public static <E> boolean unique(List<E> list){ //takes in generic list and returns true if all items unique  
 for (int i = 0; i < list.size(); i++) { //start with item 1 in list  
 for(int j = i +1; j<list.size(); j++){ //check it against all other items  
 if (list.get(i).equals(list.get(j))){ //compare items, if not equal return false  
 return false;  
 }  
 }  
 }  
 return true; //if nothing is the same, return true - quadratic  
}

This would be O(n2) – quadratic.

public static List<Integer> allMultiples(List<Integer> list, int num){ //loop through items to check for multiples of num  
 List<Integer> updateList = new ArrayList<>(); //create empty array list to add to  
 for (Integer item : list) { //loop through each item in list  
 if (item % num == 0) { //check to see if item in list is divisible by num  
 updateList.add(item); //if so add to updated list  
 }  
 }  
 return updateList; //return updated list - linear  
}

This would be O(n) – linear.

public static List<String> allStringsOfSize(List<String> list, int num) { //loop through items and add all strings of size num to new list  
 List<String> updateList = new ArrayList<>(); // create empty list to add to  
  
 for (String s : list) { //loop through each item in list  
 if (s.length() == num) { //if item is the size num  
 updateList.add(s); //add item to updatedList  
 }  
 }  
 return updateList; //return updated list - linear  
}

This would be O(n) – linear.

public static List<String> stringToListOfWords(String word){ //take string and divide it into its words  
 List<String> newList = new ArrayList<>(); //create empty list to add to  
 String[] list = word.split("\\s"); //strip the space between these words  
 for (String s : list) { //goes through the list word by word //O(n)  
 newList.add(s.replaceAll("[^a-zA-Z]", "")); //adds it to a new list with punctuation stripped

//O(n)   
 }  
 return newList; //return newList  
}

This would be O(n2) – quadratic.

public static <E> void removeAllInstances(List<E> list, E item){ //takes in a list and removes instances of item  
 boolean repeat = true; //set repeat to be true initially to start loop  
 while(repeat) { //while repeat is true stay in loop  
 repeat = false; //each time around repeat is set to false unless if statement trips  
 for (int i = 0; i < list.size(); i++) { //for each item in list  
 if (list.get(i).equals(item)) { //if item in list equals generic item parameter  
 list.remove(i); //remove item  
 repeat = true; //then set repeat to true to loop back through again to catch any repeats that may have been missed by indices changing  
 }//end if  
 }//end for  
 }//end while  
}//end method

I think this may be O(k \* n2). The reason I was thinking this was you get O(n) for the for loop of list.size(), then you get another O(n) for the remove method which includes a for loop if the indices need changed, and then it repeats in the while loop for the number of instances of the number which is then indicated by k.