Re-grade Requests (deadline next Wednesday)

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 your instructor.
CSE 8B Today

Why objects?
Why memory models?

Debugging strategies...

PSA4: Posted and ready to go!

Nim.java (posted on the website)
A VERY GOOD place to start for PSA4
1. Class A has all of its data fields public, no mutator methods, and no accessor method that returns a reference to a data field that is mutable. Is Class A 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 its fields are public.

D. No, it is not immutable because Java does not allow programmers to create immutable classes.
2. When does the order of data field initialization matter?

A. When a data field is initialized based on a reference to another data field.

B. When a data field is initialized based on a reference to a local variable.

C. They should always be initialized in alphabetical order.

D. Never.
3. Given the following class, what is printed out when `p()` is called?

```java
public class Foo {
    int x = 3;
    public void p() {
        int x = 25;
        System.out.println("x = " + x);
    }
}
```

A. x = 25
B. x = 3
C. x = 0
D. Nothing, this results in a compile error.
public void displayWords()
{
    JFrame wordFrame = new JFrame("Word cloud");
    wordFrame.setLayout(new FlowLayout(FlowLayout.LEFT));
    int fontBase = 10;
    int maxCount = getMaxCount();
    int baseScale = 3;
    for ( int i = 0; i < uniqueWords.length; i++ ){
        int fontScale = (int)(fontBase *
            (1 + baseScale*counts[i]/(double)maxCount));
        Font currFont = new Font("SansSerif", Font.PLAIN,
            fontScale+counts[i] );
        JLabel wordLabel = new JLabel( uniqueWords[i] );
        wordLabel.setFont( currFont );
        wordFrame.add( wordLabel );
    }

    wordFrame.pack();
    wordFrame.setVisible(true);
}
public void displayWords()
{
    JFrame wordFrame = new JFrame("Word cloud");
    wordFrame.setLayout(new FlowLayout(FlowLayout.LEFT));
    int fontBase = 10;
    int maxCount = getMaxCount();
    int baseScale = 3;
    for (int i = 0; i < uniqueWords.length; i++)
    {
        int fontScale = (int)(fontBase * (1 + baseScale * counts[i] / (double) maxCount));
        Font currFont = new Font("SansSerif", Font.PLAIN, fontScale+counts[i]);
        JLabel wordLabel = new JLabel(uniqueWords[i]);
        wordLabel.setFont(currFont);
        wordFrame.add(wordLabel);
    }
    wordFrame.pack();
    wordFrame.setVisible(true);
}

KNOW WHY you are making changes to your code
THINK ABOUT and use strategies to ISOLATE the problems in your code
Always get one small part working BEFORE moving on to the next.
Is debugging frustrating? YES! But there are ways to make it less frustrating.
A slight modification to our Nim class

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

We want to build a Java class to “host” the game of Nim.
What data should we store in a class?
A. The number of sticks in each pile [We will go with this for now]
B. The names of the two players
C. Which player’s turn it is
D. A&C
E. All of the above

ANY ANSWER COULD BE CORRECT.
IT’S UP TO YOU, THE CLASS DESIGNER.
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;
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.
public class Nim {
    ...
    public static void main( String[] args )
    {
        Nim game1 = new Nim();
    }
}
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: Draw the memory model! Worksheet!)
A. 3 (see next slide)
B. 100
C. Nothing, there is a compile error
D. Nothing, there is an array out of bounds exception
E. I don’t know
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
In main inside the Nim class...

```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] );
    }
}
```

OK even though board is private because we’re still in the Nim class.
public class Nim {
    private int[] board;
    public Nim() {
        int[] board = new int[4];
        board[0] = 7;
        board[1] = 5;
        board[2] = 3;
        board[3] = 1;
    }

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

The **scope of a variable** is the part of a program where it can be accessed:

- Variables declared inside methods are called **local variables**. They exist ONLY inside the method they were declared. **Parameters** are a special type of local variable.
- Variables declared outside methods (inside classes) are **member variables** or **instance variables**. They exist throughout the whole class definition (in all class methods).
- When local variables have the same name as member variables, they “take over” (they are said to “shadow” member variables). Here the local variable board shadows the member variable board.
public class Nim {
    private int[] board;
    public Nim() {
        int[] board = new int[4];
        board[0] = 7;
        board[1] = 5;
        board[2] = 3;
        board[3] = 1;
    }

    public static void main(String[] args) {
        Nim game1 = new Nim();
        game1.board[2] = 100;
    }
}

Nim object scope
- board: null

Nim constructor scope
- board: [7, 5, 3, 1]

main scope
- game1

Null Pointer exception!
public class Nim {
    private int[] board;
    public Nim() {
        int[] board = new int[4];
        this.board[0] = 7;
        this.board[1] = 5;
        this.board[2] = 3;
        this.board[3] = 1;
    }

    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] );
    }
}
public void printBoard()
{
    for (int i = this.board.length-1; i >= 0; i--)
    {
        System.out.print("Row \ " + i + ": ");
        for (int j = 0; j < this.board[i]; j++)
            System.out.print("| ");
        System.out.print("\n");
    }
}

public String toString()
{
    String toReturn = new String();
    for (int i = this.board.length-1; i >= 0; i--)
    {
        toReturn += "Row \ " + i + ": ";
        for (int j = 0; j < this.board[i]; j++)
            toReturn += "| ";
        toReturn += ("\n");
    }
    return toReturn;
}
Check out Nim.java as an example as you start PSA4
Start PSA4 NOW. REALLY.