CSE 8B Today... and Forever!

Do you plan to take another CSE course?
A. Yes, definitely
B. Probably
C. Maybe
D. Probably not
E. No, definitely not

PSA8 due today with slip day!
(you can “buy” one slip day for -3 points)

Final exam prep:
• Review sheet from discussion posted (see website)
• If you want more exercises to practice: today slides AND check problems at the end of each chapter in the book (and in the book website link, you have a lot of short-questions to practice)
• BUT, the MOST important thing:
  REDO CLICKER QUESTIONS, WORKSHEETS (exercises to write/draw in class) and PSAs!!
Announcements

• You may only use 1 slip day!! even if you have 2, you can just use 1 slip day per PSA.
• You may “buy” a third one if you run out of slip days (with an additional penalization of -3 points)
The CSE 8B central skills

- Write
  - Programs/algorithms
  - Using loops, arrays, conditionals, variables objects, subclasses, GUIs, recursion, etc (see review sheet)
- Trace/debug
  - Draw memory models
  - Given some code, what does it do?
  - Find the errors and fix them
- Explain
  - Why things go wrong
  - How things work/why code functions the way it does
  - Why the answer to a question is right or wrong

You will be given reference material. Memorize concepts not details.
Write a method that takes an array of ints and returns a new array of ints that contains only a single instance of each of the numbers in the input array.

Let’s assume, to start, that you have the following helper method:

```java
public static int countUnique( int[] a )
```

Which will return the number of unique elements in a.

**Hint:** Write a helper method that will check whether an array contains a given element.

```java
public class ArrayPlay
    public static int[] removeDuplicates( int[] arr )
{
```
Write: code that solves an array problem

Write a method that takes an array of ints and returns a new array of ints that contains only a single instance of each of the numbers in the input array.

Let’s assume, to start, that you have the following helper method:

```java
public int countUnique( int[] a )
```

Which will return the number of unique elements in a.

**Hint:** write a helper method that will check whether an array contains a given element.

**FIRST: make the plan, i.e. your “pseudo-code” or algorithm steps**

```java
public class ArrayPlay

public static int[] removeDuplicates( int[] arr )
{
    // make a new array a ONLY with the amount of unique elements

    // For each element in arr
    1 - check if it is contained in my new array a
    2 - if it’s not -> add it
    3 - if it is -> continue with the loop without changing a

}
Write: code that solves an array problem

- Write a method that takes an array of ints and returns a *new* array of ints that contains only a single instance of each of the numbers in the input array.
- Let’s assume, to start, that you have the following helper method:

```java
public static int countUnique( int[] a )
```

Which will return the number of unique elements in a.

- Hint: write a helper method that will check whether an array contains a given element.

```java
public class ArrayPlay
    public static int[] removeDuplicates( int[] arr )
    {
        // make a new array ONLY with the amount of unique elements
        int[] myArray = new int[countUnique( arr )];
        int retInd = 0;
        for ( int i = 0; i < arr.length; i++ ){
            if ( !contains( myArray, arr[i] ) ) {
                myArray[retInd] = arr[i];
                retInd ++;
            }
        }
        return myArray;
    }

    public static int countUnique( int[] a )
    {
        // TO-DO: CAN YOU WRITE A METHOD USING ArrayList myArray
        // instead of int[] myArray?
```

*This is the helper method I need to write*
Write: code that solves an array problem

Hint: consider a helper method that will check whether an array contains a given element.

```java
public static boolean contains ( int[] a, int num){
    // THIS IS THE LOOP VERSION OF CONTAINS
    for (int i=0; i<a.length; i++){
        if (a[i] == num)
            return true;
    }
    return false;
}

// TO-DO: CAN YOU WRITE A RECURSIVE VERSION OF THE SAME THING!??
public static boolean contains ( int[] a, int num, . . . ){
```
Write: code that solves an array problem

Hint: write a helper method that will check whether an array contains a given element.

```java
Public static boolean contains ( int[] a, int num){
   // THIS IS THE LOOP VERSION
   for (int i=0; i<a.length; i++){
      if (a[i] == num)
         return true;
   }
   return false;
}

// A RECURSIVE VERSION OF THE SAME THING
Public static boolean contains ( int[] a, int num, int i ){
   if (i>=a.length)
      return false;
   else{
      return (a[i]==num || contains(a, num, i+1));
   }
}

This is a boolean expression so I don’t need more than this line!
Remember that this method needs to return a boolean value (true or false), a boolean variable or a boolean expression. ANYTHING that “after all” is giving me a boolean
What is the value of myArray in main at the end of main
(assume contains works correctly as in the previous example)?

public class ArrayPlay

    public int[] removeDuplicates( int[] arr )
    {
        int[] myArray = new int[countUnique( arr )];
        int retInd = 0;
        for ( int i = 0; i < arr.length; i++ )
        {
            if ( !contains( myArray, arr[i] ) ) {
                myArray[retInd] = arr[i];
                arr[i] = 0;
                retInd ++;
            }
        }
        return myArray;
    }

    public static void main( String[] args )
    {
        ArrayPlay ap = new ArrayPlay();
        int[] myArray = {1, 1, 2, 5, 2, 2, 3};
        int[] myArrayNoDups = ap.removeDuplicates( myArray );
    }

Value of myArray in MAIN:
A. [1, 1, 2, 5, 2, 2, 3]
B. [1, 2, 5, 3]
C. [0, 1, 0, 0, 0, 2, 0]
D. [0, 1, 0, 0, 2, 2, 0]
E. [1, 0, 2, 5, 0, 0, 3]

THIS CODE is NOT exactly the same as previous example. Besides the new line (in red) now removeDuplicates is NON-STATIC, so we need an object from the corresponding class (ArrayPlay) JUST to be able to call the method. Note that the calling object ap HAS NOTHING to “do” inside the code in this case.
What is the value of myArray in main at the end of main (assume contains works correctly as in the previous example)?

public class ArrayPlay

    public int[] removeDuplicates( int[] arr )
    {
        int[] myArray = new int[countUnique( arr )];
        int retInd = 0;
        for ( int i = 0; i < arr.length; i++ )
        {
            if ( !contains( myArray, arr[i] ) ) {
                myArray[retInd] = arr[i];
                arr[i] = 0;
                retInd ++;
            }
        }
        return myArray;
    }

    public static void main( String[] args )
    {
        ArrayPlay ap = new ArrayPlay();
        int[] myArray = {1, 1, 2, 5, 2, 2, 3};
        int[] myArrayNoDups = ap.removeDuplicates( myArray );
    }

An ArrayPlay object that I’m not using right now for ANYTHING but calling a method…
public class ArrayPlay {
    public int[] removeDuplicates( int[] arr ) {
        int[] myArray = new int[countUnique( arr )];
        int retInd = 0;
        for ( int i = 0; i < arr.length; i++ ) {
            if ( !contains( myArray, arr[i] ) ) {
                myArray[retInd] = arr[i];
                arr[i] = 0;
                retInd ++;
            }
        }
        return myArray;
    }
    public static void main( String[] args ) {
        ArrayPlay ap = new ArrayPlay();
        int[] myArray = {1, 1, 2, 5, 2, 2, 3};
        int[] myArrayNoDups = ap.removeDuplicates( myArray );
    }
}

Value of myArrayNoDups:
A. [1, 2, 5, 3]
B. [1, 0, 0, 0]
C. [3, 0, 0, 0]
D. [3, 5, 2, 1]
Trace/debug: interpreting Java errors

In what method that you wrote does the error occur?

A. main
B. sum
C. main and sum
D. neither

java.lang.ArrayIndexOutOfBoundsException: 5
    at ArrayPlayReview.sum(ArrayPlayReview.java:7)
    at ArrayPlayReview.main(ArrayPlayReview.java:15)
    at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
    at sun.reflect.NativeMethodAccessorImpl.invoke(Unknown Source)
    at sun.reflect.DelegatingMethodAccessorImpl.invoke(Unknown Source)
    at java.lang.reflect.Method.invoke(Unknown Source)
Trace/debug: interpreting Java errors

What line is the source of the error (i.e. where you should go to fix it?)

A. 5  
B. 7  
C. 15  
D. You cannot tell without looking at the code  
E. Other  

java.lang.ArrayIndexOutOfBoundsException: 5
  at ArrayPlayReview.sum(ArrayPlayReview.java:7)
  at ArrayPlayReview.main(ArrayPlayReview.java:15)
  at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
  at sun.reflect.NativeMethodAccessorImpl.invoke(Unknown Source)
  at sun.reflect.DelegatingMethodAccessorImpl.invoke(Unknown Source)
  at java.lang.reflect.Method.invoke(Unknown Source)
public class ArrayPlay {
    public int sum( int[] arr ) {
        int total = 0;
        for ( int i = 0; i <= arr.length; i++ ) {
            total += arr[i];
        }
        return total;
    }
    public static void main( String[] args ) {
        ArrayPlay ap = new ArrayPlay();
        int[] myArray = {1, 1, 2, 5, 2};
        System.out.println( ap.sum( myArray ) );
    }
}

java.lang.ArrayIndexOutOfBoundsException: 5
    at ArrayPlayReview.sum(ArrayPlayReview.java:7)
    at ArrayPlayReview.main(ArrayPlayReview.java:15)
    at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
    at sun.reflect.NativeMethodAccessorImpl.invoke(Unknown Source)
    at sun.reflect.DelegatingMethodAccessorImpl.invoke(Unknown Source)
    at java.lang.reflect.Method.invoke(Unknown Source)
public class ArrayPlay {
    public int sum( int[] arr )
    {
        int total = 0;
        for ( int i = 0; i < arr.length; i++ )
        {
            total += arr[i];
        }
        return total;
    }

    public static void main( String[] args )
    {
        ArrayPlay ap = new ArrayPlay();
        int[] myArray = {1, 1, 2, 5, 2};
        System.out.println( ap.sum( myArray ) );
    }
}

java.lang.ArrayIndexOutOfBoundsException: 5
    at ArrayPlayReview.sum(ArrayPlayReview.java:7)
    at ArrayPlayReview.main(ArrayPlayReview.java:15)
    at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
    at sun.reflect.NativeMethodAccessorImpl.invoke(Unknown Source)
    at sun.reflect.DelegatingMethodAccessorImpl.invoke(Unknown Source)
    at java.lang.reflect.Method.invoke(Unknown Source)
public class MorePeople {
    public static void main( String[] args )
    {
        Person p = new Person( "Sally" );
        Person s = new Student( "Sam" );
        Student t = new Person( "John" );

        s.setName( "Steve" );
        System.out.println( p.getName() );
        System.out.println( s.getName() );
        System.out.println( t.getName() );
    }
}

class Person {
    private String name;
    public Person( String n ) { name = n; }
    public String getName() { return this.name; }
}

class Student extends Person
{
    public Person( String n ) { name = n; }
    public void setName( String newName )
    { name = newName; }
}
public class MorePeople {
    public static void main( String[] args )
    {
        Person p = new Person( "Sally" );
        Person s = new Student( "Sam" );
        Student t = new Person( "John" );
        s.setName( "Steve" );
        System.out.println( p.getName() );
        System.out.println( s.getName() );
        System.out.println( t.getName() );
    }
}

class Person {
    private String name;
    public Person( String n ) { name = n; }
    public String getName() { return this.name; }
}

class Student extends Person
{
    public Person( String n ) { name = n; }
    public void setName( String newName )
    { name = newName; }
}

A person object is not “enough” to be a student → compiler error!
We could fix it writing:
Student t = new Student(“John”);

The compiler only sees the declared type of s (Person), and Person class does NOT have a setName method → compiler error!
We could fix it writing:
((Student) s).setName(“Steve”);

Identify and explain ALL of the errors in the code on the left.
Which type of error?
A. Runtime
B. Compilation time
Write and explain: Complete the recursive step

Write a method to find an element in an array and return its index. startIndex starts at 0.

```java
public static int find( String[] myList, String toFind, int startIndex )
{
    if ( startIndex >= myList.length ) {
        return -1;
    }
    if ( toFind.equals( myList[startIndex] ) ) {
        return startIndex;
    }
    // what goes here _______
}
```

A. return find( myList, toFind, startIndex + 1 );
B. return find( myList, toFind, startIndex - 1 );
C. return startIndex + 1;
D. find( myList, toFind, startIndex + 1 );

Explain why each answer is wrong or right!
Worksheet: Complete the recursive step

A. return find( myList, toFind, startIndex + 1 );

B. return find( myList, toFind, startIndex - 1 );

C. return startIndex + 1;

D. find( myList, toFind, startIndex + 1 );
Write and explain: Complete the recursive step

A. `return find( myList, toFind, startIndex + 1 );`
   This is the correct answer. It will reduce the problem by increasing `currIndex` until it hits the base case. It will also return the value of the recursive call.

B. `return find( myList, toFind, startIndex - 1 );`
   This answer “changes” the problem, but does so in a way that takes it away from the base case, instead of towards it.

C. `return startIndex + 1;`
   This answer does not make a recursive call. So the method will fail to look in the rest of the list for the element.

D. `find( myList, toFind, startIndex + 1 );`
   This answer makes the correct recursive call, but does not return its result. We must not only make the recursive call, but also return its result.

E. None of these. Wrong because A is correct.
Write and explain: Complete the recursive step

Write a method to find an element in an array and return its index. startIndex starts at 0.

```java
public static int find( String[] myList, String toFind, int startIndex )
{
    if ( startIndex >= myList.length ) {
        return -1;
    }
    if ( toFind.equals( myList[startIndex] ) ) {
        return startIndex;
    }
    return find( myList, toFind, startIndex + 1 );
}
```

What is the **maximum** number of elements in the list that find might have to consider, taking into account the *total* number across all recursive calls?

A. 1  
B. Half the elements in the list  
C. All of the elements in the list
Write and explain: Complete the recursive step

Write a method to find an element in an array and return its index. startIndex starts at 0

```java
public static int find( String[] myList, String toFind, int startIndex )
{
    if ( startIndex >= myList.length ) {
        return -1;
    }
    if ( toFind.equals( myList[startIndex] ) ) {
        return startIndex;
    }
    return find( myList, toFind, startIndex + 1 );
}
```

What is the maximum number of elements in the list that find might have to consider, taking into account the total number across all recursive calls?
A. 1
B. Half the elements in the list
C. All of the elements in the list
The CSE 8B central skills

• Write
  • Programs/algorithms
  • Using loops, arrays, conditionals, variables objects, subclasses, GUIs, recursion, etc (see review sheet)

• Trace/debug
  • Draw memory models
  • Given some code, what does it do?
  • Find the errors and fix them

• Explain
  • Why things go wrong
  • How things work/why code functions the way it does
  • Why the answer to a question is right or wrong

You will be given reference material. Memorize concepts not details.
Thanks!

It was a great quarter

And you all a great class 😊

GOOD LUCK WITH FINALS!