Domain Model and Unit Tests

Project

UARobotFight

Team

UARobotFight

Author(s)

<list of contributing authors - name, id number and email>

name	id	email
Rui Garcia	41630	ruigarcia@ua.pt

Revision(s)

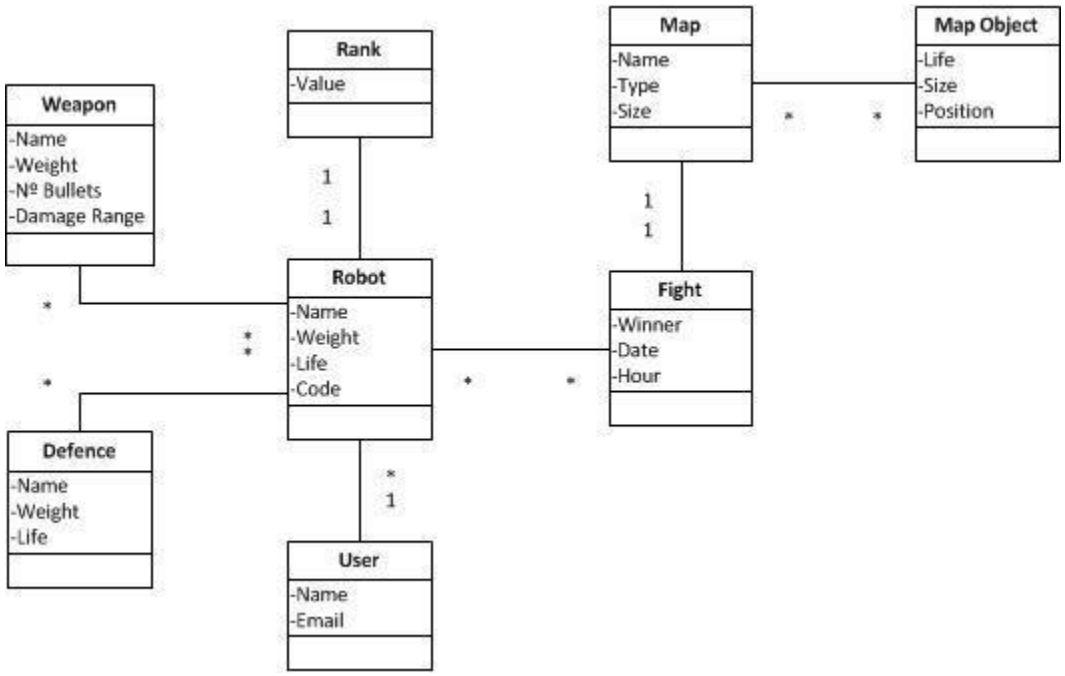
< contains the list of changes - when, who and what was changed>

date	Who	What
March 5, 2013	Rui Garcia	All

Table of Content

Domain Model and Unit Tests
Table of Content
The domain model
The concepts
Unit Tests

The domain model



The User class is responsible for storing users information. Each user can have more than one robot (Robot class), each one with an unique rank. Any robot can have multiple weapons (Weapons class) and “defences” (Defence class). A fight (Fight class) will occur between two robots, from different two users, in a specific map (Map class), which can have several objects (Map Objects class) like walls or obstacles.

The concepts

Robot

Responsability

- Represent the virtual agent which the user will program to fight with others robots.

Attributes

- Robot name;
- The robot’s entire weight, including the weight of the weapons and defenses installed;
- Initial life (health points) of the robot;
- The code that the user enters in the robot.

Functionalities / methods

- Add, edit and delete a robot;
- Modify the code associated with the robot;
- Entitle the robot for a random fight.

Requirements / Observations

- Each robot must be ranked.

Weapon

Responsability

- Represent a specific weapon that the robot can use during a fight to strike an opponent.

Attributes

- Weapon name;
- The weight of the weapon;
- The number of bullets;
- Damage range of the weapon.

Functionalities / methods

- Installs a weapon on a robot;
- Drop a weapon from a robot;

Requirements / Observations

- The weapons will affect the weight of the robot.

Defence

Responsability

- Represent a specific defence that the robot can use during a fight.

Attributes

- Defence name;
- The weight of the defence;
- The defence life.

Functionalities / methods

- Installs a defence on a robot;
- Drop a defence from a robot;

Requirements / Observations

- The defences will affect the weight of the robot.

Rank

Responsability

- Each robot will have a unique rank, depending on the results obtained in his fights.

Attributes

- The rank value.

Functionalities / methods

- Change robot rank.

Requirements / Observations

- The rank value must be unique.

Map

Responsability

- Provide a virtual space/terrain where the robots will fight.

Attributes

- Map name;
- Type of the map;
- Size and limits of the map.

Functionalities / methods

- Generate random objects in the map.

Requirements / Observations

- The map should be limited.

Map Object

Responsability

- Define an object that is part of a map, like walls and obstacles.

Attributes

- Life of the object;
- Size of the object;
- Its position on the map.

Functionalities / methods

- Add an object to the map;
- Edit the position of the object on the map;
- Drop an object from the map.

Requirements / Observations

Cannot have an object larger than the size of the map.

Fight

Responsability

- Describe an fight between two robots and its results.

Attributes

- Winner robot of the fight;
- Date and time of the fight.

Functionalities / methods

- Add an object to the map;
- Edit the position of the object on the map;
- Drop an object from the map.

Requirements / Observations

- The end results should be displayed to the user.

Unit Tests

Robot

- The robot must be associated with a user;
- The weight and life values must be greater than zero;
- The robot must have a name.

User

- The user must be authenticated for manage their robots.

Weapon

- The weight value must be greater than zero;
- If the number of bullets is limited, the gun cannot fire more than this limit.

Defence

- The weight value must be greater than zero;

Map

- The map should be limited.

Map Object

- Cannot have an object larger than the size of the map.

Rank

- The rank value must be unique.
- In the end of a fight, the rank value should be updated.

Fight

- The end results should be displayed to the user.