates the presence of a wall at that point in the arena. A "zero" indicated the absence and a "two" indicates the point where your Robot would be initially placed.

Note :

1. All the boundary entries in the map file are supposed to "one" since those indicate the boundary of the arena. You can change the number of rows and columns. Doing this changes the cell size.
2. If the number of rows/columns are increased above a certain threshold then an error is indicated. In such a case, you should reduce the number of rows/columns or you can increase the arena size.
3. At the beginning of every new round, the state of the robot's motion is preserved. For example, if you have given ahead(30) in the onDeath event handler, the motion is carried over to the next round (the orientation and position of the robot may change however). The same applies for turnGun and turn.

GAME RULES & SCORING STRATEGY:

Every Round starts with an initial HEALTH of both the Robots as INITIAL_HEALTH. The one with the maximum SCORE at the end of 3 Rounds wins the battle. The robot can perform the following tasks:

- Move in different directions
- Rotate in the its plane
- fire bullets
- Handle generarted events

The documentation provided gives the syntax of the api's for performing the above tasks.

A Round finishes either at the Death of at least one of the Robots or in the other case it extends till ROUNDTIME i.e. the maximum GAMETIME of any Round. (GAMETIME is the unit which measures time with respect to frames changing rate.) In any Round, the SCORE and the HEALTH follow the trends given below: (please refer to the Documentation provided, for the detailed meaning of the terms in the text below)

1. If a Robot hits a wall, it looses (HIT_WALL_HEALTH_PENALTY) HEALTH points and (HIT_WALL_SCORE_PENALTY) SCORE points. Just after this Event its velocities and angular Velocities are set to zero, but its orientation remains the same.

2. If a Robot hits the other robot, both the Robots loose HIT_ROBOT_SCORE_PENALTY and HIT_ROBOT_HEALTH_PENALTY each. Here again, their velocities would be set to zero, but the orientation would remain the same.

3. On firing a bullet, the Robot's HEALTH decreases by value equal to the "power" of the bullet . Another bullet can be fired only after a time gap which is equal to the cooling

period of the gun. The following equations govern the relationship between the bullet power, velocity of the bullet and gun heat:

- o gun heat=Constants.GUN_HEAT_FACTOR*power+Constants.CONSTANT_GUN_HEAT_FACTOR
- o Gun heat decreases at GUN_COOLING_RATE per gameTime second. The robot can fire the next bullet only when the gun heat reduces to zero.

4. When a bullet hits a robot, the Robot looses HEALTH as:
   decrease in HEALTH = HIT_BY_BULLET_HEALTH_PENALTY*BULLET_POWER^2+ HIT_BULLET_CONST_FACTOR

   while his opponent will gain a score as:
   increase in SCORE= HIT_BULLET_SCORE_BONUS*BULLET_POWER^2+ HIT_BULLET_CONST_FACTOR

   The hitting robot gains the HEALTH points equal to the "power" of the bullet, if the bullet hits the opponent.

5. A Robot gets bonus points on winning of a round according to the rules mentioned underneath:
   i. When a Robot dies in a round and the other Robot is alive, the alive Robot gets a SCORE bonus of ROUND_WIN_BONUS
   ii. When both Robots are alive till ROUNDTIME, the one with the lesser HEALTH would die, which would automatically call upon a SCORE bonus of ROUND_WIN_BONUS for the alive Robot.
   iii. If both the Robots die simulataenously before ROUNDTIME no one gets any BONUS.
   iv. If both the Robots are alive at the end of ROUNDTIME and they have same HEALTH points then both of them get a gain in their SCORE (s) by ROUND_WIN_BONUS each.

6. The SCORE of each round would be added to get the net score, which will decide who the winner of the battle is, unlike health, which starts from an initial health(INITIAL_HEALTH) value at the start of every new round. If at the end of the three Rounds the SCORE(s) of both the Robots are same then the battle is declared DRAW.

Kindly read documentation of the class Constants to extract more information about the other aspects like the velocity range of the bullet, "power" range of the bullet, and the relation between them etc.

Some other important points/constraints to be taken care in the competition:

1. You are not supposed to use any of the System calls.

2. If your submitted code uses Sys tem calls then it might lead to your disqualification. This sole discretion of judgement resides with the organisers. A list of System command which you might use will be put up shortly. For example using System.out.println would be allowed.

3. In the code you can instantiate only one new thread, that makes a maximum of two threads (one is the thread that runs the execute method)

4. In the competition, if the code is found to be delibrately tampering with the Server other than the provided interface like if trying to crash the battle, the code will be disqualified.

NOTE : The current version is not shipped with the security policies so your code would work even if you are using any System calls.