ArrayLists

1. ArrayLists vs Arrays
2. Where do they come from?
3. How do you construct them
4. What methods do they employ?

Arrays - often called static Arrays.
ArrayLists - often called dynamic Arrays.

// common interview question. save this info

1. RESIZABLE

Arrays have a static length. Once created, you cannot alter w/out creating a new array and copy elements over.

to change size of Array

ie:
```
int[] elems = new int[n]; // n some int
/* we run out of space in elem */
public static int[] double(int[] a) {
    int[] aa = new int[a.length*2];
    int in = 0;
    for (int e: a) {
        aa[in] = e;
        in++;
    }
```

Arraylists on the other hand are added and deleted to continuously. Capacity grows or shrinks.

2. Performance

Arrays are objects that are stored in contiguous hunks on the heap. Only in extremely large data sets w/ continuums get operations will this advantage be noticeable.

Array Lists are more flexible and yet in todays environment, they are not that much slower.

3. Primitives

Arrays can store primitives or objects. Arraylists store only objects. This is why we have wrapper classes like Double, Integer, etc.

ie: `ArrayList<Integer> list = new ArrayList<Integer>();`
`list.add(27); // auto wrapped new Integer(27)`

4. We can use loops to iterate through both. However, Arraylists can use iterators provided by Arraylist class. (not on AP exam)

5. Type safety

Arraylists in AP CS use Generics.
`ArrayList<E> list = new ArrayList<E>();`
this protects Arraylist from incompatible runtime errors.
Arrays are homogenous and any incompatibility is also caught @ compile time.

6. Length

to get length of Array arr use `arr.length` // field
to get length of Arraylist arr use `arr.size()` // method

7. adding elements

Array we use simple assign `arr[0] = 12`
Arraylist we use `arr.add(12)`
reminder: arrays may be full, adding to middle is cumbersome. Arraylist not full, easy middle insert.

Info on Arraylists

1. need import java.util.* // imported, Arrays not

2. Construction

```
List<String> list = new ArrayList<String>();
```

or

```
Arraylist<String> list = new Arraylist<String>();
// we will talk next semester about why List<String>
// is ok. Just accept it for now.
```

Creates an ArrayList of Strings but empty.
much like Gamer[] gdogs = new Gamer[5]
creates an array but no Gamer objects yet.
Note: capacity is 10 behind the scenes.
increased by doing capacity * 3/2 + 1.

3. methods
to check the current length of arraylist:

```
int sz = list.size(); // a method
```

to add to end:

```
List.add("world");
```

or add to front of index 0.

```
list.add(0,"hello");
```

3. methods con't

to grab an item (Arraylists only hold objects)

```
String s1 = list.get(0);
```

to replace an item

```
String s2 = list.set(1, "there"); //s2 holds "world"
// note both get and set return the E type.
// generic type Arraylist<E>
```

to print the Arraylist. EASY!

```
System.out.print(list);
```

prints: ["hello", "there"]

to delete an item from list.

```
String s3 = list.remove(0);
// s3 holds "hello"
```

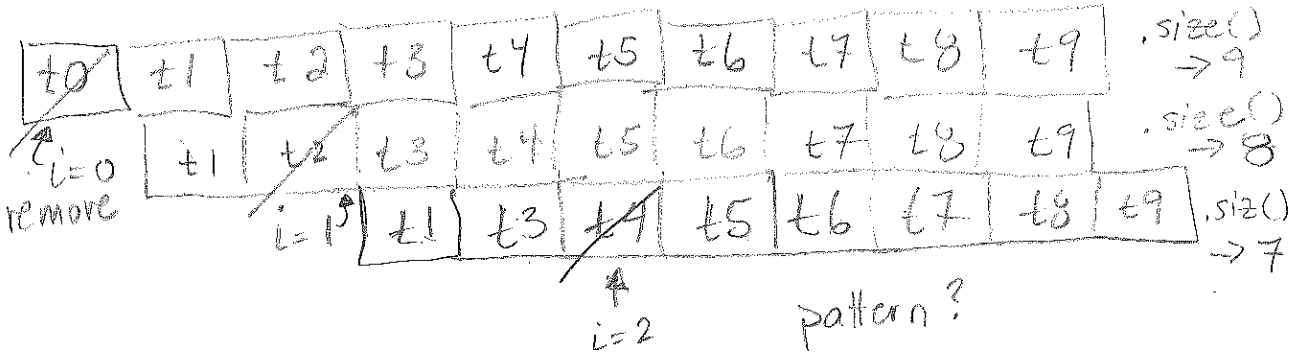
4. beware

```
Arraylist<Integer> Lints = new Arraylist<Integer>();
int i = 0;
while (i <= lints.size())
    lints.add(new Integer((int)(8*Math.random())));
    i++;
}
// infinite loop, size keeps growing
```

```
ArrayList<String> liStr = new ArrayList<String>();
for (int i=0; i< 10; i++) {
    liStr = "text" + i;
for (int i=0; i< liStr.size(); i++) {
    System.out.print(" " + liStr.remove(i));
}
// What is printed?
```

, text0 , text2 , text4 , text6 , text8

why?




```
int sum = 0;
for (int i = 0; i < liStr.size(); i++ ) {
    sum += liStr.get(i).length();
}
```

calling a method →

enhanced for version:

```
int sum=0;
for ( String s : liStr ) {
    sum += s.length();
}
```

04. Consider the following program segment.

Same as # 21

```
ArrayList<String> names = new ArrayList<String>();
names.add("Isolde");
names.add("John");
names.add("Greg");
names.add("Maria");
```

Which of the following statements display all the elements in the names array?

(A) `System.out.println(names);`

(B)
```
for (int index = 0; index < names.size(); index++)
    System.out.print(names.get(index) + ", ");
```

(C)
```
for (String name: names)
    System.out.print(name + ", ");
```

(D) All of the above

05. Consider the following program segment.

```
ArrayList<String> names = new ArrayList<String>();
names.add("Isolde");
names.add("John");
names.add("Greg");
names.add("Maria");
names.add("Heidi");

for (String name : names)
   name = "Qwerty";
System.out.println(names);
```

What is the output as a result of executing the program segment?

(A) `[Isolde, John, Greg, Maria, Heidi]`

(B) `[Isolde, John, Greg, Maria, Heidi, Qwerty]`

(C) `[Qwerty, Qwerty, Qwerty, Qwerty, Qwerty]`

(D) `Program crashes during execution with an exception error.`

06. Consider the following shuffle method that is part of a Deck class.

```
private void shuffle()
{
    for (int k = 1; k < 1000; k++)
    {
        int random1 = (int) (Math.random() * 52);
        int random2 = (int) (Math.random() * 52);
        Card temp = cards.get(random1);
        cards.set(random1,cards.get(random2));
        cards.set(random2,temp);
    }
}
```

Cards are stored in an array data structure, which is an attribute of the `Deck` class.
The array can be static or dynamic.
What is indicated by the shuffle method shown above about the cards array?

(A) cards is an object of the `ArrayList` class.

(B) cards is an object of a dynamic array, but not necessarily the `ArrayList` class.

(C) cards is an object of a static array.

(D) cards is an object of an initializer list.

(E) cards can be an object of a dynamic array or a static array.

07. Consider the following code segment.

```
ArrayList<String> names = new ArrayList<String>();
names.add("Isolde");
names.add("John");
names.add("Greg");
names.add("Maria");
names.add("Heidi");
names.set(3,names.get(4));
names.set(4,names.get(3));
System.out.println(names);
```

Isolde John Greg Maria Heidi
Heidi Heidi

What is printed as a result of executing the code segment?

(A) `[Isolde, John, Greg, Maria, Heidi]`

(B) `[Isolde, John, Greg, Heidi, Maria]`

(C) `[Isolde, John, Greg, Heidi, Heidi]`

(D) `[Isolde, John, Greg, Greg, Maria]`

(E) An `IndexOutOfBoundsException` error message

Free Response Question 01
You have 4 minutes to complete this question.

Consider the **Names** class below, which is a class to store and process a variety of names.

Complete method **swapNames** below, which swaps the names that correspond to the provided argument indexes.

```
class Names
{
    ArrayList<String> list;

    public Names()
    {
        list = new ArrayList<String>();
    }

    // additional methods to add, delete, and display elements in a Names object.

    // precondition:    p and q represent indexes of existing elements in the list array.
    // postcondition:   the elements, represented by p and q, have swapped locations.
    public void swapNames(int p, int q)
    {
        String s = list.set(p, list.get(q));  // set returns old elem!
        list.set(q, s);

    }
}
```

Assumed that since p & q rep indexes, they are valid and size() > 0!

Free Response Question 02
You have 4 minutes to complete this question.

Consider the **Names** class below, which is a class to store and process names.

Complete method **reverseList** below, which reverses all the names in the **list** array.
In completing method **reverseList** you may use method **swapNames**, described below.

```
class Names
{
    ArrayList<String> list;

    public Names() { list = new ArrayList<String>(); }

    // additional methods to add, delete, and display elements in a Names object.

    // precondition:    p and q represent indexes of existing elements in the list array.
    // postcondtion:    the elements, represented by p and q, have swapped locations.
    public void swapNames(int p, int q)

    public void reverseList()
    {
        int i = 0, k = list.size() - 1;
        while (i < k)
            swapNames(i, k);
            i++;
            k--;


    }

}
```