

Example of Java class

```
class LinkedList {  
    Node head; // head of list  
  
    /* Linked list Node*/  
    class Node {  
        int data;  
        Node next;  
  
        // Constructor to create a new node  
        // Next is by default initialized  
        // as null  
        Node(int d) { data = d; }  
    }  
}
```

A Java class has multiple members and member functions

Creating composite data types in C

- C programming language does not have 'class'
- Note: C++ supports OOP
- In C language, we can create user-defined composite data types as **structures**.

Structures in C

- A structure is a user defined composite data type in C
- A structure is used to group items of possibly different types into a single type
- Unlike Java/C++ classes, structures do not have member functions.
- Syntax is

```
struct tag_name {  
    T1 member1;  
    T2 member2;  
    /* declare as many members as desired,  
       but the entire structure size must be  
       known to the compiler. */  
};
```

Example1: Points with x and y coordinates

- A point has two coordinates: 'x' and 'y'
- In C we can create a new data-type 'Point' as

```
struct Point
{
    int x, y;
};
```

- Objects of type 'Point' can be created as

```
int main()
{
    struct Point p1; // p1 is an object of type Point
}
```

Example1: Points with x and y coordinates

- There is a shortcut for 'struct Point p1'

```
typedef struct Point
{
    int x, y;
} Point;
```

add **typedef** before **struct**

- Now, objects of type 'Point' can be created as

```
int main()
{
    Point p1; // p1 is an object of type Point
}
```

Accessing the members of a structure

- Structure members are accessed using dot (.) operator.

```
Point p1;  
// For p1(2,3)  
p1.x = 2;  
p1.y = 3;
```

Pointer to a structure object (1)

- We can create pointers to point to structure objects.
- If we have a pointer to structure, members are accessed using arrow (->) operator.
- Example: pointer to the point objects

```
Point p1;  
// For p1(2,3)  
p1.x = 2;  
p1.y = 3;  
printf("%d %d", p1.x, p1.y); // (.) used to access members
```

```
Point *p2;  
p2 = &p1; // p2 points to p1  
printf("%d %d", p2->x, p2->y); // (->) used to access members
```

Pointer to a structure object (2)

- We can dynamically allocate memory for structure objects
- Example:

```
Point *p;  
p = (Point *) malloc(sizeof(point));  
  
// Now assign coordinates (5,6) to *p  
p->x = 5;  
p->y = 6;
```