Classes, Objects & Linked List Intro

Programming Paradigm

Style or standard way of writing a program

Without programming paradigm

- less structured
- hard to read & understand
- hard to test
- difficult to maintain

OOPS

Store names & marks of students.

Use array : names : ["Alice", "Bob", "Charlie"]

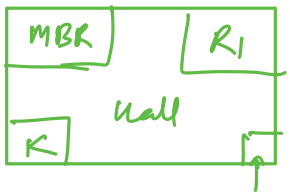
marks : [95, 92, 78]

Scalability issues

Maintainability issues

Data Association

Classes & Objects



Class → Blueprint of an idea

eg → floor plan of an apartment

Object → Real instance of class

same properties & abilities defined in blue print

1. Class takes no space in memory
2. Not a real entity
3. Multiple entities/instances of a same class.

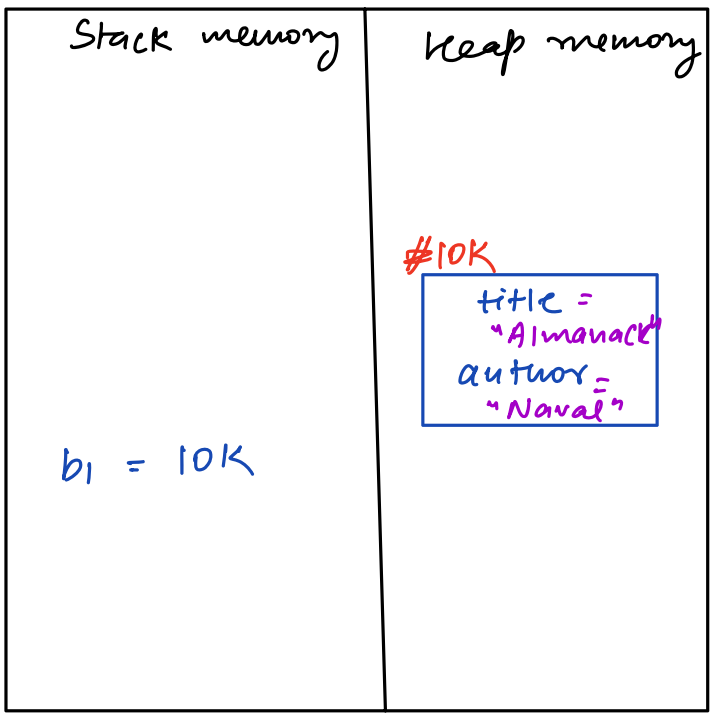
Class in Java

```
class Book {
    String title;
    String author;

    void read() {
```

```
        System.out.println ("Reading");
    }
    void buy() {
        System.out.println ("Bought");
    }
}
```

```
main() {
    //creating object
    Book b1 = new Book();
}
```

user-defined datatype



b1.title = "Almanack"

b1.author = "Naval"

```
print(b1.title)
print(b1.author)
```

Class in Python

```
class Book:
    title = ""
    author = ""

    def read(self):
        print("reading")

    def buy(self):
        print("bought")


b1 = Book()

b1.title = "Almanack"
b1.author = "Naval"
```

Constructors

```
class Student {
    String name;
    int age;
    double psp;
}
```

```
Student st = new Student();
```

Student → datatypee
st → variable name
Student() → default constructor

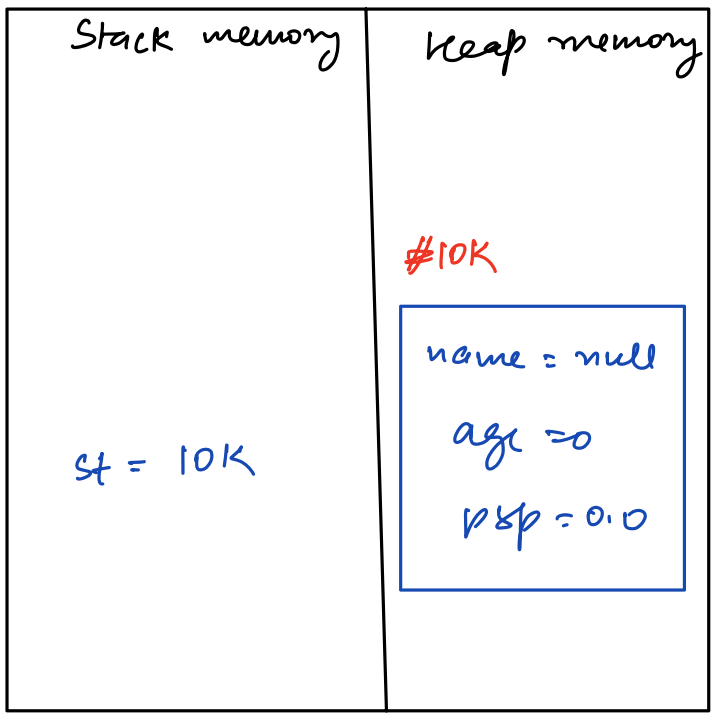
Default Constructor

Creates new object & assign default value to data members / variables.

- int → 0
- String → null / ""
- float → 0.0

```
class Student {
    String name;
    int age;
    double psp;

    Student() {
        name = null;
        age = 0;
        psp = 0.0;
    }
}
```

} not created by us



Is there any condition when default constructor is not created ?

→ if we create our own constructor, then default will not be created.

Do we pass any parameter to default constructor ?

→ NO, only assign default values

Anything special in constructor name ?

→ Same as class name

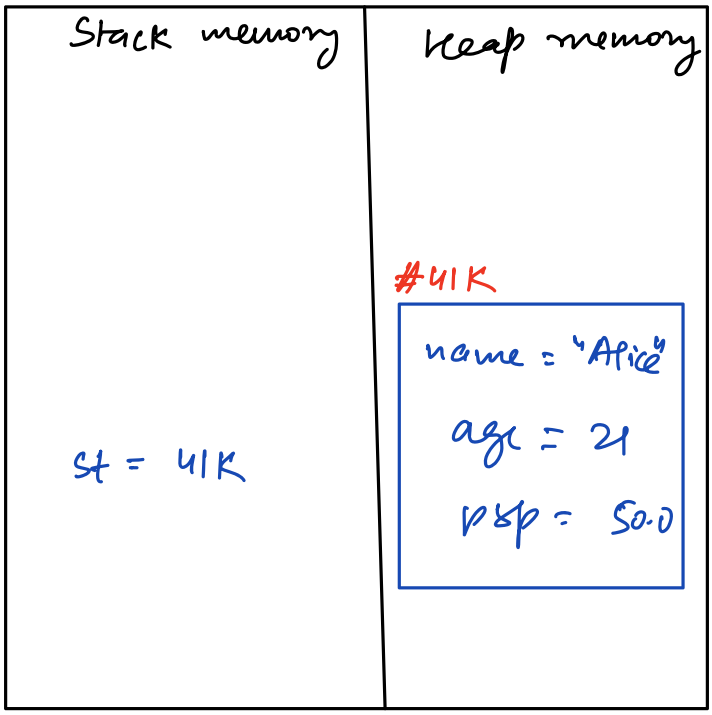
Return datatype of constructor ?

→ Class name itself

Default constructor →
1. No parameters
2. Name is same as class name
3. Datatype return is class name

Non-parameterized Constructor

```
class Student {
    String name;
    int age;
    double psp;

    Student() {
        name = "Alice";
        age = 21;
        psp = 50.0;
    }
}
```




```
Student s1 = new Student()
```

Parameterized Constructor

```
class Student {
    String name;
    int age;
    double psp;

    Student(String sname, int sage, double spsp) {
        name = sname;
```

```
        age = sage;
        psp = spsp;
    }
}
```

```
Student s1 = new Student("Uttej", 28, 90.0)
```

```
class Student {
    String name;
    int age;
    double psp;

    Student(String name, int age, double psp) {
        this.name = name;
        this.age = age;
        this.psp = psp;
    }
}
```

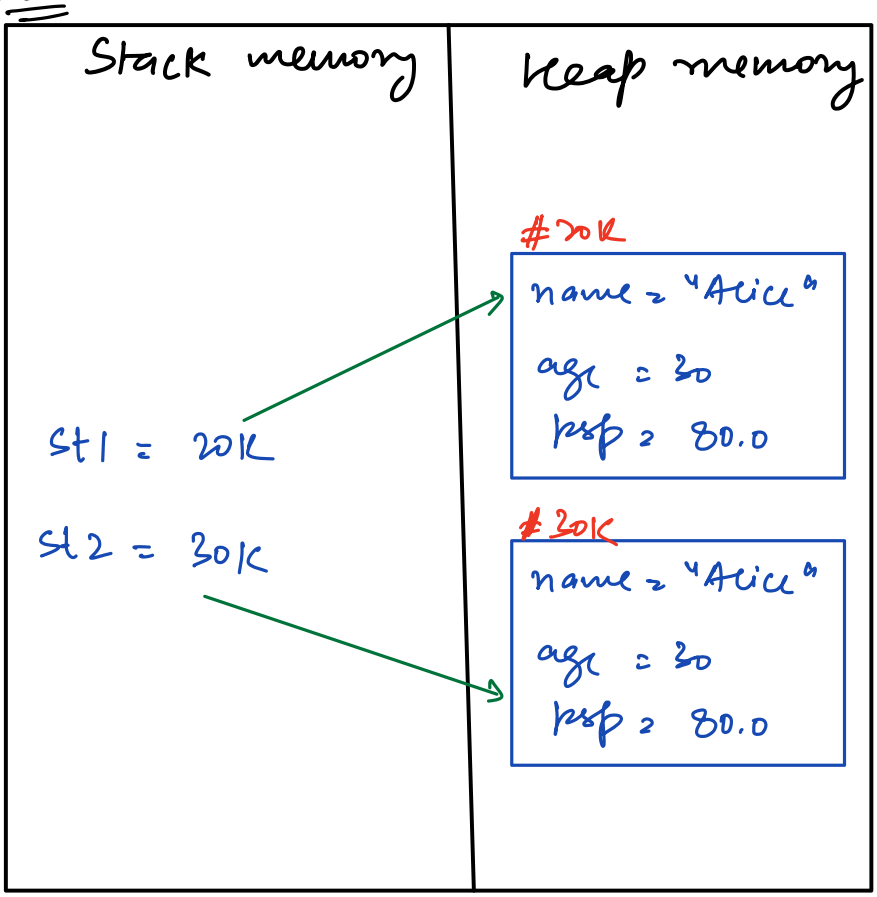
Constructor in Python

```
class Student:

    def __init__(self):          # non-parameterized
        self.name= "Alice"
        self.age = 20
        self.psp = 50.0

    def __init__(self, name, age, psp):
        self.name = name
        self.age = age
        self.psp = psp

s1 = Student("sriram", 30, 90.0)
```

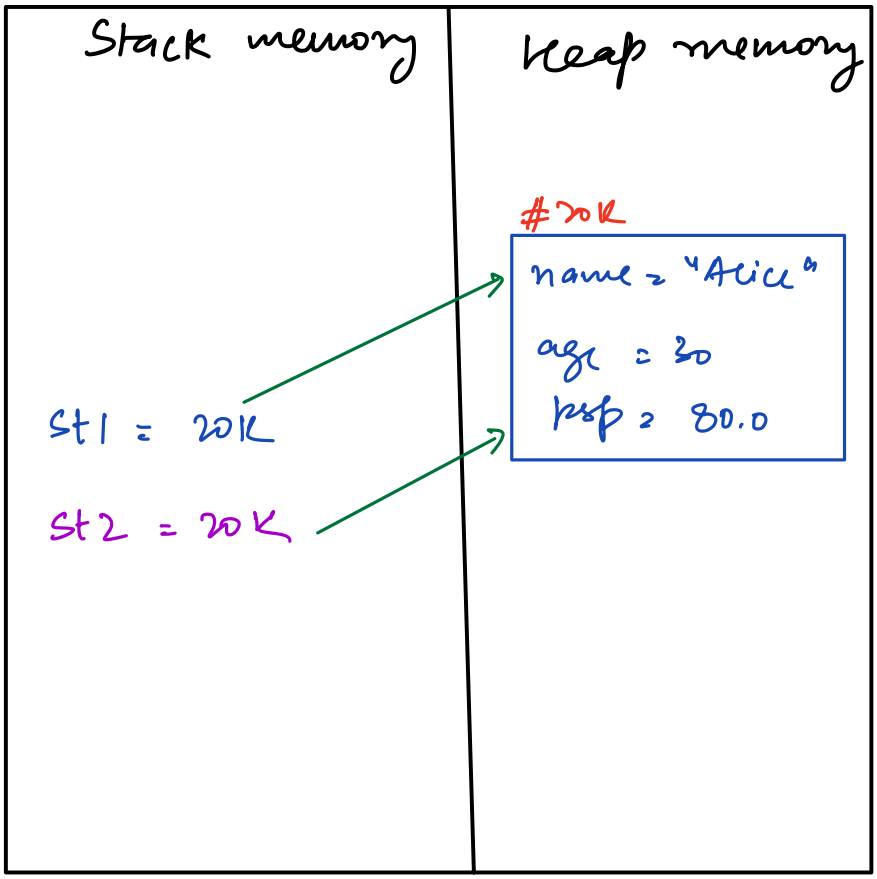
Shallow copy / Deep copy

We already have object of class, we want to create new object with same values.

Goal



Student st2 = st1 ;



- new object is not created & St2 points to same object as St1.

St2.name = "Bob"

print (St1.name) ⇒ "Bob"

] Shallow Copy

Deep copy

St2 = new Student("Alice", 30, 80.0)

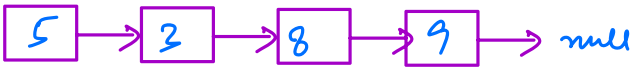
OR

new Student(St1.name, St1.age, St1.psp)

st2 = new Student(st1) ??

```
class Student {
    String name;
    int age;
    double psp;

    Student(String name, int age, double psp) {
        this.name = name;
        this.age = age;
        this.psp = psp;
    }

    Student(Student s) {
        this.name = s.name;
        this.age = s.age;
        this.psp = s.psp;
    }
}
```

Linked list



Arrays / Dynamic arrays

continous memory location

Linear DS that utilizes free memory in best way is Linked list.

Representation of LL (Linked list)


```
class Node {
    int data;
    Node next;
    Node (int x) {
        data = x
```

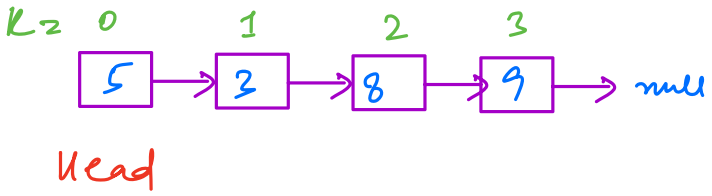
Head

```
        next = null
    }
}
```

```
Node Head = new Node (5);
Head.next = new Node (3);
Head.next.next = new Node (8);
```

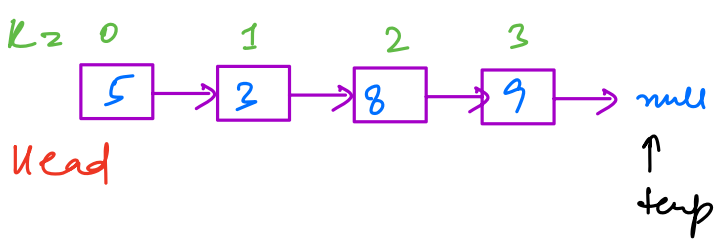
Question

Access K^{th} element of LL ($K >= 0$)



Array → a[K] TC=O(1)

```
Node get(Node head, int K) {

    Node temp = Head;

    for( i=0 to K-1) {  // K times
        if ( temp == null)
            return null
        temp = temp.next
    }

    return temp;
}
```



if $K >= n$ ⇒ null pointer exception

$T(= O(K) \Rightarrow O(\min(K, n))$