Lecture# 13 outline and homework

Today's Content:

- **Inheritance**

```
void printall(void);///A gloabal function which we make friend of player class

class Date {
	int year;
	int month;
	int day;

public:
	Date(int y = 1, int m = 1, int d = 1)
	{
		cout << "\nIn date constructor\n";
		year = y;
		month = m;
		day = d;
	}
	void printdate()
	{
		cout <<"\nDate: "<< year << ":" << month << ":" << day << endl;
	}

};


class player//Parent...super...base...Geralized....class
{
	int Id;//		//> .
	char name;//		//> .
	int size;//		//>  non-static and  non-constant data members
	int *Scores;//		//> .
	float Average;//	//> .

	static int count;  //static data members

	const char gender;//Constant data member

	//Date DoB; ///composition
	Date *DoB; ///composition

	//Date &DoM;///aggregation

	Date *DoM;///aggregation

public:
```

```
	player(Date *, int =1, int = 1, int= 1, int = 0, char = 'a', int s = 2, char = 'M', int * = NULL);//Default parameterized constructor

	////Copy constructor//discuss during lecture
	player(const player&);

	// .......  Utility Functions ........
	player& calAverage(void);
	player& print(void);

	//......  Setter or Mutator Functions ......
	void setId(int);
	void setName(char);
	void setsize(int);
	void setScores(int *);//interesting

	// ..... Accessor or Getter functions .......
	int getID(void) const;
	char getName(void) const;
	float getAverage(void);
	int getsize(void) const;
	//How to write getscores function ?????

	static void showcount() // static function
	{
		//cout << name;
		cout << "\nValue of count" << count;
	}

	~player(); //Destructor

	///operator overloading
	void operator=(const player &);

	///implement here other arithmetic operators like operator-, operator*, operator/, operator%, operator--

	friend void printall();//Granting printall() function as friend of class player

};

class cricketplayer : public player {//child...sub...derived...specialized....class

	int ranking;
	char type;//B->bat,b->bowlwer,A->allrounder
public:
	cricketplayer(Date *, int = 1, int = 1, int = 1, int = 0, char = 'a', int s = 2, char = 'M', int = 100, char = 'A', int * = NULL);

	void  print();///redefination of print() function

	~cricketplayer() {
		cout << "\nIn Cricket-Destructor\n";
	}
	//
};
```

```
////////////////////// . . . define player class functions out of line/scope . . .
////////

int player::count = 0;//assigning value to static data member of class

player::player(Date *dm, int y, int m , int d, int i, char n, int s, char g, int *arr) :
Id(i), name(n), size(s), gender(g)//Constant data member must need intilizer with
constructor
{
	DoM = dm;

	cout << "\nInside parameterized player Constructor : \n";

	DoB = new Date ( y,m,d );

	if (arr == NULL)
	{

		Scores = new int[size];
		cout << "Enter values of " << size << "  player : ";

		for (int i = 0; i < size; i++)
		{
			cin >> Scores[i];
		}

	}
	else
	{
		Scores = new int[size];
		cout << "Enter values of   " << size << " students : ";
		for (int i = 0; i < size; i++)
		{
			cout << "\nEnter " << i + 1 << " Value : ";
			cin >> Scores[i];
		}

	}
	player::calAverage();//calculating average in constructor
	count++;
}



///Defination of copy constructor
////Copy constructor
player::player(const player & p) :gender(p.gender)
{
	cout << "\nIn Copy Constructor\n";
	this->Id = p.Id;
	this->size = p.size;

	this->Scores = new int[this->size];
	cout << "\nEnter " << size << " Values for scores";
	for (int i = 0; i < this->size; i++)
	{
```

```
			cout << "\nEnter " << i + 1 << " Value : ";
			cin >> this->Scores[i];
		}
		player::calAverage();
		count++;
}



// .......  Utility Functions ........

player& player::calAverage(void)
{

		cout << "\nInside CalculateAverage() Function\n";
		int s = 0;
		for (int i = 0; i < size; i++)
		{
			s += Scores[i];
		}
		this->Average = float(s) / size;

		return *this;
}
player& player::print()
{
		DoB->printdate();
		DoM->printdate();
		cout << "\n.............................";
		cout << "\nInside print() function";
		cout << "\nName of player is :   " << name;
		cout << "\nID of player is :   " << Id;
		cout << "\nTotal matches played are : " << size;
		cout << "\nScores of player is:   ";
		for (int i = 0; i < size; i++)
		{
			cout << Scores[i] << " ";
		}
		cout << endl;

		cout << "\nAverage of player is:   " << this->Average;
		cout << "\n.............................\n";
		return *this;
}

//......  Setter or Mutator Functions ......
void player::setId(int i)
{
		cout << "\nInside setId() function";
		Id = i;
}

void player::setName(char c)
{
		cout << "\nInside setName() function";
		//name = c;

}
```

```
void player::setsize(int s)
{

	cout << "\nInside setsize() function";
	this->size = s;
}
void player::setScores(int *arr)
{
	cout << "\nInside setScores() function";
	delete[] Scores;
	Scores = NULL;
	Scores = new int[size];

	for (int i = 0; i < size; i++)
	{
		Scores[i] = arr[i];
	}


}
// ..... Accessor or Getter functions .......
int player::getID(void) const
{
	cout << "\nInside getId() function\n";
	return this->Id;
}
char player::getName(void) const
{
	cout << "\nInside getName() function\n";
	return name;
}
float player::getAverage()///an interesting fact inside function
{
	cout << "\nInside getAverage() function\n";
	player::calAverage();
	return Average;

}
int player::getsize(void) const
{
	return (*this).size;//return this->size;
}

//Definition of Destructor
player::~player() //Destructor
{
	cout << "\nInside Player class Destructor \n";

	delete[] Scores;
	delete DoB;
	count--;
}
void player::operator=(const player &p)
{
	this->Id = p.Id;
	this->size = p.size;
	delete[] Scores;
```

```
	this->Scores = new int[size];
	cout << "\nEnter " << size << " Values for scores";
	for (int i = 0; i < this->size; i++)
	{
		cout << "\nEnter " << i + 1 << " Value : ";
		cin >> this->Scores[i];
	}
	player::calAverage();
}

//.......Defination of printall() global function ..........//
//Read this function carefully and implement it in main ..............///
void printall()
{
	cout << "\n......Inside printall() global function.......\n";
	Date *pt = new Date{ 2,2,2 };
	player p(pt, 2, 7, 11, 3, 'A', 4);

	cout << "\n.............................";
	cout << "\nInside print() function";
	cout << "\nName of player is :   " << p.name;
	cout << "\nID of player is :   " << p.Id;
	cout << "\nTotal matches played are : " << p.size;
	cout << "\nScores of player is:   ";
	for (int i = 0; i < p.size; i++)
	{
		cout << p.Scores[i] << " ";
	}
	cout << endl;

	cout << "\nAverage of player is:   " << p.Average;
	cout << "\n.............................\n";
}
//////////cricketplayer
cricketplayer::cricketplayer(Date *dm, int y, int m, int d, int i, char n, int s, char g,
int r, char t, int *arr):player(dm, y, m, d, i, n, s, g)
{
	cout << "\n In cricket-Constructor\n";
	ranking = r;
	type = t;
}
void cricketplayer::print()//redefined print function
{
	cout << "\nIn Cricket-print():  \n";
	player::print();

	cout << "\nRanking  : " << ranking;
	cout << "\nType  : " << type;
}



///implementation of operator<<()
int main()
{
	//Date o1(23, 7, 17);
	// write your implementstion code here
```

```
        Date *d1 = new Date{ 2,2,2 };
        cricketplayer cp1(d1, 2, 7, 11, 3, 'A', 4,'b',3);//p2(p1);
        cp1.print();
        cp1.player::print();

        delete d1;
}
```