PSA 3 due tomorrow

CSE 8B Today

PSA4: Coming soon to a website near you

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

Exams: handed back at the end of class

Options for exam review: Discussion Wednesday and Friday, office hours this week and next. See Piazza for details

A note about extra credit...
1. What defines the *state* of an Object (or in other words, the properties that characterize an Object)?

A. Its constructor.

B. Its methods.

C. Its comments.

D. Its data fields.

D. Its data fields.
2. What special kind of method is invoked to create a new Object?

A. Constructor  
B. Main method  
C. Static method  
D. Instance method  
E. None of the above
3. Suppose you create a class called Car:

```java
class Car {
    String model;
    int year;
    String color;
}
```

What is the default value of String model?

A. null

B. "" (the empty string)

C. 0

D. This would result in a compile error.
Introduce yourself to your new group

Pepper Canyon Hall 109 - CSE 8B - Spring 2013

Peer Instruction Layout:

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>L</td>
<td>K</td>
<td>J</td>
<td>H</td>
<td>G</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
</tbody>
</table>

Initial groups: Weeks 1-2
Second groups: Weeks 3-6
Final groups: Weeks 7-10

Initial groups:
- L: Week 1
- K: Week 2
- J: Week 3
- H: Week 4
- G: Week 5
- F: Week 6
- E: Week 7
- D: Week 8
- C: Week 9
- B: Week 10

Second groups:
- L: Week 3
- K: Week 4
- J: Week 5
- H: Week 6
- G: Week 7
- F: Week 8
- E: Week 9
- D: Week 10
- C: Week 1
- B: Week 2

Final groups:
- L: Week 7
- K: Week 8
- J: Week 9
- H: Week 10
- G: Week 1
- F: Week 2
- E: Week 3
- D: Week 4
- C: Week 5
- B: Week 6
- A: Week 7
- 5: Week 8
- 10: Week 9
- 11: Week 10
- 12: Week 1
- 13: Week 2
- 14: Week 3
- 15: Week 4
- 16: Week 5
- 17: Week 6
- 18: Week 7
- 19: Week 8
- 20: Week 9
private char rc( char x, int y )
{
    if (!Character.isLetter(x)) return x;
    y = y % 26;
    if ( y < 0 )
        y += 26;
    char z = (char)(x + y);
    if ( Character.isUpperCase(x) && z > 'Z' )
        z = (char)(z - 26);
        if ( Character.isLowerCase(x) && z > 'z' ) z =
            (char)(z - 26);
    return z;
}

What does this method do??
A. I have no idea
B. I think I might know
C. I am (pretty) sure I know
A note about style...

private char rotateChar( char c, int rotation )
{
    if ( !Character.isLetter( c ) )
        return c;

    rotation = rotation % 26;
    if ( rotation < 0 )
        rotation += 26;

    char charToReturn = (char)(c + rotation);

    if ( Character.isUpperCase(c) && charToReturn > 'Z' )
        charToReturn = (char)(charToReturn - 26);

    if ( Character.isLowerCase(c) && charToReturn > 'z' )
        charToReturn = (char)(charToReturn - 26);

    return charToReturn;
}
private char rotateChar(char c, int rotation) {
    if (!Character.isLetter(c))
        return c;

    rotation = rotation % 26;
    if (rotation < 0)
        rotation += 26;

    char charToReturn = (char)(c + rotation);

    if (Character.isUpperCase(c) && charToReturn > 'Z')
        charToReturn = (char)(charToReturn - 26);

    if (Character.isLowerCase(c) && charToReturn > 'z')
        charToReturn = (char)(charToReturn - 26);

    return charToReturn;
}
private char rotateChar( char c, int rotation ) {
    if (!Character.isLetter( c )) {
        return c;
    }
    rotation = rotation % 26;
    if ( rotation < 0 ) {
        rotation += 26;
    }
    char charToReturn = (char)(c + rotation);
    if ( Character.isUpperCase( c ) && charToReturn > 'Z' ) {
        charToReturn = (char)(charToReturn - 26);
    }
    if ( Character.isLowerCase( c ) && charToReturn > 'z' ) {
        charToReturn = (char)(charToReturn - 26);
    }
    return charToReturn;
}

if ( Character.isLowerCase(c) && charToReturn > 'z' ) {
    charToReturn = (char)(charToReturn - 26);
}

return charToReturn;  

Short, targeted method
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”)

<table>
<thead>
<tr>
<th>Addresses in memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>xA 600</td>
</tr>
<tr>
<td>x 5</td>
</tr>
<tr>
<td>s 712</td>
</tr>
<tr>
<td>600</td>
</tr>
<tr>
<td>712</td>
</tr>
<tr>
<td>&quot;A&quot;</td>
</tr>
</tbody>
</table>

...
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”)*

- xA references the array at memory location 5, 5
- s references the string “A” at memory location ...

---

---

---

---
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.
There are two ways to directly change what is stored in memory (i.e. not simply what’s stored “in the box”).

1. An indexed location in an array appears on the LHS of an assignment operator
2. You use the “dot” notation to modify an object’s properties (we’ll see this later)

Note that some methods (like Arrays.sort()) will also modify objects in memory, but this is because they use either (1) or (2) to do so.
From arrays to classes

Properties of an object are determined by its *instance variables*

- What types are they? What values do they have?
- (also known as ‘fields’ or ‘member variables’)

Behaviors of an object are determined by its *instance methods*

- What parameters do they take? What values do they return (if any)? What do they do?
- *Constructors* are a special kind of method...
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

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
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.
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;
}
```
Worksheet: 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;
    private int turn;

    public Nim() {
        board = new int[]{7, 5, 3, 1};  // won't work
    }
}
```
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();
        Nim game2 = new Nim();
        game1.board[2] = 100;
        game1 = game2;
        System.out.println( game2.board[2] );
    }
}

What is printed by the above code? (Hint: Draw the 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
Exam 1

- Median: 7.5
- 49 perfect scores!

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 7 and an 8, you’re doing fine, but do come to discussion or office hours to make sure you understand everything you got wrong.

If you scored between a 6 and a 7, definitely come to office hours. You’re getting it, but we can help you improve your understanding.

If you scored lower than a 6, come talk to Prof. Alvarado about ways to improve your studying and understanding.
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.**
Split into 4 lines by last name

A-E     F-L     M-S     T-Z

Sort yourselves within each line