```
class string_test {

    public static void main(String argv[]) {
        String string_array[] = new String[10];
        string_array[0] = new String("Hello ");
        string_array[1] = new String("Goodbye");
        string_array[2] = new String();
        System.out.println(string_array[0]);
        System.out.println(string_array[1]);
        System.out.println(string_array[2]);
    }}
```



```
class Test {

    private  int x;
    public void set(int new_x) {x = new_x;}
    public int get() {return(x);}
}
class string_test {
    public static void main(String argv[]) {
        Test test_array[] = new Test[10];
        test_array[0] = new Test();
        test_array[1] = new Test();
        System.out.println(test_array[0].get());
        System.out.println(test_array[1].get());
        System.out.println(test_array[2].get());
    }}
```



```
0
0
Exception in thread "main" java.lang.NullPointerException
        at string_test.main(string_test.java:12)

Process string_test exited abnormally with code 1
```



```
class Point {
    private int x , y;
    public Point(int n_x, int n_y) {
        x=n_x; y=n_y;
    }
    public void plot() {
        System.out.print("x=");
        System.out.print(x);
        System.out.print(" y=");
        System.out.println(y);

    }
    public void set(int new_x,int  new_y) {
        x = new_x; y = new_y;
    }
}
```

Output slightly modified

```
p is x=3 y=4
q is x=3 y=4

p is x=10 y=23
q is x=10 y=23

p is x=42 y=42
q is x=42 y=42

Process point_test finished
```


Passing objects to Functions

```
class point_test {
    public static void do_something(Point z) {
        z.translate(12,23);
    }
    public static void main(String[] argv) {
        Point p = new Point(3,4);
        System.out.print("p is ");
        p.plot();
        do_something(p);
        p.plot();
    }
}
```



```
class point_test {
    public static void main(String[] argv) {
        Point p = new Point(3,4);
        Point q = p;
        p.plot();     q.plot();

        q.set(10,23);
        p.plot();  q.plot();

        p.set(42,42);

        p.plot();  q.plot();
    }
}
```


Java has pointers

```
Point p = new Point();
Point q = p;
```

Now `q` points to the object `p`.

With Java you can only make things point at each other, you can't do any other clever tricks with pointers.

Passing objects to Functions

```
p is x=3 y=4
x=15 y=27

Process point_test finished
```


Making Copies

Add another constructor

```
class Point {
    private int x , y;
    public Point(int n_x, int n_y) {
        x=n_x; y=n_y;
    }
    public Point(Point p) {
        x=p.x; y=p.y;
    }
}
```



```
class point_test {
    public static void do_something(Point z) {
        z.translate(12,23);
    }
    public static void main(String[] argv) {
        Point p = new Point(3,4);
        Point q =  new Point(p);
        q.plot();
        do_something(p);
        q.plot();
    }
}
```


Result

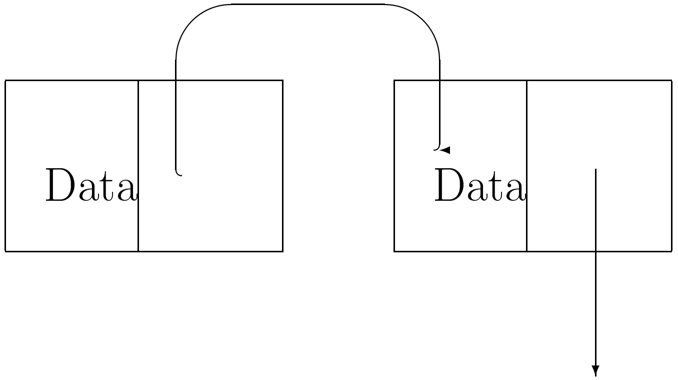
```
java point_test

x=3 y=4
x=3 y=4

Process point_test finished
```

Issues with Copying

- Very rarely done for you.
- Sometimes there is more than one answer to what is a copy. Suppose you had a list of names, do you want to copy just the list of pointers (references) to the names or the names as well.
- Java provides an `interface clone()` which you can use. (Best avoided at the moment stick to adding new constructors).
- Remember you are passing references around not objects.

Linked Lists

A linked list is a simple data structure, it allows you represent lists of objects that can grow an shrink.



The class Node1

```
class Node1 {

   private int data;
   private Node1 vidare;
   private boolean empty;

   public Node1(int in, Node1 vid) {
      vidare=vid;
      empty=false;
      data = in;
   }
```



```
   public Node1() { //Constructs an empty list
      empty=true;
   }
   public Node1 concat(int in) {
      Node1 n = new Node1(in,this);
      return n;
   }
   public int first() {
      return data;
   }
  public Node1 rest() {
      return vidare;
   }
   public boolean isEmpty() {
      return empty;
   }      }
```

Recursive Algorithms on Lists

When defining algorithms on lists, you need to think about what a list is.

A list is either:

- Empty
- Or has an element and a pointer to the rest of the list.

Example length of a list

```
class node_test {
    public static int length(Node1 list) {
        if (list.isEmpty()) return(0);
        else
            return(1+length(list.rest()));
    }
    public static void main(String[] argv) {
        Node1  empty_node = new Node1();
        Node1  meaning_of_life = new Node1(42,empty_node);
        Node1  fred = new Node1(333,meaning_of_life);
        System.out.println(length(fred));

    }}
```


Example Use

```
class node_test {
public static void main(String[] argv) {
        Node1  empty_node = new Node1();
        Node1  meaning_of_life = new Node1(42,empty_node);
        Node1  fred = new Node1(333,meaning_of_life);
        System.out.println(fred.first());
        System.out.println(fred.rest().first());
   }
}
```

What is going on with `fred.rest().first()` ?

Recursive Algorithms on Lists

So you work out what the algorithm does

- On an empty list
- and on a list with an element and a pointer to the rest of the list.