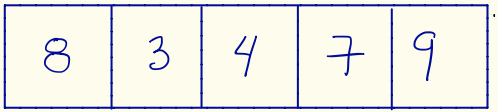
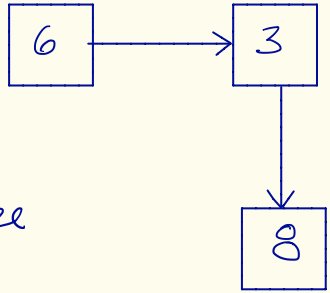
Memory

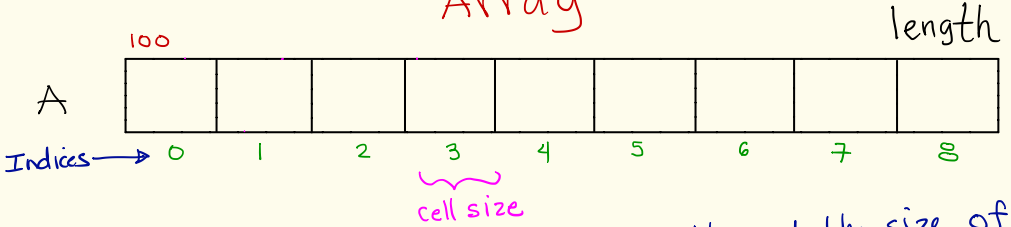




Data structure: way to organize data in memory

Memory

Array



Knowing the address of the first cell and the size of a cell we can compute the address of any cell:

address of cell with index i = address of first cell + i x size of a cell

```
int[] A;
A = new int[10];
for (int i=0; i< A.length; ++i)
    A[i] = 1;
```

Array declaration

Allocation of memory to the array

Array initialization

(A.length: number of cells in A)

We need an additional variable to count the number of values that have been stored in the array.

Lists and Arrays

- An array has fixed size
- All elements of an array are of the same type

```
public class Person {

    /* Attribute declarations */
    private String lastName;          // last name
    private String firstName;         // first name
    private String email;             // email address

    /* Constructor initializes the person's name and
       email address */
    public Person() {
        lastName = "";
        firstName = "";
        email = "";
    }
    public Person(String first, String last, String mail) {
        firstName = first;
        lastName = last;
        email = mail;
    }
```

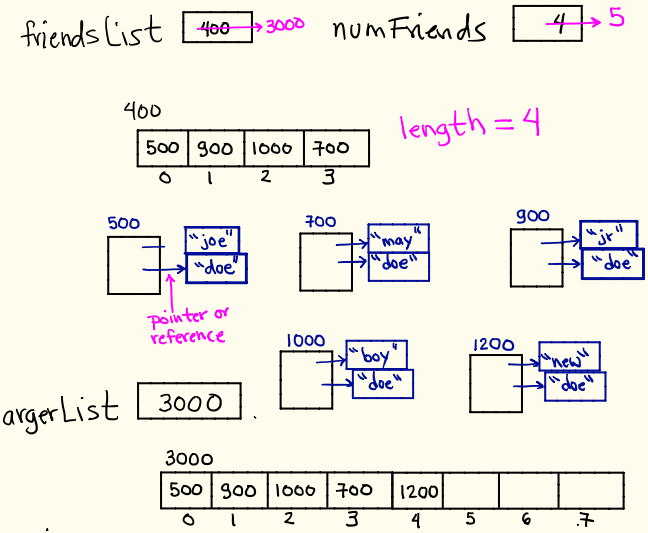
```
    public String getName() {
        return firstName + " " + lastName;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail (String mail) {
        email = mail;
    }

    public void setName(String newFirst, String newLast) {
        firstName = newFirst;
        lastName = newLast;
    }

    public boolean equals(Person other){
        if (this.firstName.equals(other.firstName) &&
                this.lastName.equals(other.lastName))
            return true;
        else
            return false;
    }

    public String toString() {
        String s = firstName + " " + lastName + "\t" + email;
        return s;
    }
```

```
public class SocialNetwork {

        // default size of array
        private final int DEFAULT_MAX_FRIENDS = 10;

        /* Attribute declarations */
        private Person[] friendList;    // list of friends
        private int numFriends;         // current number of
                                        // persons in list

 /**
  * Constructor creates person array of default size
  */
 public SocialNetwork () {
         friendList = new Person[DEFAULT_MAX_FRIENDS];
         numFriends = 0;
 }

 /**
  * Constructor creates person array of specified size
  * @param max: size of array
  */
 public SocialNetwork(int max) {
         friendList = new Person[max];
         numFriends = 0;
 }
```

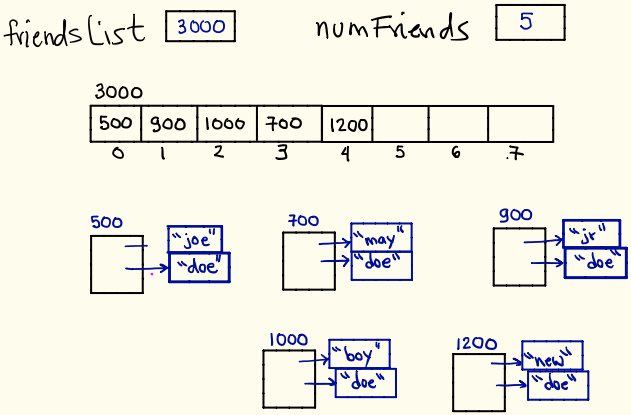
Constant

```
public void add (Person newFriend) {
    if ( numFriends == friendsList.length )
        expandCapacity();
    friendsList[numFriends] = newFriend;
    numFriends++;
}
```

```
private void expandCapacity() {
    Person[] largerList;
    largerList = new Person [2*friendsList.length];
    for (int i=0; i<friendsList.length; ++i)
        largerList[i] = friendsList[i];
    friendsList = largerList;
}
```



Algorithm remove (target)
In: Object to remove from array friendList
Out: true if target is removed; false otherwise




```
public boolean removePerson target() {
    // search the list for the specified friend
    int i = 0;
    while (i < numFriends) && !friendList[i].equals(target){
        i++;
    }
    if (i >= numFriends) return false;
    else {
        // person found, remove by replacing with last one
        friendList[i] = friendList[numFriends - 1];
        numFriends --;
        return true;
    }
}
```

Two Dimensional Arrays

Scope of a variable

```
public class Test {
    private int var1;
    private int i;
    public Test() {
        var1 = 7;
        i = 10;
    }
    private void algo1(int j){
        i = j;
        j = 14;
    }
    private void algo2(int i) {
        i = 20;
        var1 = 100;
    }
    public void algo3(){
        int var1 = 2, j = 4;
        for(int i=1; i<3; ++i)
            var1 = var1 + i;

        int k = i;
        algo2(5);
        System.out.println(i);
        algo1(j);
        System.out.println(j+" "+i)
        System.out.println(var1);
    }
    public static void main(String[] args) {
        Test t = new Test();
        t.algo3();
    }
}
```