NOTES Intro to Data Structures

Data Structures & Other Objects Using Java (Main)

abstract data types (ADTs) - a class that has information hiding

information hiding - 1) encapsulation;

2) to know only as much as you need, but no more -

ex. you should know how different method interact/use each other in a class, but not how each method works specifically

~strong emphasis in info hiding is partly motivated by mathematical research into how programmers can improve their reasoning about data types that we use in programs

Java package - group of related classes that can be imported and used in code with a `.java` file name

How to write your own package:

1) pick a good short name

2) include your Internet domain name

ex. Defining `Throttle.java` in the `edu.colorado.simulations` Package

```
package edu.colorado.simulations; //package declaration
public class Throttle {
   //field: int top, position...
   //constructor...
   //methods: getFlow(), shift(int amount), shutOff(), etc
}
```

Example of writing a class into a package

```
import edu.colorado.simulations.*; //* imports ENTIRE package

import edu.colorado.simulations.Throttle;
//to use Throttle class
//Now you can make a Throttle instantiation:
Throttle small = new Throttle(size);
```

(ex. We've used `import java.util.*;` before... see next slide)

Java class libraries (JCL) - useful prewritten packages

examples:

java.lang is automatically imported into all Java programs:

--> interface things, class things (boolean, char, Object, String, etc.)

java.util *

--> Random, Scanner, Arrays, Date, Collections, Formatter, Timer

printf

java.text

--> formatting dates, numbers, money, etc.

~Use Java oracle online for help and to see APIs

application program interface (API) - set of routines, protocols, tools for building software applications; specifies how the components should interact (think of it as a wiki for code)

default access - when there is no explicit modifier, sometimes called package access... but we don't use this phrase too much...

~we use private instance variables with public methods

(protected access is in your future)

collection classes - a class in which each object contains a collection of elements

(when we talk more about Interfaces in the future, we will talk more about the Java Collection<E> class...

simplest collection class is Vector --> like an array, because it has indexed values, but it automatically grows if you place an object at a location beyond its current capacity)

array - (type of collection) dynamically-created object that serves as a container to hold constant number of values of the same type; memory space is allocated for values of a particular type

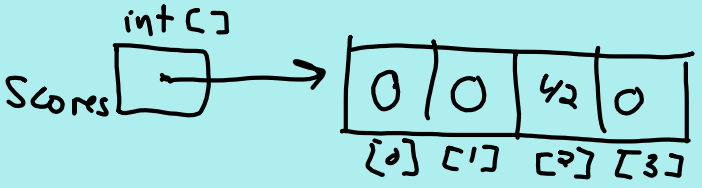
ARRAYS (Review)

Code example:

```
int [] = scores;  //declaring an array
scores = new int[4]; //initializing size

     //automatically filled with default values (zeros)
scores[2] = 42;
```

the size is immutable - not changeable



2 Common Array Exceptions:

1) `NullPointerException`

   a.) thrown during runtime (if reference is `null`)

   b.) thrown during compile time (if uninitialized local variables)

int[] a;
a[0]=

2) `ArrayIndexOutOfBoundsException`

   -trying to access, set, use, etc. index values that don't exist

look at Java Array API:

https://docs.oracle.com/javase/7/docs/api/java/util/Arrays.html

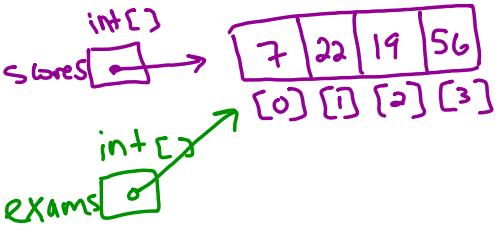
```
public static Object newInstance(Class<?> componentType, int length)
   throws NegativeArraySizeException;
   //if you try to create an array of negative size
```

component type - class object representing the component type of the new array

length - the length of the array; the number of elements in an array

Assignment Statements:

```
int[] scores;
int[] exams;
scores = new int[4];
scores[0] = 7;
scores[1] = 22;
scores[2] = 19;
scores[3] = 56;
exams = scores; //reference the same array
```



Array Parameters:

if the method that took an array as a parameter changes the components of the array, the changes affect the actual array argument (CHANGES EVERYWHERE! because it is a reference data type)

```
int[] arr = {1, 2, 3, 4}; //different way to initialize arrays
changes(arr);

//this changes everywhere
//because we sent in the
//reference into the
//method

public static void changes(int[] arr)
{
   arr[0]=10;
   arr[1]=20;
   arr[2]=30;
   arr[3]=40; //CHANGES EVERYWHERE!!
}
```

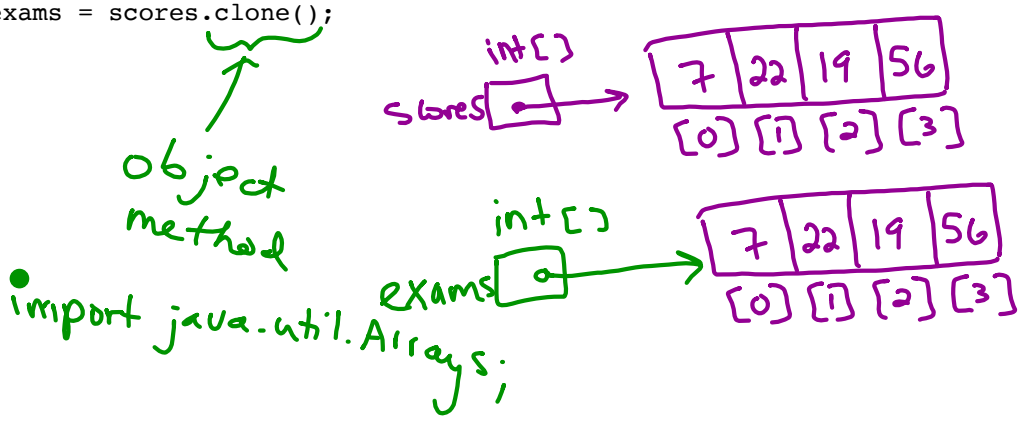
no return value

Clone Method:

//the clone method creates a copy of an object, separate from the

//original so you don't alter the original object

```
exams = scores.clone();
```



Enhanced for loops: "for each"

for (<data type> <variable name> : name of the array)

of the array's elements

```
public static int sum(int[] a)
{
   int answer;
   answer = 0;
   for(int item : a) //for each int element "item" in int array a...
      answer += item; //do this thing
   return answer;
}
```

Checking elements in an array / Traversing an array:

```
public static boolean search(double[] data, double target)
{
   for(double item : data)
   {
      if(item == target)
         return true;
   }
   return false;
}
```