Software Engineering

Books or notes are **not** allowed.
Write only on these sheets. **Concise** and **readable** answers please.

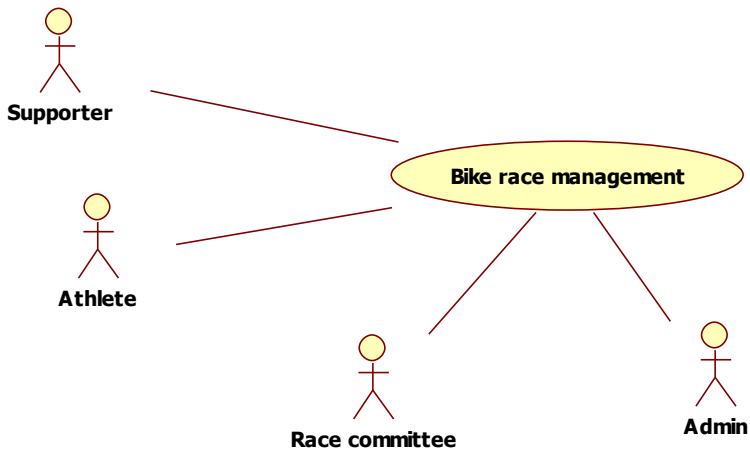
Surname, name, matricola _____

Bike race management
A bike race (ex Giro d'Italia, Tour de France) is made of many stages. A stage typically happens in one day, starts from a place and arrives to another place. Stages are numbered (first, second etc). A stage has a length and a maximum duration time.
Many athletes compete in the race. Each athlete has a name, surname, birthdate, and a race number assigned by the race committee.
Each athlete runs each stage and obtains a certain time on it (or possibly retires from the stage and the race). Assume that times (start time – end time) are entered manually for each athlete and each stage. Based on these times a ranking of athletes is produced for each stage. There is also a race ranking, given from the sum of times of an athlete in each stage.
An athlete can receive a penalization, either a time to be added, or in the worst case a disqualification.

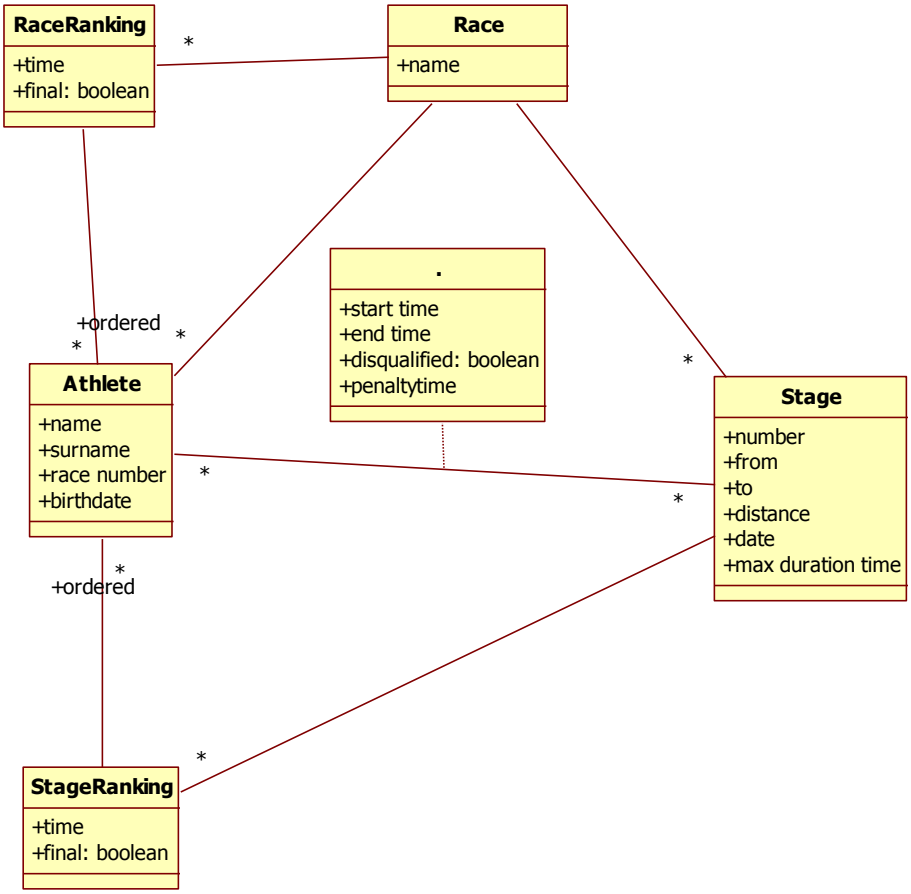
In the following you should analyze and model a client server application to support the management of a bike race.

1 – a. Define the **context diagram** (including relevant interfaces)

Actor	Physical interface	Logical interface
Admin	PC	GUI (defines users and privileges)
Athlete	PC / smartphone	GUI (subscribes, watches results)
Race committee	PC / smartphone	GUI (configures race, sets times, penalties, validates results)
Supporter	PC / smartphone	GUI (watches athletes and results)



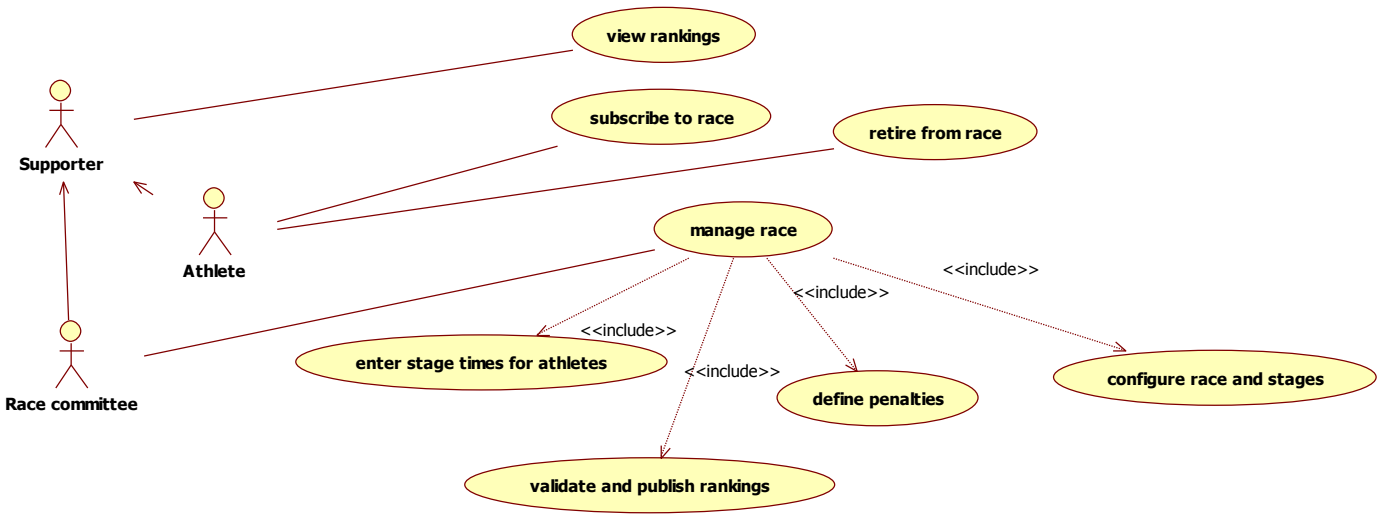
1-b Define the **glossary** (key concepts and their relationships) (UML class diagram) for the application



StageRanking and RaceRanking have a time stamp, the idea is that many rankings are generated, until a final one. Another option is to have just one ranking.
A Ranking is an ordering of Athletes, and this is represented by the relationship.

Times for athletes depend both on the athlete and the stage, so they must be represented as an association class.

1-c Draw the Use Case Diagram for the application. For each Use Case give self-explainable long names, or a short textual description



1-D List the **NON functional requirements** that you deem important for the application

ID	Description
1	Usability: all functions should be usable by average user (user of smartphone since 3+ years) with no training
2	Security: athletes and supporters should not be able to modify any data
3	Performance: all functions should respond in <0,5 sec

1-e Describe below the scenario specific to an employee of the organization who enters start time for a stage for an athlete.

Precondition: Race R is defined, stage S in race R is defined, Athlete A exists and has subscribed to race R

Postcondition: start time for athlete A on stage S of race R is defined

Step	Description
1	Select race R, stage S, athlete A
2	Set start time for athlete A, in stage S in race R
3	
4	
5	

2 (7 points) -Define black box tests for the following function, using equivalence classes and boundary conditions.

In a bike race, the bikers must complete the entire track within a maximum time, otherwise their race is not valid. The maximum time is computed, for each race, based on the winner's time, on the average speed on the track, and on the category of the track.

For tracks of category 'A' (easy tracks) the maximum time is computed as the winner's time increased by 5% if the average speed is lower than 30 km/h (30 included), 10% if the average speed is between 30 and 35 km/h (35 included), and 15% if the average speed is higher than 35 km/h.

For tracks of category 'B' (normal tracks) the maximum time is computed as the winner's time increased by 20% if the average speed is lower than 30 km/h (30 included), 25% if the average speed is between 30 and 35 km/h (35 included), and 30% if the average speed is higher than 35 km/h.

For tracks of category 'C' (hard tracks) the maximum time does not depend on average speed, and is always computed as the winner's time increased by 50%.

The function computeMaxTime receives as parameters winner_time (the time of the winner, in minutes), avg_speed (the average speed of the track, in km/h) and track_type (a char, whose valid values are 'A', 'B', or 'C'). It gives as output the maximum time, 0 if there are errors in the input.

double computeMaxTime(double winner_time, double avg_speed, char track_type)

Examples:

computeMaxTime(50, 27, 'A') -> 50 + 50*0.05 = 52.5
computeMaxTime(60, 33, 'B') -> 60 + 60*0.25 = 75
computeMaxTime(80, 40, 'C') -> 80 + 80*0.5 = 120

Winner_time	Avg_speed	Track_type	Valid / Invalid	Test Case
[mindouble, 0.0]	*	*	I	T1(-5, 33, ‘B’) -> error Tb(**0**, 33, ‘B’) -> error
*	[mindouble, 0.0]	*	I	T2(40, -5, ‘B’) -> error Tb(40, **0**, ‘B’) -> error
*	*	∉ { ‘A’, ‘B’, ‘C’ }	I	T3(40, 33, ‘F’) -> error
(0.0, maxdouble)	(0.0, 30.0]	‘A’	V	T4(40, 5, ‘A’) -> 42 Tb(**0.001**, 5, ‘A’) -> 0.00105 Tb(40, **0.001**, ‘A’) -> 42 Tb(40, **30.0**, ‘A’) -> 42
“	“	‘B’	V	T5(40, 5, ‘B’) -> 48
“	“	‘C’	V	T6(40, 5, ‘C’) -> 60
“	(30.0, 35.0]	‘A’	V	T7(40, 32.5, ‘A’) -> 44 Tb(40, **30.01**, ‘A’) -> 44 Tb(40, **35.0**, ‘A’) -> 44
“	“	‘B’	V	T8(40, 32.5, ‘B’) -> 50
“	“	‘C’	V	T9(40, 32.5, ‘C’) ->60
“	(35.0, maxdouble]	‘A’	V	T10(40, 40, ‘A’) -> 46 Tb(40, **35.0**, ‘A’) -> 46
“	“	‘B’	V	T11(40, 40, ‘B’) ->52
“	“	‘C’	V	T12(40, 40, ‘C’) ->60

3 (7 points) – For the following function define the control flow graph, and define test cases to obtain the highest possible node coverage, edge coverage, multiple condition coverage, loop coverage, path coverage. For the test cases, **write only the input value**.

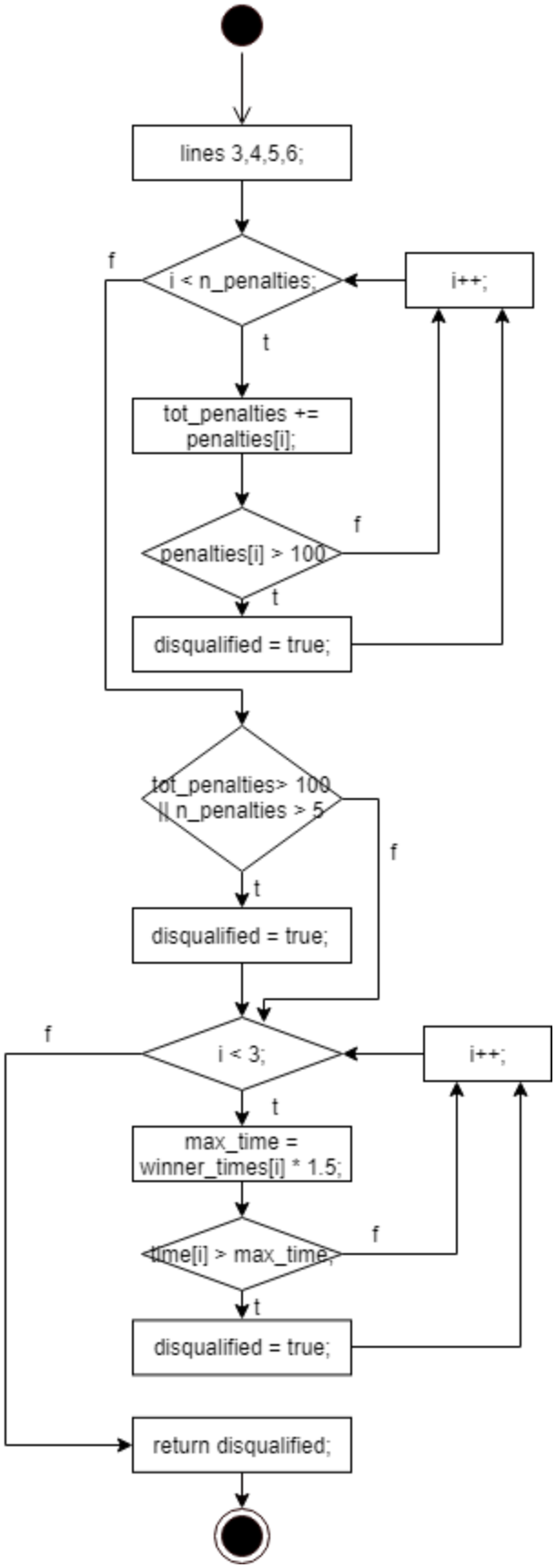
Write control flow graph here

```
1  boolean racer_disqualified(int times[3],
      int winner_times[3], int n_penalties, int* penalties) {
2
3     bool disqualified = false;
4     int i;
5     int max_time;
6     int tot_penalties;
7
8     for (i = 0; i < n_penalties; i++) {
9           tot_penalties += penalties[i];
10          if (penalties[i] > 100)
11                disqualified = true;
12    }
13    if (tot_penalties > 100 || n_penalties > 5)
15          disqualified = true;
16
17    for (i=0; i<3; i++) {
18
19          max_time = winner_times[i] * 1.5;
20
21          if (time[i] > max_time[i])
22                disqualified = true;
23    }
24
25  }
```

Coverage type	Number of test cases needed to obtain 100% coverage	Coverage obtained with test cases defined (%)	Test cases defined
Node	1	100	T1
Edge	2	100	T1, T2
Multiple condition line 13	4	100	T1 (TF), T2 (FF), T3 (FT), T4 (TT)
Loop line 8	3	100	T6 (0 loops), T5 (1 loop), T3 (n loops)
Path	2*2^3*2^(n_penalties-1)	100% coverage obtainable with automated test case generation mechanisms	-

Write test case ID (T1, T2 ..) in the rightmost column, and test cases here

T1({10, 10, 10}, {20, 20, 20}, 2, {150, 50});
T2({10, 10, 10}, {20, 20, 20}, 2, {5, 5});
T3({10, 10, 10}, {20, 20, 20}, 6, {5, 5, 5, 5, 5, 5});
T4({10, 10, 10}, {20, 20, 20}, 5, {50, 50, 50, 50, 50, 50});
T5({10, 10, 10}, {20, 20, 20}, 1, {50});
T6({10, 10, 10}, {20, 20, 20}, 0, {});


lines 3,4,5,6;
i < n_penalties;
f
t
i++;
tot_penalties += penalties[i];
penalties[i] > 100
f
t
disqualified = true;
tot_penalties> 100 || n_penalties > 5
f
t
disqualified = true;
i < 3;
f
t
i++;
max_time = winner_times[i] * 1.5;
time[i] > max_time;
f
t
disqualified = true;
return disqualified;

4 (1 points) – There are many software processes. What factors characterize a software process?

Number of iterations, sequential vs parallel activities, new development vs maintenance, emphasis on documents
(the question is about *process* characteristics, not *project* characteristics)

5 (1 point) – What techniques can be used to validate the functional requirements of an application to be built?

Inspection of requirements, prototype building, GUI prototype building, writing acceptance test cases

6 (1 point) – In the context of configuration management, what is the purpose of 'check in' 'check out' operations?

Enforce sequential changes to CI (in lock modify unlock mode) or support parallel changes (in copy modify merge mode) without inconsistencies.

(question is not about what is check in checkout – it is about why they are introduced)

7 (1 point) – Describe briefly the pair programming technique in Xtreme Programming

Two people working on one machine to develop both production code and test cases. One person writes, the other controls and suggests improvements / modifications. The pair reverses roles often.

8 (1 point) – An application is developed for an organization in 6 months by 3 people. Then it is used by the organization during 12 years. Argument whether maintenance costs could be higher /lower than development costs

Maintenance costs will probably be much higher than development cost, since the duration of operations (12 years) is way longer than development (6 months)
Effort for development: 3*6 = 18 person months
Effort for maintenance: assuming 6 months per year (half a person maintaining the application) = 6*12 = 72 person months

More precisely, if maintenance requires more than 18person months / 12 = 1,5 person months per year = 33 person days per year, then maintenance costs will be higher than development costs.