CSE 8B Today

It’ll get harder before it gets easier...

Objects, classes, and (yes) MORE memory models!

Getting ready for PSA4 next week ...
Exam 1

- Median: 7.9
- 28 perfect scores!
- Handing them back at the END of THIS CLASS

If you got >8, nice work! You’re doing very well. Make sure you understand small points you missed (if any).

If you got between a 6 and an 8, you’re getting it and doing fine, but try to come to office hours (with me or a TA) to make sure you understand everything you got wrong if you still have doubts after the class review.

If you got between 4 and 6, come to office hours (with me or a TA) to make sure you understand all the mistakes.

If you scored lower than a 4, come talk to Ana about ways to improve your studying and understanding or concepts you may need to catch up.
Re-grade Requests (deadline next Thursday)

As per the syllabus:

You have **one week** from the time a PSA or Exam is returned to **request** a regrade. After that, the grade is set in stone. To request a regrade, please contact the person who graded the assignment/quiz/exam originally. **If you are requesting an exam regrade, write up a note explaining the issue and submit the exam with the note stapled to the front to Ana (in the class or bring it to her office hours)**
1. We always need an object of the corresponding class to call a method, except for certain methods which are called from the class name, these methods are. . .

A. Main method
B. Instance method
C. Static method
D. Constructor
E. None of the above

These are called automatically at different situations

We need an “instance” (an object variable) to call instance methods
2. Suppose you create a class called Car:
   
   ```java
   class Car {
       String model;
       int year;
       String color;
   }
   ```

   What is the default value of `String model` (when we make a new instance of this class)?

   A. 0
   B. null
   C. "" (the empty string)
   D. This would result in a compile error.
3. Given the following class, what is printed out when p() is called?

```java
public class Foo {
    int x;
    public Foo() {
        x = 10;
    }
    public void p() {
        int x = 25;
        System.out.println("x = " + x);
    }
}
```

A. x = 0
B. x = 25
C. x = 35
D. Nothing, this results in a compile error.
4. We have a Class MyClass which has:

- all its instance variables (fields) public
- no *mutator* methods
- and no *accessor* method that returns a reference to any field.

Is Class MyClass immutable? Why or why not?

A. Yes, it is immutable because there are no mutator methods.

B. Yes, it is immutable because there are no accessor methods.

C. No, it is not immutable because Java does not allow.

D. No, it is not immutable because its fields are public.
Changing the value of data in memory

```java
public static int allNumsAreOdd( int[] myArray) {
    int i =0;
    for ( i = 0; i < myArray.length; i++ ) {
        if ( myArray[i] %2 == 1 ) {
            System.out.println( "Odd number" );
        } else {
            System.out.println( "Found an even number!" );
            return -i;
        } // end of else statement
    } // end of for loop
    return i;
} // end of the method

int[] myA = {5, 4, 3, 1};
int result = ArrayPlay.allNumsAreOdd( myA );
```

What is the value of the variable result after
the last line above? ________________

A. True
B. 1
C. -1
D. 4
public static int allNumsAreOdd( int[] myArray) {
    int i =0;
    for ( i = 0; i < myArray.length; i++ ) {
        if ( myArray[i] %2 == 1 ) {
            // this condition is true when the number is odd
            System.out.println( "Odd number" );
        }
        else {
            System.out.println( "Found an even number!" );
            return -i;
        } // end of else statement
    } // end of for loop
    return i;
} // end of the method

> int[] myA = {5, 4, 3, 1};
> int result = ArrayPlay.allNumsAreOdd( myA );

What is printed in the terminal by the call to allNumsAreOdd?

A. Only the first line
B. The two first lines
C. All the lines
D. None of them

HINTS!
RETURN VALUE! And then . . .
STATIC METHODS! are defined the SAME way (except using static keyword)
### Variables, types and memory (again)

#### Primitive Types
(booleans, byte, char, int, short, long, float, double)
- Occupy a pre-determined amount of space
- You can think of their values as being stored directly in a variable

```java
int x = 5;
char c = 'A';
```

#### Objects
(All arrays and class instances)
- Occupy a variable amount of space
- The “value” of an object is stored in memory (on the heap) and the reference to the location where the object is located in memory is stored in the variable

```java
int[] xA = {5, 5};
String s = "A";
```

### Memory (“the heap”)

<table>
<thead>
<tr>
<th>xA</th>
<th>600</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>5</td>
</tr>
<tr>
<td>c</td>
<td>‘A’</td>
</tr>
</tbody>
</table>

Addresses in memory

600  5  5
...
712  “A”
Variables, types and memory (again)

Primitive Types
(booleans, byte, char, int, short, long, float, double)
- Occupy a pre-determined amount of space
- You can think of their values as being stored directly in a variable

```
int x = 5;
char c = 'A';
```

Objects
(All arrays and class instances)
- Occupy a variable amount of space
- The “value” of an object is stored in memory (on the heap) and the reference to the location where the object is located in memory is stored in the variable

```
int[] xA = {5, 5};
String s = "A";
```

Memory ("the heap")

```
xA
  5 5
 s
  "A"
```
Assignment operator with variables

When a (simple) variable name (i.e., *NOT an index into an array like xA[1]*) appears on the left hand side (LHS) of an assignment operator (=), Java will first evaluate the expression on the right hand side (RHS) of the operator, and then place that value into the variable on the LHS. The value of any variable, including reference variables, is just the data stored *in its box*.

```java
int x = 5;
char c = 'A';
int y = 6;
x = y;
```

```
int[] xA = {5, 5};
String s = “A”;
int[] yA = {6, 6};
xA = yA;
```

In the diagrams we’ll replace “addresses” with “arrows” (see next slide)  

Memory ("the heap")

```
600  5  5
...
712  “A”
1024  6  6
```
Assignment operator with variables

When a (simple) variable name (i.e., NOT an index into an array like xA[1]) appears on the left hand side (LHS) of an assignment operator (=), Java will first evaluate the expression on the right hand side (RHS) of the operator, and then place that value into the variable on the LHS. The value of any variable, including reference variables, is just the data stored in its box.
Changing the value of data in memory

public void doStuff( int[] myArray ) {
    int[] arr = new int[myArray.length];
    for ( int i = 0; i < myArray.length; i++ ) {
        arr[i] = myArray[i] * 2 ;
    }
    myArray[0] = -1;
}

> ArrayPlay a = new ArrayPlay();
> int[] myA = {1, 2, 3};
> a.doStuff( myA );
Changing the value of data in memory

```java
public void doStuff(int[] myArray) {
    int[] arr = new int[myArray.length];
    for (int i = 0; i < myArray.length; i++) {
        arr[i] = myArray[i] * 2;
    }
    myArray[0] = -1;
}
```

> ArrayPlay a = new ArrayPlay();
> int[] myA = {1, 2, 3};
> a.doStuff(myA);

**Memory ("the heap")**

```
ArrayPlay object scope
```

```
1  2  3
```

```
2  4  6
```

**DoStuff scope**

```
myArray
arr
...```

**MID-TERM 1 PROBLEM 2**
public static int allNumsAreOdd( int[] myArray) {
    int i =0;
    for ( i = 0; i < myArray.length; i++ ) {
        if ( myArray[i] %2 == 1 ) {
            // this condition is true when the number is odd
            System.out.println( "Odd number" );
        } else {
            System.out.println( "Found an even number!" );
            return -i;
        } // end of else statement
    } // end of for loop
    return i;
} // end of the method

> int[] myA = {5, 4, 3, 1};
> int result = ArrayPlay.allNumsAreOdd( myA );
Finish the code in the following method named `onlyLettersUp` that takes a `String s` as input and returns a new `String` object where all letters in `s` have Upper case. i.e., uppercase letters stay the same, while lower case letters have become upper case letters. Non-alphabetic characters in the string are replaced with a dot character (.)

Here are a couple examples. If you are at all unsure about how this method should work, please ask.

- `onlyLettersUp( "You Got It!!" )` returns the string "YOU.GOT.IT..
- `onlyLettersUp( "U_C_S_D.2014" )` returns the string "U.C.S.D....."

You need to USE a `char[] chars` to access the elements from the `String` at some point.

```java
public String onlyLettersUp ( String s ) {
    char[] chars = . . .
```
The game of Nim

4 Rows of matches

On your turn, you can take as many matches as you want from a single row

The winner is the player who takes the last match

See Nim.java file with the sample code (class website)
Representing Nim as a Java Class

```java
public class Nim {
    /** The current state of the nim board.
     * One entry for each row.
     * The numbers in the board specify how many sticks
     * are left in each row. */
    private int[] board;

    /** Whose turn it currently is. */
    private int turn;

    // . . . More stuff you may think to config. the game
}
```
public class Nim {
    /** The current state of the nim board.
     * One entry for each row.
     * The numbers in the board specify how many sticks
     * are left in each row. */
    private int[] board;  
    For simplicity, no “turn” variable.
The Nim constructor

Complete the nim constructor so that it initializes the board as shown here and initializes turn to player 1

```java
public class Nim {
    private int[] board;

    public Nim() {
        board = new int[4];
        board[0] = 7;
        board[1] = 5;
        board[2] = 3;
        board[3] = 1;
    }
}
```
The Nim constructor

Complete the nim constructor so that it initializes the board as shown here and initializes turn to player 1

```java
public class Nim {
    private int[] board;

    public Nim() {
        this.board = new int[4];
        this.board[0] = 7;
        this.board[1] = 5;
        this.board[2] = 3;
        this.board[3] = 1;
    }
}
```

this means “the calling object” (or here, the object being created). The use of this is optional (usually) but HIGHLY encouraged so you are 100% sure which variable you mean.
In main inside the Nim class...

```java
public class Nim {
    ...
    public static void main(String[] args) {
        Nim game1 = new Nim();
    }
}
```

Draw the memory model for the above code
public class Nim {

    public static void main(String[] args) {
        Nim game1 = new Nim();
    }
}
complete the MM to answer the question

```java
public class Nim {
    ...
    public static void main( String[] args )
    {
        Nim game1 = new Nim();
        Nim game2 = new Nim();
        game1.board[2] = 100;
        game1 = game2;
        System.out.println( game1.board[2] );
    }
}
```

What is printed by the above code? (Hint: Extend the above memory model!)

A. 3  
B. 100  
C. Nothing, there is a compile error  
D. Nothing, there is an array out of bounds exception  
E. I don’t know

OK!! even though board is private, because we’re still in the Nim class.

See it in next slide
public class Nim {
    ...
    public static void main( String[] args )
    {
        Nim game1 = new Nim();
        Nim game2 = new Nim();
        game1.board[2] = 100;
        game1 = game2;
        System.out.println( game1.board[2] );
    }
}

Row 0  Row 1  Row 2  Row 3

complete the MM to answer the question
Printing objects

// Method in the Nim Class
public String toString()
{
    System.out.println( "Printing a Nim board" );
}

// In main in Nim.java

Nim n = new Nim();
System.out.println( n );

What is printed by the this code?
A. Printing a Nim board
B. "" (An empty string)
C. Nothing, there is a compile error
D. It prints the reference to the object in memory
A toString method DOES NOT PRINT
(It is called automatically when the object is printed)

// method in the Nim class
public String toString()
{
    String toReturn = "Printing a Nim board";
    return toReturn;
}

// In main in Nim.java

Nim n = new Nim();
System.out.println( n );
// The last line is the same as
// System.out.println( n.toString() );
Complete the toString method for Nim

```java
public String toString()
{
    String toReturn = new String();
    for (int i = this.board.length - 1; i >= 0; i--)
    {
        toReturn += "Row " + i + ": " ;

        for ( __________________________________________ )
        {
            toReturn += "| ";
        }
        toReturn += ( _______________ );
    }
    return toReturn;
}
```

**HINTS:**

1) `System.out.print(...)` → Prints text without a newline

2) **Appending** "\n" (a string containing the newline character) to a string will force a newline in that place when the string is printed
Complete the toString method for Nim

```java
public String toString()
{
    String toReturn = new String();
    for (int i = this.board.length-1; i >= 0; i-- )
    {
        toReturn += "Row " + i + ": " ;

        for (int sticks = 0; sticks < this.board[i]; sticks++)
        {
            toReturn += "| ";
        }
        toReturn += ( "\n" );
    }
    return toReturn;
}
```

Row 3: |
Row 2: | | |
Row 1: | | | | |
Row 0: | | | | | | | |

Hint: Appending “\n” (a string containing the newline character) to a string will force a newline in that place when the string is printed.
Reading input from the console

You can use the Scanner object (the same one we used to read from files) to read input from the console. When you request input from the Scanner, it will pause the program until the user enters something.

```java
Scanner input = new Scanner(System.in);
System.out.println( this ); // print the board
System.out.println( "Player " + currPlayer + "'s turn." );
System.out.print( "Which row? " );
int row = input.nextInt(); // pause and wait until the user
// enters a row. Read what the
// user types as an int, and
// return it, so that the
// variable row gets assigned
// the int value entered.
System.out.print( "How many sticks? " );
int numSticks = input.nextInt();
```

This code is in Nim’s `hostGame` method
Reading input from the console

```java
Scanner input = new Scanner(System.in);
System.out.println( this );  // print the board
System.out.println( "Player " + currPlayer + "'s turn." );
System.out.print( "Which row? " );
ing row = input.nextInt();  // pause and wait until the user
// enters a row. Read what the
// user types as an int, and
// return it, so that the
// variable row gets assigned
// the int value entered.
System.out.print( "How many sticks? " );
ing numSticks = input.nextInt();
```

**The class Scanner is found in the java.util package.** What line must be included at the top of the file to use the Scanner as in this example? (there may be more than one right answer.)

A. import java.*;
B. import java.util.*;
C. import java.util.Scanner;
D. You don’t need to include any special line. Java automatically includes everything from the util package

Nothing *happens* until we try to “get something” from input
Scanner input = new Scanner(System.in);
System.out.println( this ); // print the board
System.out.println( "Player " + currPlayer + "'s turn." );
int row = -1;
int numSticks = -1;

while ( ______________________________________ ) {
    System.out.print( "Which row? " );
    row = input.nextInt();
    System.out.print( "How many sticks? " );
    numSticks = input.nextInt();
}

Which line correctly completes the code above so that the game will re-prompt the user until they enter a legal row and number of sticks?

HINTS:
isEmpty tests to see if the whole board is empty.
allowsMove tests to see if a given number of sticks can be taken from a given row.
! means “not”

A. !this.isEmpty()
B. !this.allowsMove( row, numSticks )
C. numSticks <= 0
D. row <= 0
E. numSticks <= 0 || row <= 0
Check out Nim.java as an example before you start PSA4