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

Getting ready for PSA4 next week ...
Variables, types and memory (again)

**Primitive Types**
(boolean, 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”)**

Addresses in memory
Variables, types and memory (again)

**Primitive Types**
(boolean, 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` is stored at memory location `5` with value `5`.
- `s` is stored at memory location `"A"`.
- The string "A" is stored somewhere on the heap, and the reference to its location is stored in the variable `s`.
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 );
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
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
```
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;                // 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
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.
public class Nim {
    ...
    public static void main( String[] args )
    {
        Nim game1 = new Nim();
    }
}

Draw the memory model for the above code.
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
// 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
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;
}
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.

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

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();

Nothing happens until we try to “get something” from input

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
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