Data Structures **Stacks:** can be used for keeping track of items. E.g., locker in use, locker available;  
**Dictionaries:** reminder, dictionaries can store “objects” or “lists of objects”. This will allow a “key” value to point to an int, string, (primitive type), or more complex object, or a list of int/string, or list of complex objects.  
**Arrays:** remember, you can initialize a lot of arrays, to store items that are available, unavailable, etc. Use this to compare items in this list, vs. items in another list.  
**Linked Lists:** remember, you can create nodes to store the previous node, current node, or next node. You can go through

**Test: CREATE A SET OF ALPHABET – test to see if which letters are in the array (Boolean and of two arrays?)**

Sorting  
**Bubble Sort  
Quicksort  
Mergesort?  
Insertion Sort  
Selection Sort  
Java7+ uses “Timsort” on Collections**

**Array sort:**import java.util.Arrays;  
int[] arr = {13, 7, 6, 45, 21, 9, 101, 102}; // Our arr contains 8 elements  
Arrays.sort(arr);  
  
// Sort subarray from index 1 to 4, i.e.,   
// only sort subarray {7, 6, 45, 21} and   
// keep other elements as it is.   
Arrays.sort(arr, 1, 5);  
  
**--------------------------------------------------------------------------**// Create a list of strings   
ArrayList<String> al = new ArrayList<String>();   
al.add("Geeks For Geeks");   
al.add("Friends");   
al.add("Dear");   
al.add("Is");   
al.add("Superb");

/\* Collections.sort method is sorting the elements of ArrayList in ascending order. \*/  
Collections.sort(al);  
  
/\* Collections.sort method of ArrayList in REVERSE order. \*/  
Collections.sort(al, Collections.reverseOrder()); **Array Initialization/usage - Sample 1:**String str = "geekss@for@geekss";   
String[] arrOfStr = str.split("@", -1);   
for (String a : arrOfStr)   
System.out.println(a);  
  
**Array Initialization/usage - Sample 2:**  
String alphabet = "a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,t,u,v,w,x,y,z";  
String[] arrAlphabet = str.split(",",-1);  
for (String letter : arrAlphabet)   
System.out.println(letter);

**String Comparison operations:**if (text.matches("[0-9]+") && text.length() > 2) { /\* doStuff \*/ }  
  
**Character to String:**  
Character.toString(myChar) **String to Char:**String g = "line";  
char c = g.charAt(0); // returns 'l'  
char[] c\_arr = g.toCharArray(); // returns a length 4 char array ['l','i','n','e']  
  
  
**Dictionary Initialization/usage:**Dictionary<Integer, Integer> myValues = new Hashtable<Integer, Integer>();  
myValues.put(1,1000);  
myValues.put(2,2000); //INSERT INTO DICTIONARY  
Integer temp = myValues.get(2); //CHECK DICTIONARY  
System.out.println(temp);  
  
myValues.remove(2); //DELETE FROM DICTIONARY  
  
**HashSet Initialization/usage:**import java.util.\*;Set<String> set = new HashSet<String>();  
set.add(“Learning”);  
set.add(“Easy”);  
Boolean checkExists = set.contains(“Learning”);  
  
  
//def ispangram2(str1, alphabet=string.ascii\_lowercase):  
//alphaset = set(alphabet)   
//return alphaset <= set(str1.lower())   
  
  
**List/ArrayList initialization/usage:**List<AmazonLocker> lockers = new ArrayList<AmazonLocker>();  
AmazonLocker tempLocker = new AmazonLocker(5);  
lockers.add(tempLocker);

**Printing:**System.out.println(“Hello, world!”);

**Main function:**  
public static void main (String[] args) {  
 //do stuff  
}  
  
**Class declaration:**public class AmazonPackage {   
 //private members/variables  
 //public constructor  
 //private/public methods

**Recursion**Commonly used with Lists/LinkedLists, in order to recursively traverse through the list;  
Can be used with arrays as well; **RegEx Example:**import java.util.regex.Pattern

Interfaces are IMPLEMENTED  
CLASSES can be INHERITED  
  
HackerRank Solutions by Rodney Shag:  
<https://github.com/RodneyShag/HackerRank_solutions>  
  
<https://github.com/RodneyShag/HackerRank_solutions/blob/master/Data%20Structures/Arrays/Left%20Rotation/Solution.java>

**JAVA Primitives vs. their Wrapper Classes:**  
byte has Byte  
short has Short  
int has Integer  
long has Long  
boolean has Boolean  
char has Character  
float has Float  
double has Double  
  
**JAVA String literal vs String object:**//STRING LITERALSString a = "Java";   
String b = "Java";   
System.out.println(a == b); // True  
  
//STRING OBJECTS  
String c = new String("Java");   
String d = new String("Java");   
System.out.println(c == d); // False