PROJECT DESIGN REPORT

"BEAT"

IE 201.01

Group No: 2

Onur Alpağan

Faruk Tüfekçi

Furkan Sezer

Instructors: Ali Tamel Ünal, Caner Taşkın

December 25th, 2016

INDEX

1. Class Diagram

1) Manager class

Manager class is responsible for the initialization and maintenance of the game. At the beginning of the game, it will draw all game objects. It determines when and where the bonuses occur.

2) Rectangle Class

This class is a parent class for Ball, Bonus and Bar classes. It has two Vector class object as attribute for size and position of the Rectangle. It also has a color attribute and draw function.

3) Player class

Player class is an abstract base class of HumanPlayer, ComputerPlayer classes. After player names are entered, the game begins and opponent will be ComputerPlayer and it starts at the upper position. The game has also user controlled HumanPlayer. It has name, fuel and score attributes and also a bar object.

4) HumanPlayer and ComputerPlayer Classes

These are subclasses of Player class. HumanPlayer expects keyboard hits from user in order to tell bar class to move parallel to goal line; to decide direction of launch. ComputerPlayer follows an algorithm to be opponent.

5) Ball Class

It is a subclass of Rectangle class so it has position and size attributes. Also it has velocity attribute as a vector object. It has IsIntouchWithBar, IsIntouchWithBonus and IsOutOfScope functions. IsOutOfScope function check if a goal occurs.

6) Bar Class

It is a subclass of Rectangle class. We can move our bar parallel to goal line by move function.

7) Bonus Class

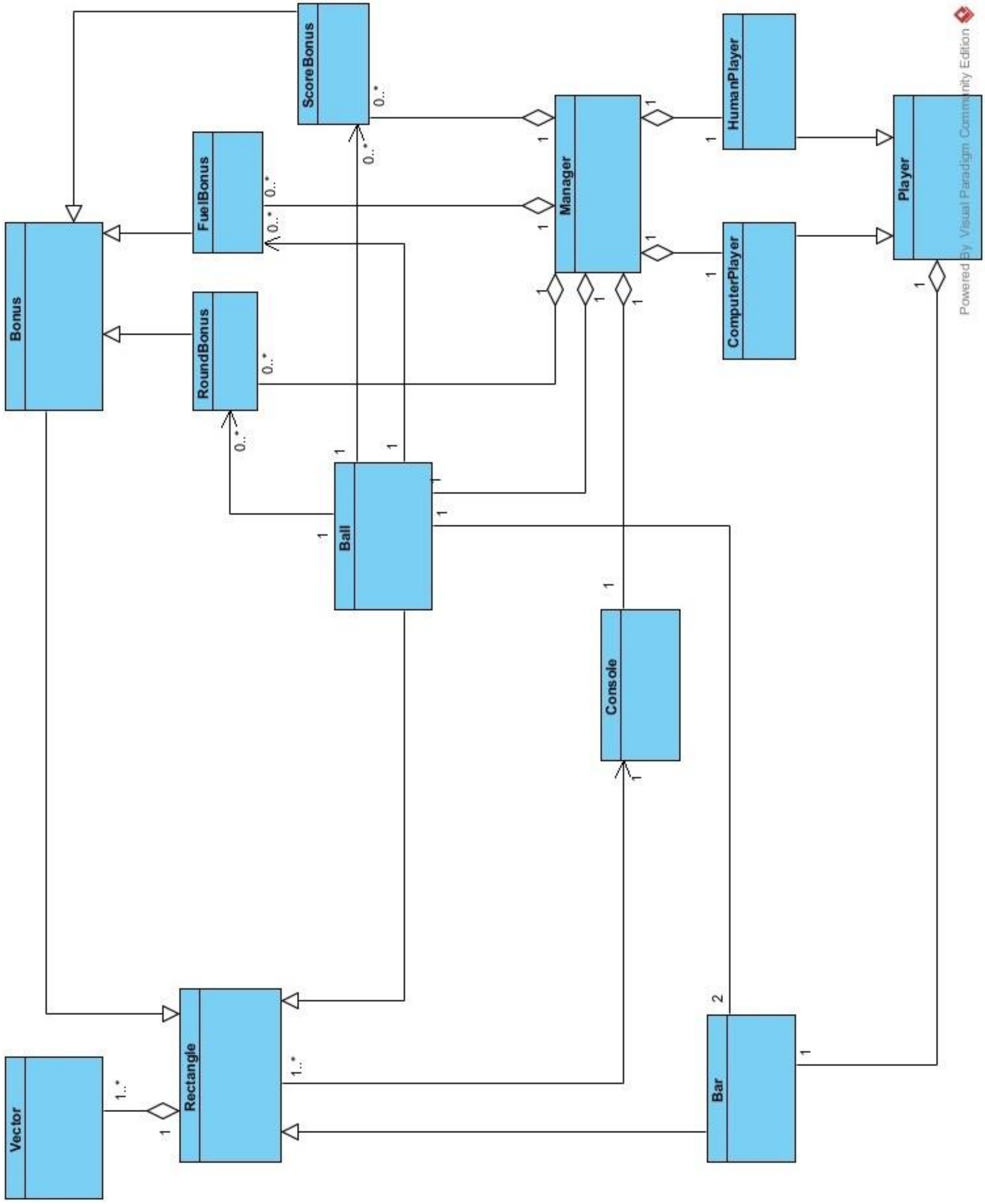
It is a subclass of Rectangle class. There are three types of bonus points in the game that each affects game in different ways.

7) ScoreBonus, RoundBonus and FuelBonus Classes

They are subclasses of Rectangle class. FuelBonus brings Player fuel, ScoreBonus brings Player score point and RoundBonus gives a chance to current Player shoot in the next round.

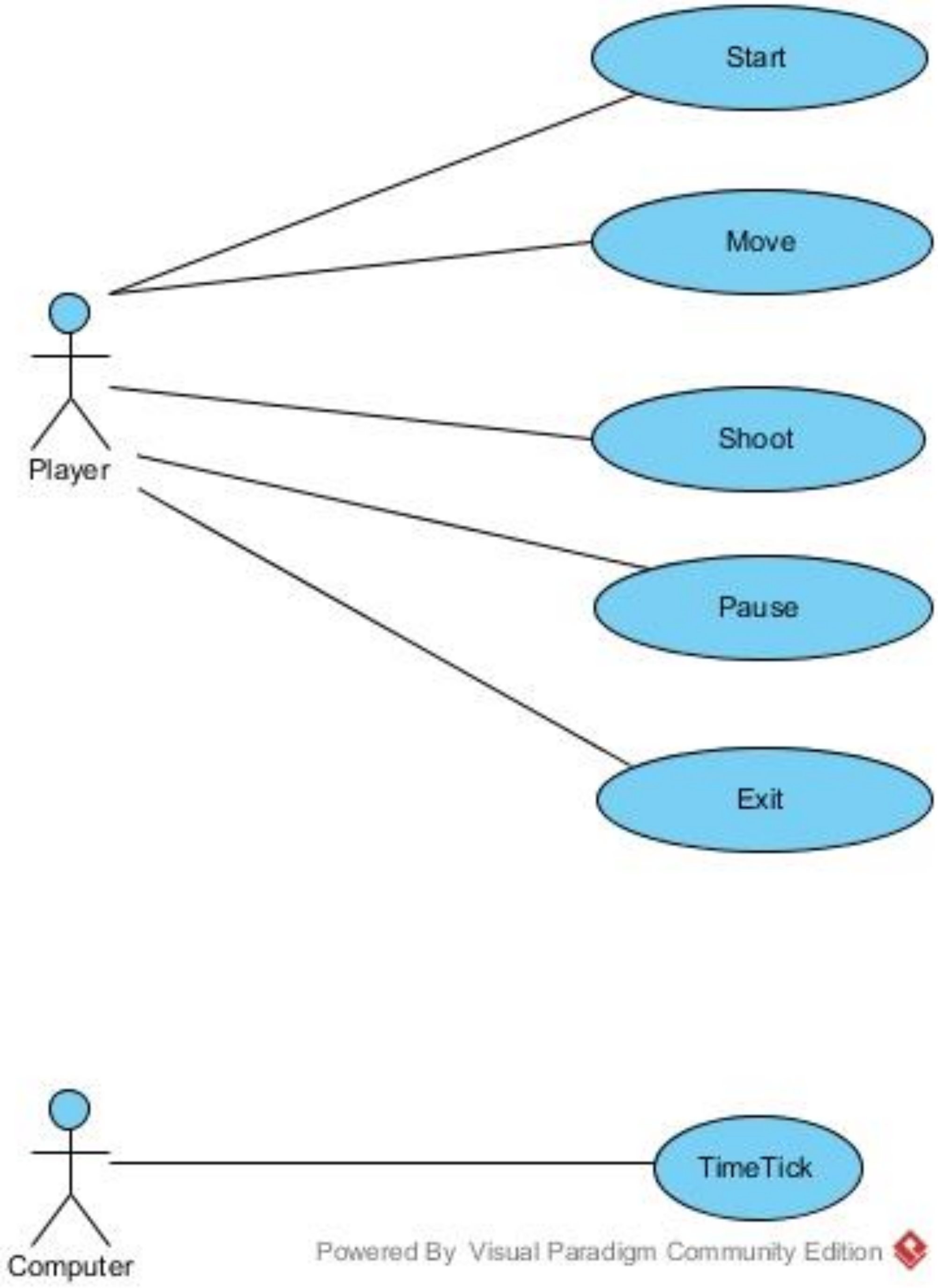
8) Vector and Console Classes

These classes can be called as utility classes. Vector class is used to store position, size and velocity knowledge by other classes. Console class is used to print the object onto console.


Bonus
ScoreBonus
FuelBonus
RoundBonus
Manager
HumanPlayer
ComputerPlayer
Player
Ball
Console
Rectangle
Vector
Bar
0..*
0..*
0..*
0..*
0..*
0..*
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1..*
1..*
2
Powered By Visual Paradigm Community Edition

2. Use Case Diagram

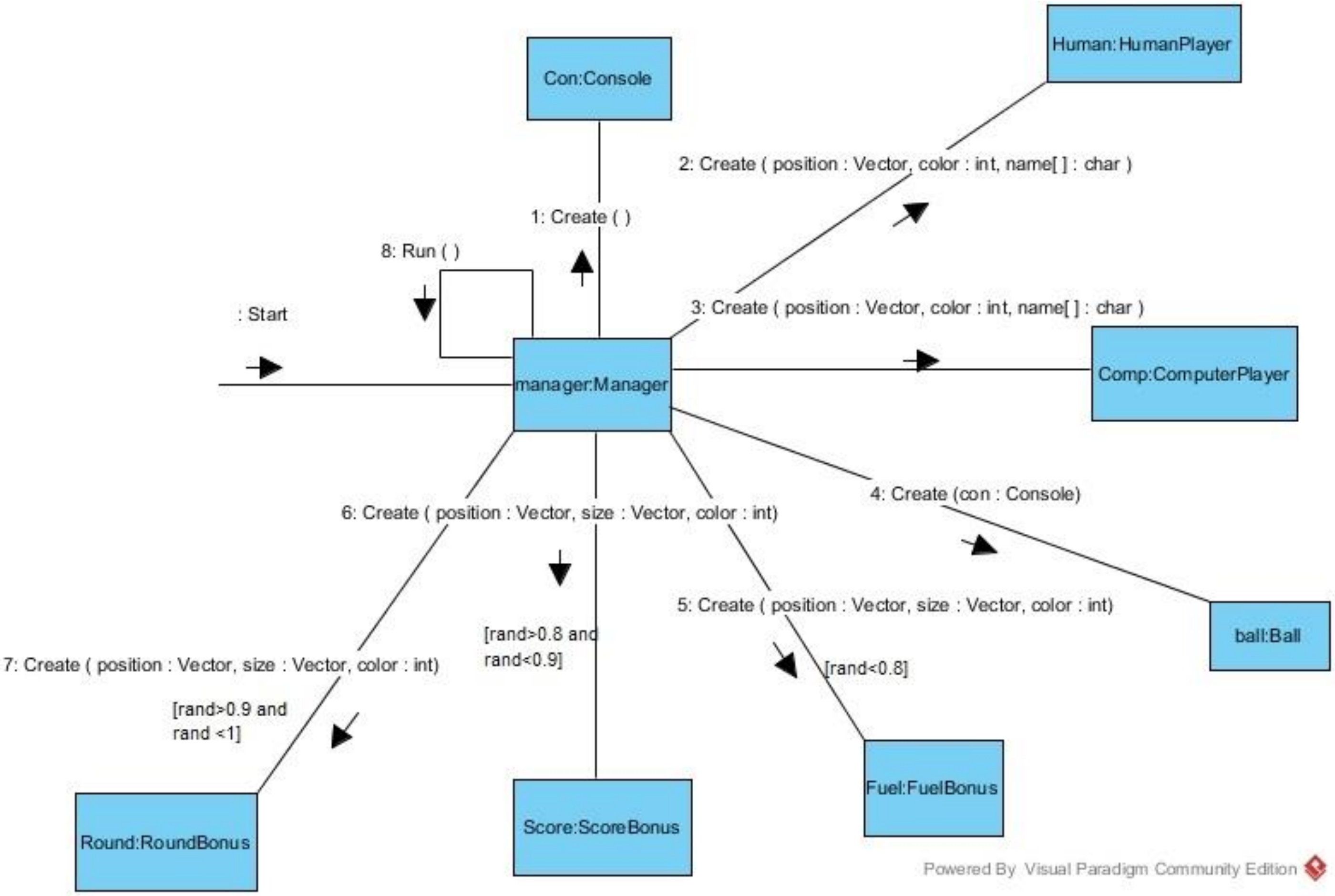
We have two external agents Player and Computer. Start use case is used to create all objects and to start the game. Player moves the bar with Move use case. Shoot use case is used to shoot the ball. Player can pause the game for a while or exit the game with Pause-Exit use cases. Computer will make decisions with TimeTick use case.



3. Collaboration and Sequence Diagrams

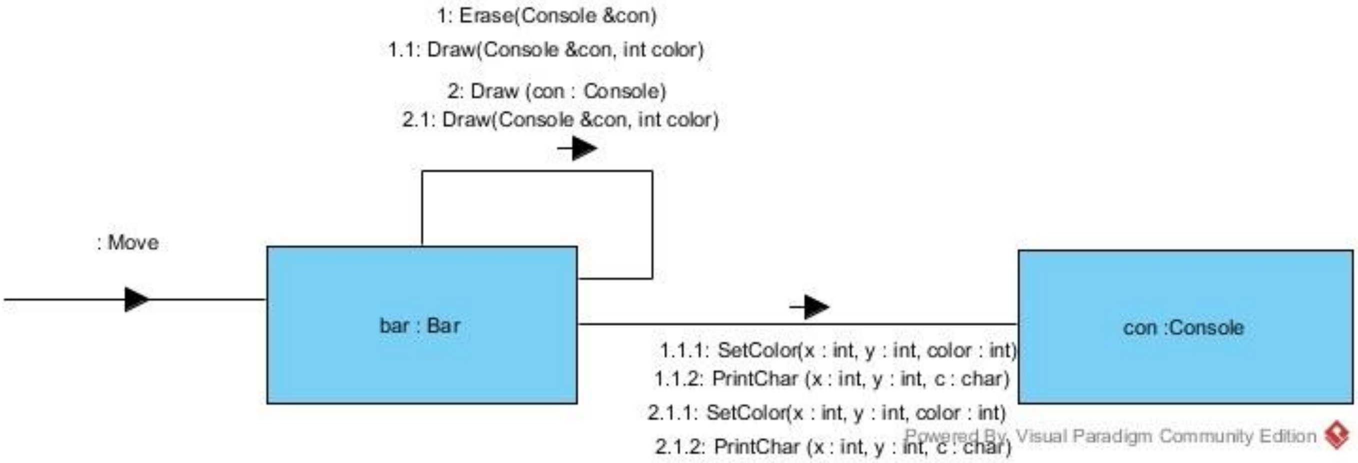
3.1 Start Collaboration Diagram

All objects of the game will be initialized with proper constructors. After object initializations, manager will start the game by calling its function Run.


Con:Console
Human:HumanPlayer
2: Create (position : Vector, color : int, name[ ] : char)
1: Create ( )
8: Run ( )
: Start
3: Create (position : Vector, color : int, name[ ] : char)
manager:Manager
Comp:ComputerPlayer
4: Create (con : Console)
6: Create (position : Vector, size : Vector, color : int)
5: Create (position : Vector, size : Vector, color : int)
ball:Ball
[rand>0.8 and
rand<0.9]
7: Create (position : Vector, size : Vector, color : int)
[rand<0.8]
[rand>0.9 and
rand <1]
Fuel:FuelBonus
Round:RoundBonus
Score:ScoreBonus
Powered By Visual Paradigm Community Edition

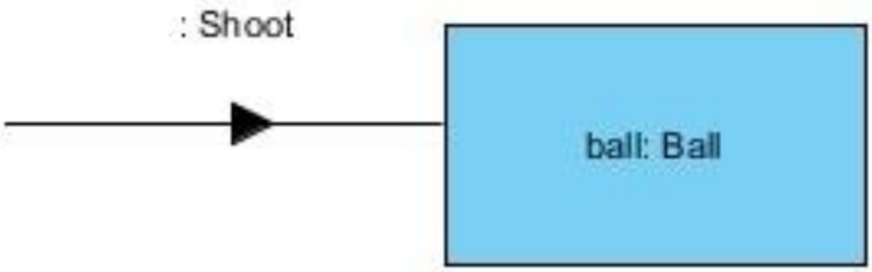
3.2 Move Collaboration Diagram

Player moves the bar with move use case.


1: Erase(Console &con)
1.1: Draw(Console &con, int color)
2: Draw (con : Console)
2.1: Draw(Console &con, int color)
: Move
bar : Bar
con :Console
1.1.1: SetColor(x : int, y : int, color : int)
1.1.2: PrintChar (x : int, y : int, c : char)
2.1.1: SetColor(x : int, y : int, color : int)
2.1.2: PrintChar (x : int, y : int, c : char)
Powered By Visual Paradigm Community Edition

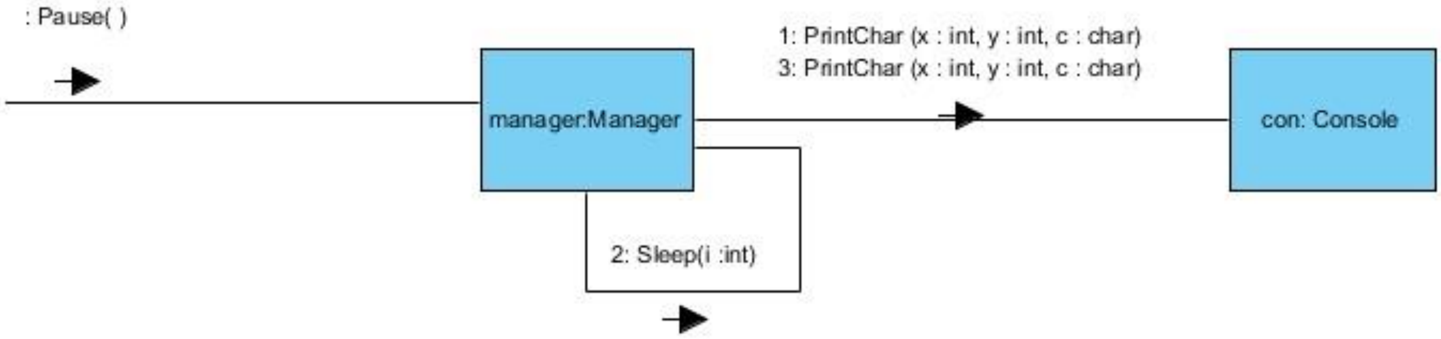
3.3 Shoot Use Case Collaboration Diagram

Player can shoot the ball only if IsInTouchWithBar() function turns true meaning that the ball is caught by bar



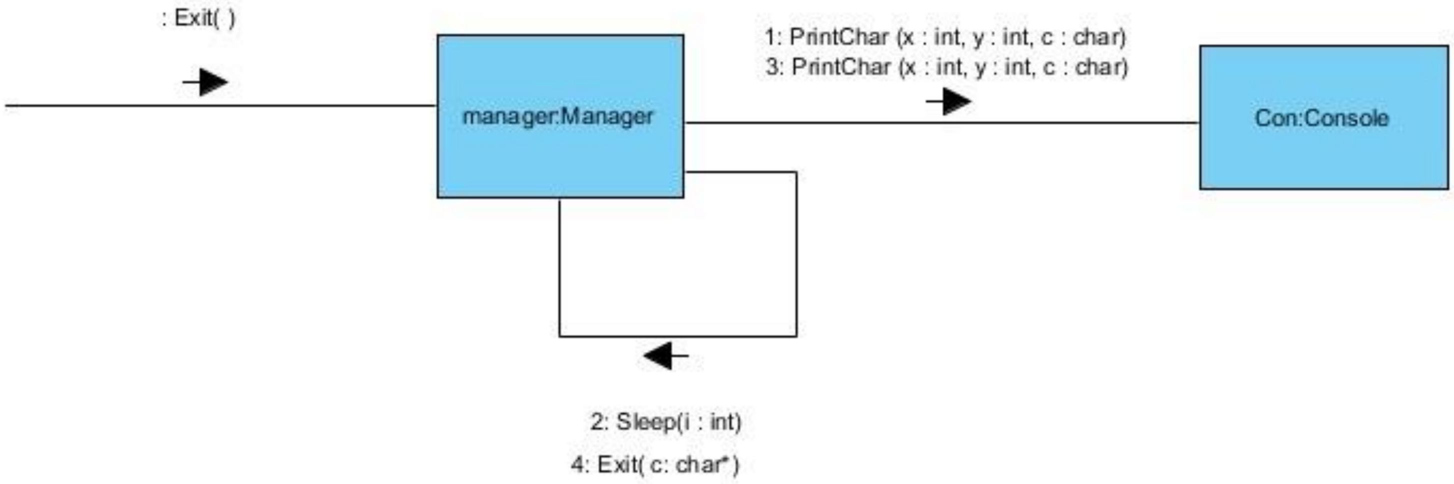
3.5 Pause Collaboration Diagrams

Player can pause the game. First calling of PrintChar function prints sign of pause (“P”) onto console. Then Sleep function is called and game is paused for a limited time. After that time, game continues. Second calling of PrintChar function is for deleting the sign of pause.


: Pause( )
1: PrintChar (x : int, y : int, c : char)
3: PrintChar (x : int, y : int, c : char)
manager:Manager
con: Console
2: Sleep(i :int)

3.6 Exit Collaboration Diagrams

Player can exit the game. . First calling of PrintChar function prints sign of exit (“E”) onto console. Then Sleep function is called and game is paused for a limited time. Second calling of PrintChar function is for deleting the sign of exit. Then, Exit function is called; thus, game is over.


: Exit( )
1: PrintChar (x : int, y : int, c : char)
3: PrintChar (x : int, y : int, c : char)
manager:Manager
Con:Console
2: Sleep(i : int)
4: Exit(c: char*)

3.7 TimeTick Sequence Diagram

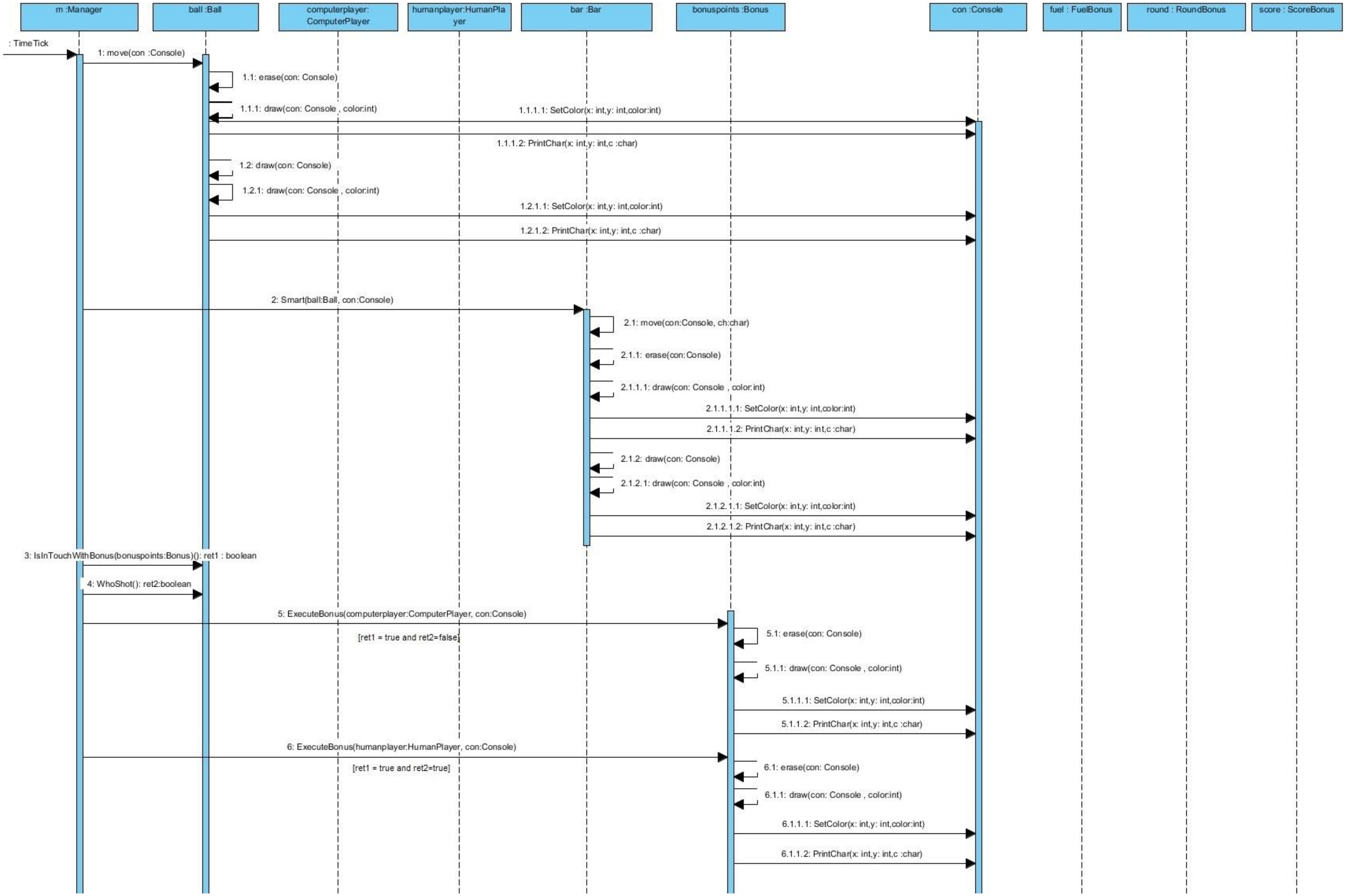
Firstly, move function of ball object is called. Thus, ball moves randomly to a player's bar. Basically, draw functions of objects draw them to the console. Their erase functions erase them from the console. Secondly, Smart function of bar object is called. Computer can move its bar to catch the ball with that function.

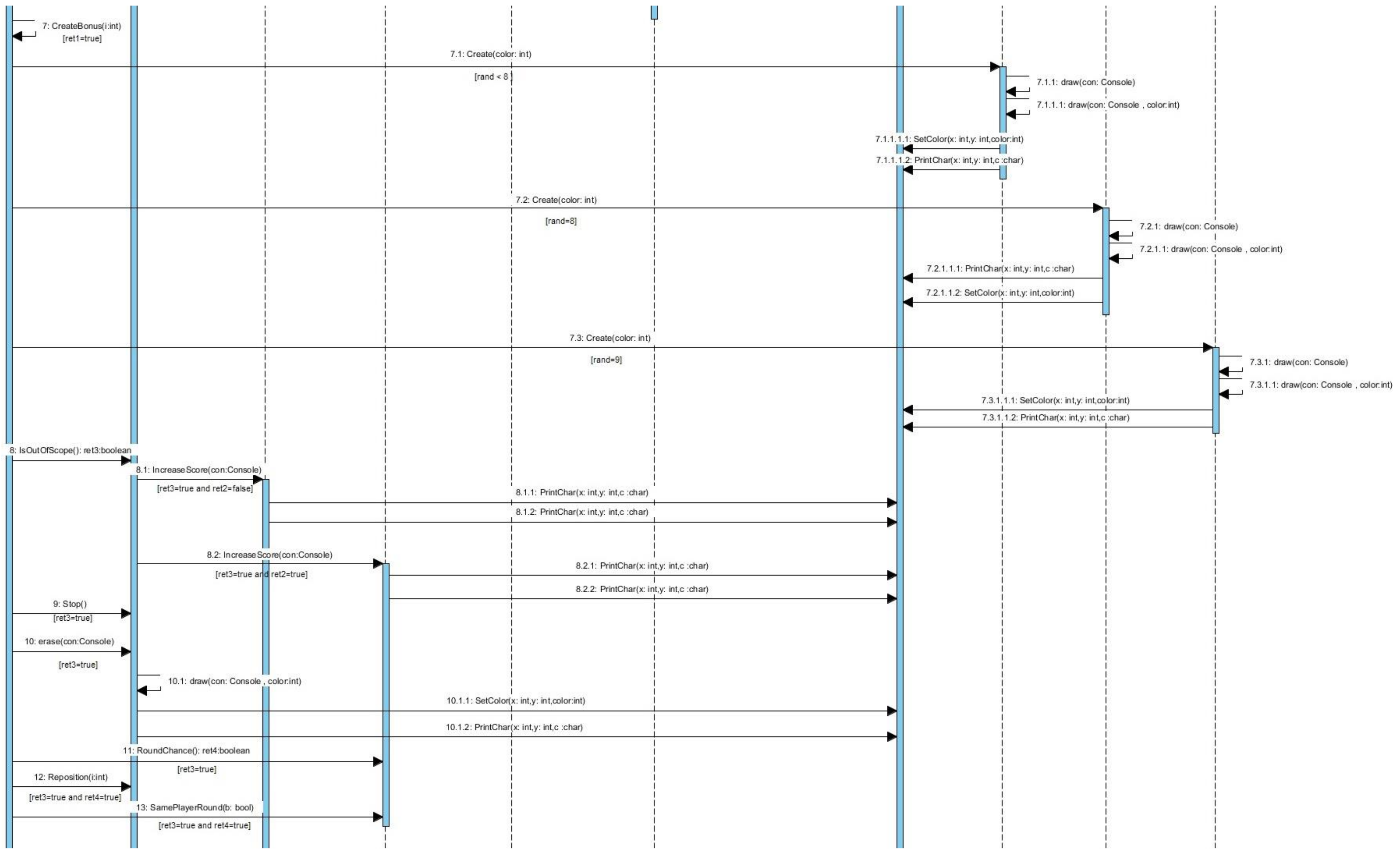
When ball continues to move in game scope, it can hit a bonus. To check if a collision occurs, IsInTouchWithBonus function is called. After that WhoShot function is called. If the ball hit a bonus, ExecuteBonus function of that bonus object is called. ExecuteBonus is a virtual function which acts differently for each bonus type. The taken bonus object is deleted and a new bonus object is created randomly after that.

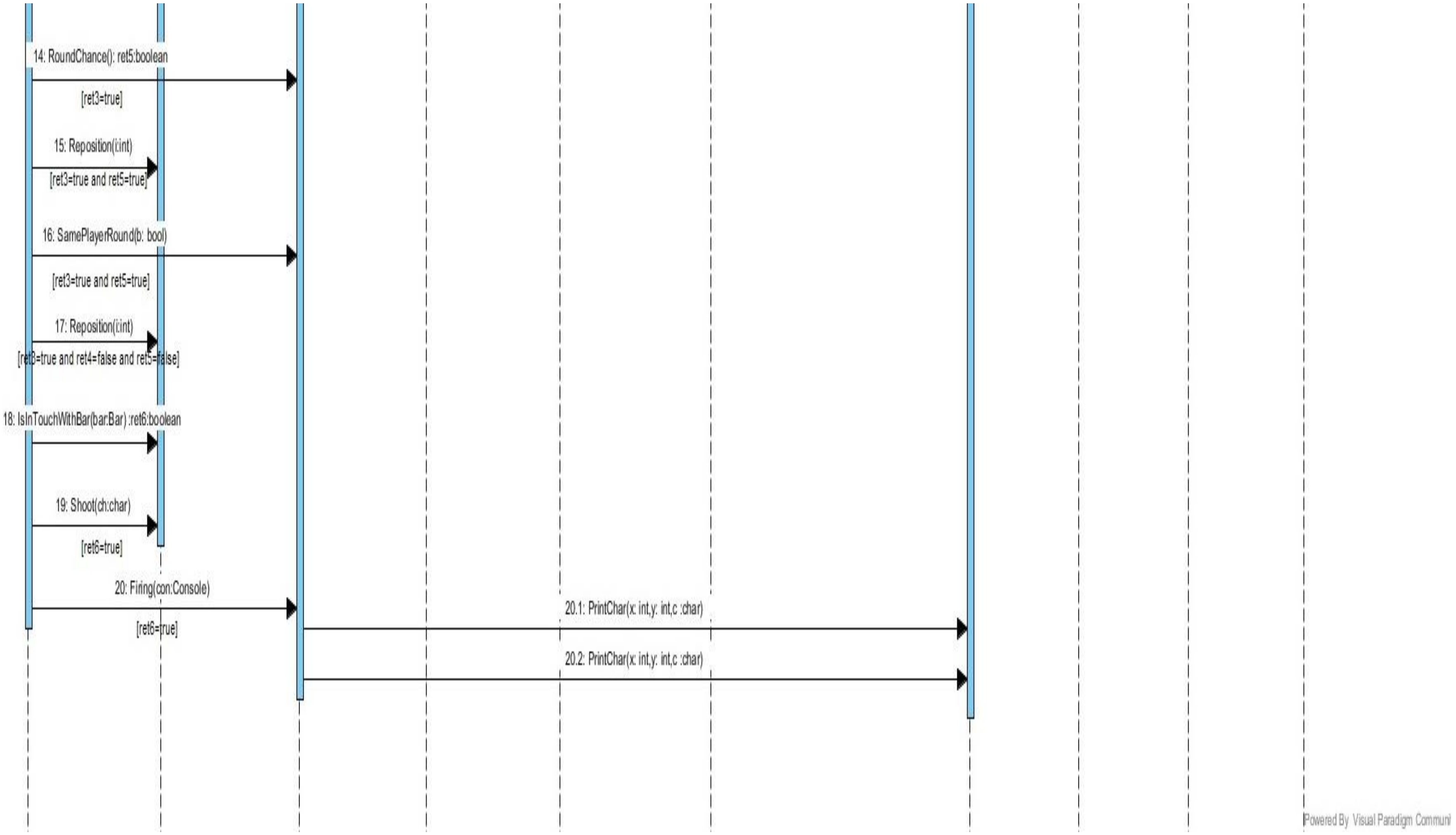
IsOutOfScope function controls whether a goal occurs or not. When a goal occurs, the score of the player is increased according to the return value of WhoShot function. Stop function sets the velocity of the ball to zero. If a player takes a round bonus, in the next round the initial direction of the ball will be set towards that player. Otherwise, its direction will be random.

When the ball is in touch with a player's bar, that player can shoot the ball to score a goal.

Computer will shoot the ball in one of three directions randomly. When it shoots the ball its fuel is decreased by one.

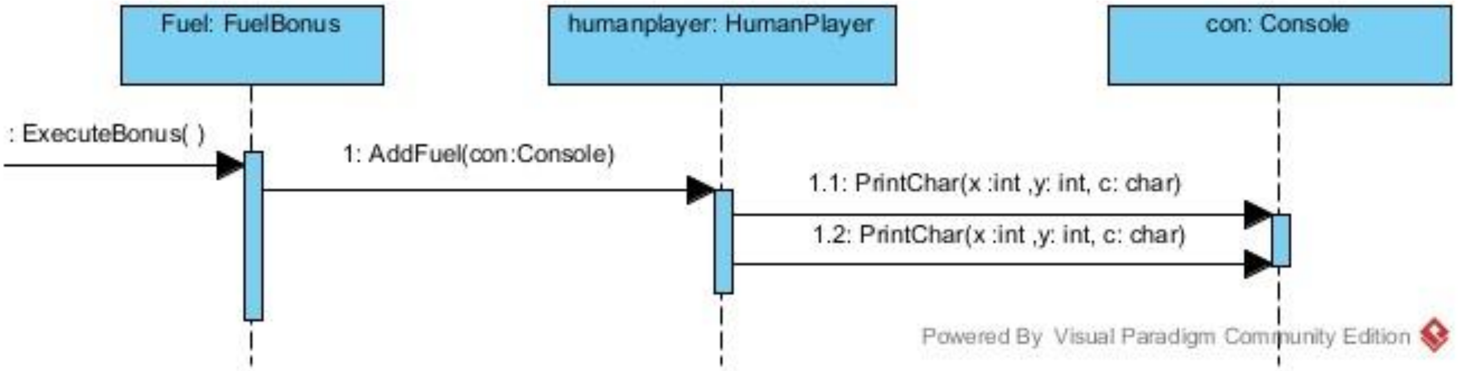

m :Manager
ball :Ball
computerplayer: ComputerPlayer
humanplayer:HumanPla yer
bar :Bar
bonuspoints :Bonus
con :Console
fuel : FuelBonus
round : RoundBonus
score : ScoreBonus
: TimeTick
1: move(con :Console)
1.1: erase(con: Console)
1.1.1: draw(con: Console , color:int)
1.1.1.1: SetColor(x: int,y: int,color:int)
1.1.1.2: PrintChar(x: int,y: int,c :char)
1.2: draw(con: Console)
1.2.1: draw(con: Console , color:int)
1.2.1.1: SetColor(x: int,y: int,color:int)
1.2.1.2: PrintChar(x: int,y: int,c :char)
2: Smart(ball:Ball, con:Console)
2.1: move(con:Console, ch:char)
2.1.1: erase(con:Console)
2.1.1.1: draw(con: Console , color:int)
2.1.1.1.1: SetColor(x: int,y: int,color:int)
2.1.1.1.2: PrintChar(x: int,y: int,c :char)
2.1.2: draw(con: Console)
2.1.2.1: draw(con: Console , color:int)
2.1.2.1.1: SetColor(x: int,y: int,color:int)
2.1.2.1.2: PrintChar(x: int,y: int,c :char)
3: IsInTouchWithBonus(bonuspoints:Bonus)(): ret1 : boolean
4: WhoShot(): ret2:boolean
5: ExecuteBonus(computerplayer:ComputerPlayer, con:Console)
[ret1 = true and ret2=false]
5.1: erase(con: Console)
5.1.1: draw(con: Console , color:int)
5.1.1.1: SetColor(x: int,y: int,color:int)
5.1.1.2: PrintChar(x: int,y: int,c :char)
6: ExecuteBonus(humanplayer:HumanPlayer, con:Console)
[ret1 = true and ret2=true]
6.1: erase(con: Console)
6.1.1: draw(con: Console , color:int)
6.1.1.1: SetColor(x: int,y: int,color:int)
6.1.1.2: PrintChar(x: int,y: int,c :char)

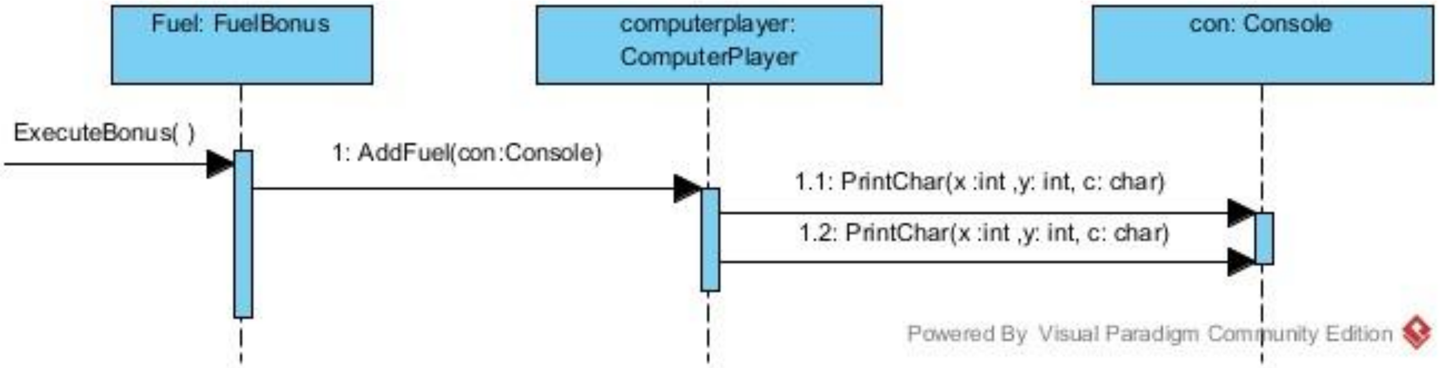

7: CreateBonus(i:int)
[ret1=true]
7.1: Create(color: int)
[rand < 8]
7.1.1: draw(con: Console)
7.1.1.1: draw(con: Console , color:int)
7.1.1.1.1: SetColor(x: int,y: int,color:int)
7.1.1.1.2: PrintChar(x: int,y: int,c :char)
7.2: Create(color: int)
[rand=8]
7.2.1: draw(con: Console)
7.2.1.1: draw(con: Console , color:int)
7.2.1.1.1: PrintChar(x: int,y: int,c :char)
7.2.1.1.2: SetColor(x: int,y: int,color:int)
7.3: Create(color: int)
[rand=9]
7.3.1: draw(con: Console)
7.3.1.1: draw(con: Console , color:int)
7.3.1.1.1: SetColor(x: int,y: int,color:int)
7.3.1.1.2: PrintChar(x: int,y: int,c :char)
8: IsOutOfScope(): ret3:boolean
8.1: IncreaseScore(con:Console)
[ret3=true and ret2=false]
8.1.1: PrintChar(x: int,y: int,c :char)
8.1.2: PrintChar(x: int,y: int,c :char)
8.2: IncreaseScore(con:Console)
[ret3=true and ret2=true]
8.2.1: PrintChar(x: int,y: int,c :char)
8.2.2: PrintChar(x: int,y: int,c :char)
9: Stop()
[ret3=true]
10: erase(con:Console)
[ret3=true]
10.1: draw(con: Console , color:int)
10.1.1: SetColor(x: int,y: int,color:int)
10.1.2: PrintChar(x: int,y: int,c :char)
11: RoundChance(): ret4:boolean
[ret3=true]
12: Reposition(i:int)
[ret3=true and ret4=true]
13: SamePlayerRound(b: bool)
[ret3=true and ret4=true]


14: RoundChance(): ret5:boolean
[ret3=true]
15: Reposition(i:int)
[ret3=true and ret5=true]
16: SamePlayerRound(b: bool)
[ret3=true and ret5=true]
17: Reposition(i:int)
[ret3=true and ret4=false and ret5=false]
18: IsInTouchWithBar(bar:Bar) :ret6:boolean
19: Shoot(ch:char)
[ret6=true]
20: Firing(con:Console)
[ret6=true]
20.1: PrintChar(x: int,y: int,c :char)
20.2: PrintChar(x: int,y: int,c :char)
Powered By Visual Paradigm Communi

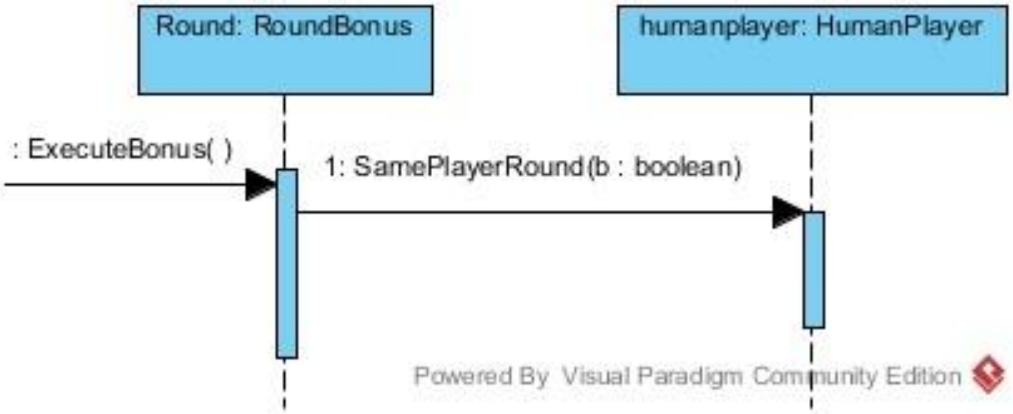
3.8 Bonus Polymorphism Scenarios Sequence Diagrams

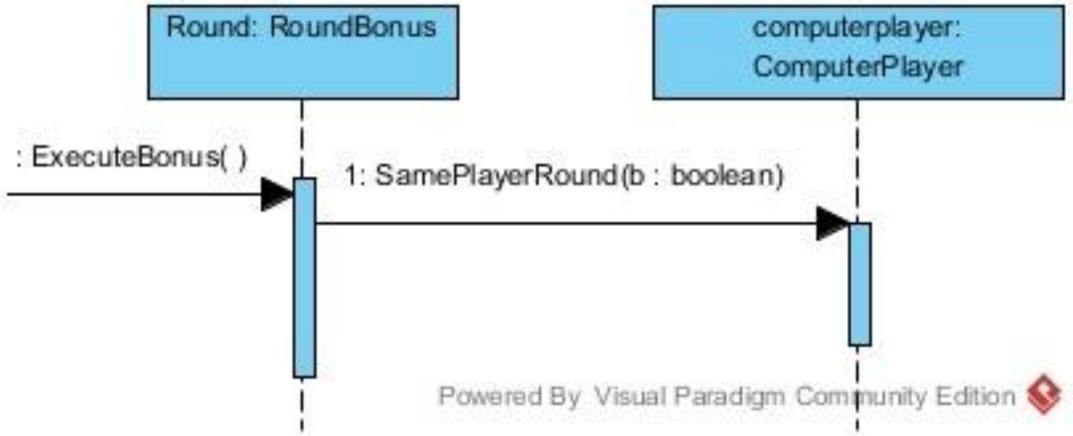
There are three different bonuses and these are executed in different ways when ball hits a bonus. FuelBonus is executed with adding fuel, RoundBonus is executed with giving chance to player in order to begin in next round and ScoreBonus is executed with increasing score.


Fuel: FuelBonus
humanplayer: HumanPlayer
con: Console
: ExecuteBonus( )
1: AddFuel(con:Console)
1.1: PrintChar(x :int ,y: int, c: char)
1.2: PrintChar(x :int ,y: int, c: char)
Powered By Visual Paradigm Community Edition

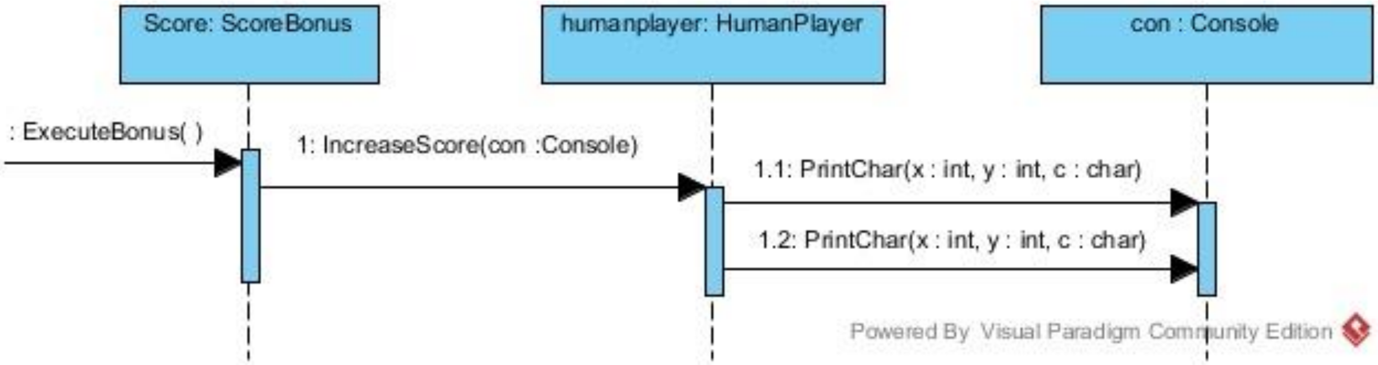

Fuel: FuelBonus
computerplayer: ComputerPlayer
con: Console
ExecuteBonus( )
1: AddFuel(con:Console)
1.1: PrintChar(x :int ,y: int, c: char)
1.2: PrintChar(x :int ,y: int, c: char)
Powered By Visual Paradigm Community Edition

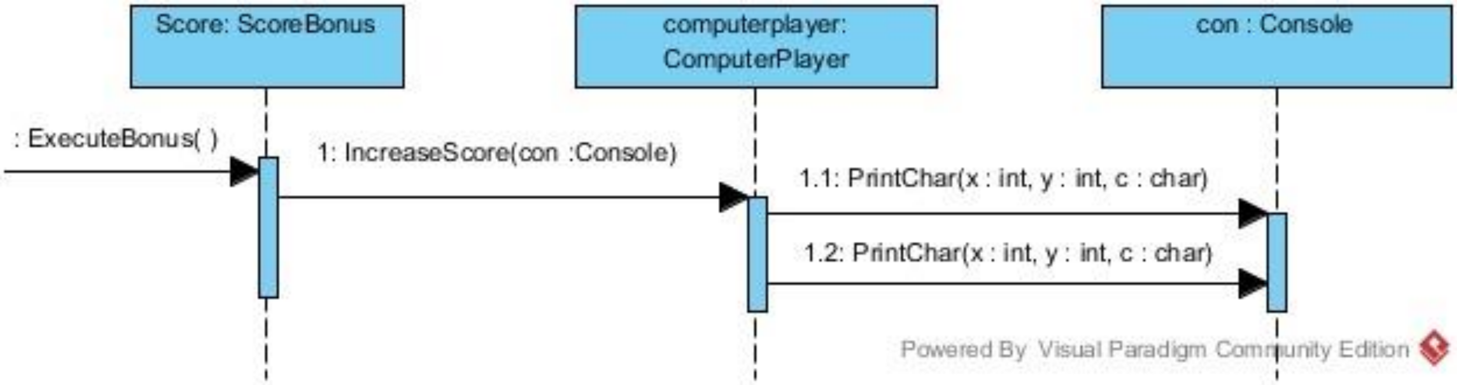
AddFuel function increases the fuel of the player by one.


Round: RoundBonus
humanplayer: HumanPlayer
: ExecuteBonus( )
1: SamePlayerRound(b : boolean)
Powered By Visual Paradigm Community Edition


Round: RoundBonus
computerplayer: ComputerPlayer
: ExecuteBonus( )
1: SamePlayerRound(b : boolean)
Powered By Visual Paradigm Community Edition

SamePlayerRound makes the round attribute of the player true.


Score: ScoreBonus
humanplayer: HumanPlayer
con : Console
: ExecuteBonus( )
1: IncreaseScore(con :Console)
1.1: PrintChar(x : int, y : int, c : char)
1.2: PrintChar(x : int, y : int, c : char)
Powered By Visual Paradigm Community Edition


Score: ScoreBonus
computerplayer: ComputerPlayer
con : Console
: ExecuteBonus( )
1: IncreaseScore(con :Console)
1.1: PrintChar(x : int, y : int, c : char)
1.2: PrintChar(x : int, y : int, c : char)
Powered By Visual Paradigm Community Edition

IncreaseScore increases the score attribute of the player.