ADT Doesn't specify how ADT will be implemented

Abstract Data Type

Model of a data structure that shows how data can be collected and what operations can be done.

Outline of what your class will affect and how it might be bilt

Class

A template for creating different objects which defines its properties and behaviours

Object

Can be variables, functions, data structures

Constructor

Initializes newly created object.

Special methods that are called when an object is instantiated.

Declaring an Object

```
Object obj = new Object();
```

Declaring an Array

```
int[] intArray = new int[3];
                = {1, 2, 3};
                = new int[]{1, 2, 3};
```

```
double[][] doubleArray = new double[x][y];
 int[][][] Box = new int[x][y][z];
```

Advantages

- Represent multiple data items of a single type using one name
- Implement other data structures
- 2D arrays represent matrices

Disadvantages

- Need to know in advance how many elements there are because
- Arrays are fixed size
  - Memory allocated for array cannot be in/decreased
    causes a problem   memory space wasted
  - Elements are stored in fixed locations. So insertion/deletion difficult

Arrays stored in RAM is stored in Row Major Order

Row major order - elements 0 1 2 3 4 5 6 7 8 9 10 11

$\Downarrow$ stored

```
0 1 2 3
4 5 6 7
8 9 10 11
```

Column major order -

```
0 3 6 9
1 4 7 10
2 5 8 11
```

Declaring a Java Matrix

```
int[] Box = int[5][3]
```

Assign -1 to all 15 slots:

```
for (int i=0 ; i < Box.length; i++)
      for (j = 0; j < Box[0].length, j++)
Box[i][j] = -1;
```

} Nested for-loop

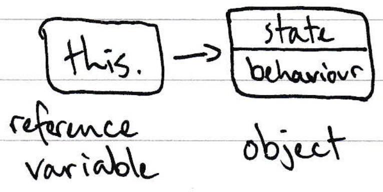
For single array

```
public static void main (String[] args) {

int[]array = {9, 3, 4, 1};

  for (int i = 0; i < array.length; i++) {
     int ARRAY = 0;
        ARRAY = array[i]; - passing as parameter
     System.out.println("Array = " + ARRAY);
  }


  System.out.println("Sum = " + SUM(array));
  System.out.println("Range = " + RANGE(array));
   // Finding sum
  private static int SUM (int array[]) {
     int sum = 0;
     for (int i = 0; i < array.length; i++)
        sum += array[i];
     return sum;
  }

   // Finding max, min
  private static int RANGE (int array[]) {
     int max = array[0];
     int min = array[0];

     for (int i=0; i < array.length; i++) {
        if (max < array[i])
           max = array[i];

        if (min > array[i])
           min = array[i];
     }
     int range = max - min;
     return range;
  }
}
```

Instance methods / variables occur once per instance of a class

Class methods / variables are associated with a class and occur once per class

```
                                   class name
                                  /
                public class Car {
Class members  /  private int door;           \
instance      —   private int speed;           } Class body
variables      \  private String colour;      /

Class members     public void run() {
instance     /    }
method
                }
```



<u>Overloading</u> allows the same method name to be reused when changing slightly different parameters

<u>StringBuffer ADT</u>

```
StringBuffer stringBuf = new StringBuffer();
        stringBuf.append("abc"); //stringBuf now contains "abc"
        stringBuf.append(123); //stringBuf now contains "abc123"

stringBuf = new StringBuffer("TH8");
stringBuf.insert(2,"x"); //stringBuf now contains "THx8"
stringBuf.insert(3,113); //stringBuf now contains "THx1138"
char ch = stringBuf.charAt(2);
stringBuf.setCharAt(2,'a'); //stringBuf now contains "THa1138"
```

indexOf() method

```
String str = new String("Hello World");
System.out.print("Found Index : ");
System.out.println(str.indexOf('d'));
```

```
   Found Index : 10
```

substring() method

```
System.out.print(str.substring(3));
                                (7)
   Found Index : lo World       (9)
                   orld
                     ld
```

compareTo() method

```
String str = new String("Hello World");
String str2 = new String ("Hello World not");

int result = str.compareTo(str2);
System.out.print("Difference in characters = " + result);
```

```
   Difference in characters = -4
```

Bitset ADT

Bitset requires less memory because each element is stored as a single bit.
 Will auto increase size as needed

```
Bitset multiplesOf2 = new Bitset(16);
      multiplesOf2.set(2); // set method changes a single bit to true
      multipliesOf2.clear(2); // clear method "clears" a bit by setting it to false
if (multiplesOf2.get(2))...   // get method returns a boolean value
                                 regarding whether a specific bit position is
                                 set
```