|  |
| --- |
| RipCityEvent |
| \*Double total\_time;  \*Double reg\_time;  -Vector<string> names;  \*Vector<Player\*>Players; |
| +RipCityEvent()  +Void Registration()  +Virtual void SetTimes()  +Virtual void TheStats() |

UML Diagram:

|  |
| --- |
| Player |
| -Int layInMade  -Int LayinPercent  -Int FreeThrowMade  -Int FreeThrowPercent  -Int ThreePtMade  -Int ThreePtPercent  -Int HalfMade  -Int HalfPercent  -String name  -Int totalmakes |
| Player(string name)  Int getLayinPerc()  Int getfreeThrowPerc()  Int getThreePtPerc()  Int getHalfPerc()  Void setlayinMade(int made)  Void setfreeThrowMade(int made)  Void setThreePtMade(int made)  Void setHalfMade(int made)  String getName()  Int getlayinmade()  Int getFreeThrowMade()  Int getThreePointMade()  Int getHalfMade()  Int getTotalMakes() |

|  |
| --- |
| ShootingEvent |
| -Double time\_between\_ppl  -Double time\_shooting  -Double time\_between\_shots  -Double FGpercentage  -Double finalscore  -Int baskets  -Int totalshots  -Map<string, int> MadeShot  -Priority\_queue<int> Podium |
| +ShootingEvent()  +Void setTimes()  +Void Shooting()  +Void theStats() |

Psuedo-Code:

RipCityEvent::void Registration():

This is where all the names will be added to vector of names, how long the event will be will be answered, and the vector of names will then become a vector of Player

Open files (first and last name)

For(till file is done)

{ Name = first + last

Names.push(name)

Next line

}

“How many hours will event be?”

Cin>>total\_time

For(size of names vector)

Players.push(new Player(names[i]);

RipCityEvent::Virtual void SetTimes and TheStats:

These are both virtual because in the future there could be additional events that need times to be set and the final simulation analysis stats to be shown.

ShootingEvent::void setTimes()

“How long will time between shooters, and time shooting be? How many baskets?”

Cin >> all three numbers;

ShootingEvent::void Shooting()

For(size of Players)

If(player has decided to shoot this type of shoot)

Time\_between\_shots= a certain const number;

FGpercentage is random within a certain range;

Totalshots = totalshots + these shots

Players[i]->setThisShot(these shots);

Finalscore = finalscore + these shots \* (1,2,3 or 4 depending on the shot)

Else

Players[i]->setthisShot(0);

Add names and scores to two different priority queue so results can be shown alphabetically and by score

Have a free raffle drawing

ShootingEvent::void theStats()

Display total shots, total makes, of each shot, overall FG%, average points scored, and anything else that is helpful.

NOTE: Player functions not in pseudo code because they are just getters and setters.

USE-CASES:

1.) Choosing the time and number of baskets

User: chooses how long event will be System: assigns the event to be that long

User: chooses how long between people System: assign time\_between\_people to that number

User: choose how long players will shoot for System: assigns time\_shooting to that number

User: chooses how many baskets will be at the event System: assigns that many baskets

Specification Requirements:

In this simulation, there will be a shooting competition with multiple courts. These courts will have one person on at a time, there will also be registration. Each person will register then will shoot, there will be lay-ins, free-throws, three-pointers, and half-court shots available all worth increasing amount of points but drastically different chances of making them. Everyone who has been able to shoot by the time the event is over will be put in a priority queue and displayed from first to last. The idea with this project is that it will also be able to add more competition events in the future, like maybe a trivia game about the history of Portland and the Trail Blazers. The user will be able to pick how long the event will be, how long the time to shoot will be, how much time passes between people shooting, how long it takes to register, and how many half courts are able to be used. At the end, the program will display how many people participated in this event. There will need to be three multiple classes for this to work and stay clean.