Name: Fida Rahman
Date: 28th April 2021

Project Title: Cricket Player Directory

Summary of project: Cricket is very popular sport in South Asian countries (similar to baseball). This will be a directory for cricket players. It is an app for ground managers to manage players playing in his ground (common/fixed stadium). It will help the user (manager) to view and add players. And then finally will get a printed version in the name of Players.txt.

In this directory, every player will be either a batsman, bowler or wicket keeper. For informational purpose, wicket keeper is special batsman with extra qualities.

Player class will be the parent class along with other classes like batsman, bowler and wicketkeeper. Club class is the driver class used for maintaining all the players. Batsman and bowler class inherit from player class and wicket keeper inherits from batsman class. Batsman, bowler and wicketkeeper will have different variables/data types to differentiate between them.

This will let the user (manager) choose player and see their profiles and attributes. Each player will be organized according to their skills and attributes.

New: I added the text file using savePlayers function. All the players info inputted is stored inside this which makes a file Players.txt.

Three different vectos- Batsman,Bowler,Wicketkeeper. Print function used to print the list of players (for example: batsmanList,bowlerList,wicketkeeperList)

Classes:

1. **Player:** This will be the parent class. All the instances of batsman, bowler, club and wicket keeper will have everything in common of players. The user can always add players into the list. Each player will have their own attributes.

2. **Batsman:** These are types of players whose role is to bat. They have special ability to bat that bowlers do not have. This is a child class. Inherits from player.

3. **Bowler:** These are types of players whose role is to bowl. They have special ability to bowl that batsman do not have. This is a child class. Inherits from player.

4. **Wicket Keeper:** Wicket keeper is a special kind of batsman with special fielding ability.
5. **Club:** It is the driver class. It is used to maintain all the players.

Player Class

Abstract class: No
Subclass of: N/A
Composed of: N/A

Data Members

Variable Name	Data Type	Static	Description
name	String	no	Holds first and last player name
age	integer	no	Holds age of player
country	string	no	Holds origin of player
fieldingAction	integer	no	Holds the ability/strength to show fielding skill
ground	string	yes	Manager of a particular(fixed) ground keeps track of players

Member Functions

Prototype	Static	Virtual	Overloading Operator	Friend of this class	Description
string getName()	No	No	No	no	Returns the data kept inside the variable name
integer getAge()	No	No	No	no	Returns the data kept inside the variable age
string getGround()	yes	no	no	no	Returns the data kept inside the variable ground Ground is common, so it is static

void setName()	No	No	No	no	sets the data inside the variable name
void setAge()	No	No	No	no	sets the data inside the variable age

Batsman Class

Abstract class: No
Subclass of: Player class
Composed of: N/A

Data Members

Variable Name	Data Type	Static	Description
battingHand	string	no	Left-handed or right-handed
batBrand	string	no	The company that sponsers the bat
favoriteShot	string	no	The best ball he can bowl

ground	string	yes	It is a common ground where all the players are managed by the manager. They play/practice on the same ground

Member Functions

Prototype	Static	Virtual	Overloading Operator	Friend of this class	Description

friend void getBatsmanInfo (Batsman bt)	No	No	No	Yes	Returns the data(battingHand, strength, batBrand and faoriteShot)kept inside the variable batsmanInfo Batsman info is a very essential information for the user. I want to retrieve and display all the information of the batsman object. A friend function can access all the private data members. So, this is a friend function.
virtual void fieldingAction()	No	Yes	no	no	Returns the data kept inside the variable fieldingAction Batsman class is the parent for wicket keeper class. Batsman's fieldingAction is a virtual function.
int getFavoriteShot()	No	No	No	no	returns the data inside the variable favoriteShot

String getGround()	yes	No	No	no	sets the data inside the variable ground It is a common ground where all the players are managed by the manager. They play/practice on the same ground

Club Class

Abstract class: No
Subclass of: N/A
Composed of: Player

Data Members

Variable Name	Data Type	Static	Description
batsmanList	Vector<Batsman>	No	Maintains all the batsman from the directory
bowlerList	Vector<Bowler>	No	Maintains all the bowler from the directory
wicketkeeperList	Vector<Wicketkeeper>	No	Maintains all the wicketkeeper from the directory

Member Functions

Prototype	Static	Virtual	Overloading Operator	Friend of this class	Description
void operator+=(Batsman *newBatsman)	No	No	Yes	No	Push back to add batsman
void operator+=(Bowler *newBowler)	No	No	Yes	No	Push back to add bowler
void operator+=(Wicketkeeper *newWk);	No	No	Yes	No	Push back to add wicketkeeper.

Bowler Class

Abstract class: No
Subclass of: Player class
Composed of: N/A

Data Members

Variable Name	Data Type	Static	Description
bowlingHand	string	no	Left-handed or right-handed
strength	integer	no	Holds the ability of performing well
type	string	no	Spin bowler or pace bowler
favoriteDelivery	string	no	The best ball he can bowl

Member Functions

Prototype	Static	Virtual	Overloading Operator	Friend of this class	Description
string getBowlingHand()	No	No	No	no	Returns the data kept inside the variable bowlingHand
integer getStrength()	No	No	No	no	Returns the data kept inside the variable strength
string getType()	No	no	no	no	Returns the data kept inside the variable type
int getFavoriteDelivery()	No	No	No	no	returns the data inside the variable favoriteDelivery
void setType()	No	No	No	no	sets the data inside the variable type

void setBowlingHand()	No	No	No	no	sets the data inside the variable bowlingHand

Wicket-Keeper Class

Abstract class: No
Subclass of: Batsman class
Composed of: N/A

Variable Name	Data Type	Static	Description
keepingHand	String	no	Holds preferred area of strength
glovesBrand	string	no	Spin bowler or pace bowler
fieldingAction	integer	no	Fielding ability of the wicket keeper. For ex: diving

Member Functions

Prototype	Static	Virtual	Overloading Operator	Friend of this class	Description
string getKeepingHand()	No	No	No	no	Returns the data kept inside the variable keepingHand
string getGlovesBrand()	No	no	no	no	Returns the data kept inside the variable glovesBrand
void fieldingAction()	No	No	No	no	sets the data inside the variable fieldingAction it is overriding the fieldingAction of the

					batsman class. Batsman class is the parent for wicket keeper class.
string setGlovesBrand()	No	no	no	no	Sets the data inside the variable glovesBrand

Demonstration of OOP Concepts

1. Encapsulation: (**It is demonstrated in all the classes I made**)
   a. All the classes are encapsulated.
   b. This means, all their attributes are private

2. Inheritance: (**check all the .h files (except the Player class which is the parent class) you will see inheritance with comments**)
   a. Batsman inherits from the parent Player class
   b. Bowler inherits from the parent Player class
   c. Wicketkeeper inherits from the Batsman class

3. Polymorphism:(**check Player.h line 36- virtual string getFieldingAction();**
   a. The wicket keepers are special kind of batsman with special fielding ability.
   b. For example: the function fieldingAction is overridden.

4. Static Member/Functions: (**check Player.h line 18- static const string ground;)**
   a. It is a common ground where all the players are managed by the manager/user. They play/practice on the same ground.
   b. For example: integer getGround() is static because all the players practice at a common practice ground. “RPL” is the ground name in the program.

5. Friend Functions: (**Check Batsman.h line 43- friend void getBatsmanInfo(Batsman bt);**)

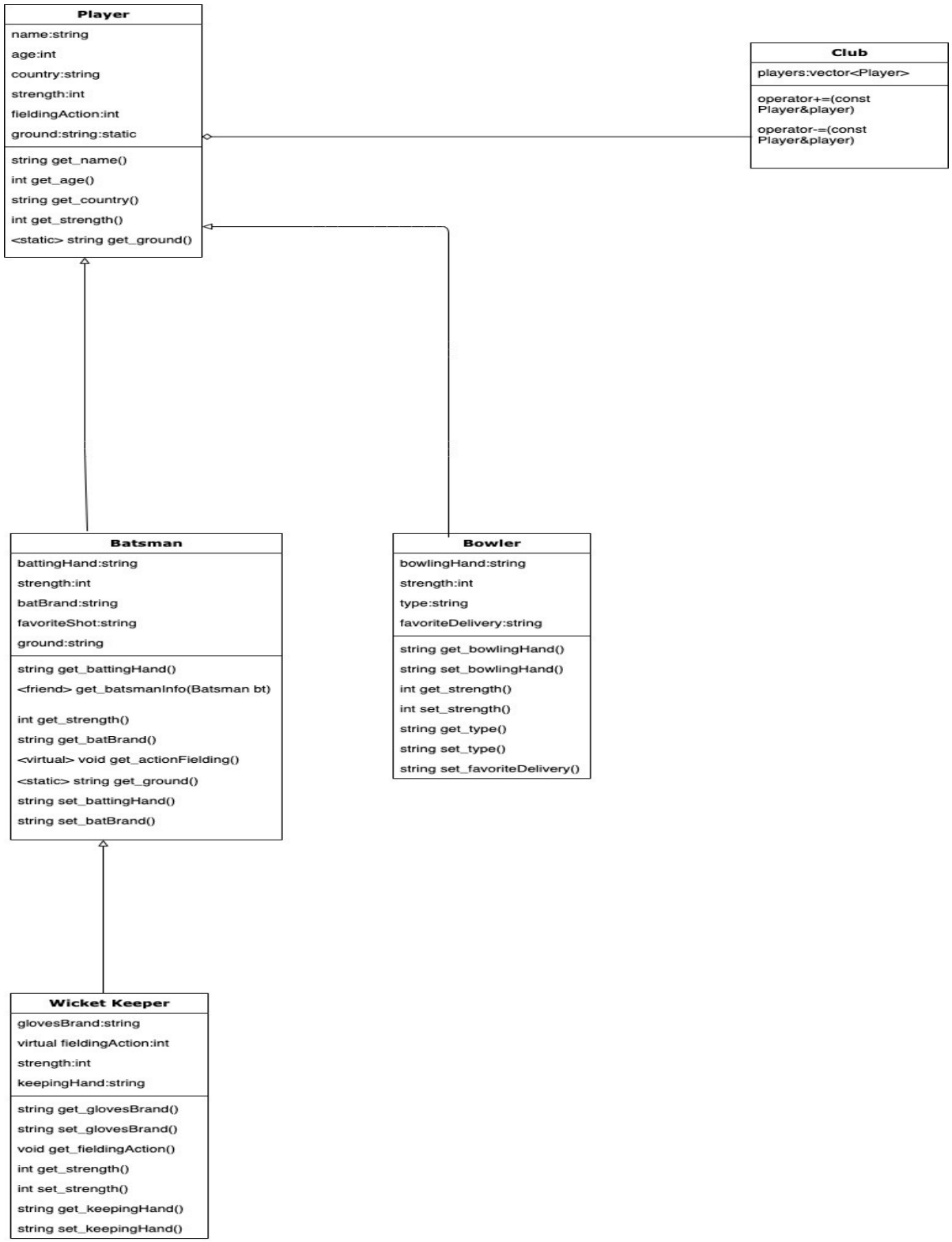
   A friend function has the right to access all private and protected members of the class. Batsman info is very essential information for the user. I want to retrieve and display all the information of the batsman object. This is why I am made this a friend function.

   So, friend void getBatsmanInfo() is a friend function.

6. Overloaded operators: (Club.h line 39,41,43)
   The user can add as many players as he wants to the directory using += operator.

7. Text file(s): (main.cpp line 182)
   We are writing the directory of players after the program is completed for future access (names, age, etc….). The file name is “Players.txt.”

UML


Player
name:string
age:int
country:string
strength:int
fieldingAction:int
ground:string:static
string get_name()
int get_age()
string get_country()
int get_strength()
<static> string get_ground()
Club
players:vector<Player>
operator+=(const Player&player)
operator-=(const Player&player)
Batsman
battingHand:string
strength:int
batBrand:string
favoriteShot:string
ground:string
string get_battingHand()
<friend> get_batsmanInfo(Batsman bt)
int get_strength()
string get_batBrand()
<virtual> void get_actionFielding()
<static> string get_ground()
string set_battingHand()
string set_batBrand()
Bowler
bowlingHand:string
strength:int
type:string
favoriteDelivery:string
string get_bowlingHand()
string set_bowlingHand()
int get_strength()
int set_strength()
string get_type()
string set_type()
string set_favoriteDelivery()
Wicket Keeper
glovesBrand:string
virtual fieldingAction:int
strength:int
keepingHand:string
string get_glovesBrand()
string set_glovesBrand()
void get_fieldingAction()
int get_strength()
int set_strength()
string get_keepingHand()
string set_keepingHand()