Example Midterm Problem

Suppose the following class has been defined:

```java
public class IntRef {
    public int data;
    public IntRef(int data) {
        this.data = data;
    }
}
```

What does the following code print out when run?

```java
IntRef var1 = new IntRef(1);
IntRef var2 = new IntRef(2);
IntRef var3 = new IntRef(3);
var2 = var1;
var2.data = 4;
var1 = var3;
var2.data = 5;
var1.data = 6;
System.out.println("1: " + var1.data);
System.out.println("2: " + var2.data);
System.out.println("3: " + var3.data);
```

Fast-Forwarding

- Usually, I teach the following topics next:
  - Growing ArrayIntLists automatically
  - Error handling and testing
    - Preconditions and postconditions
    - Throwing and catching exceptions
    - How to write good test code
  - But I want to cover linked lists before the midterm, so those topics are skipped for now. We may revisit them later.
- Next three lectures:
  - References
  - Linked nodes
  - Building a LinkedIntList class

CIS 211

References and Linked Nodes

slides created by Marty Stepp and Ethan Apter

http://www.cs.washington.edu/143/

Values vs. References

- Does the following swap method work? Why or why not?

```java
public static void main(String[] args) {
    int a = 7;
    int b = 35;
    // swap a with b?
    swap(a, b);
    System.out.println(a + " " + b);
}
```

```java
public static void swap(int a, int b) {
    int temp = a;
    a = b;
    b = temp;
}
```
Assignment and References

In Java, assignment (a = b) always copies the value stored by variable a to the value stored by variable b.

Example:
```java
int a = 5;
int b = 6;
a = b;  // value of a is now 6
```

When a and b are reference types (arrays or objects), the value that gets copied is a memory address.

Example:
```java
int[] a = {5};  // a's value is mem. loc. 0x0034FACE
int[] b = {6};  // b's value is mem. loc. 0x0034BEEF
a = b;  // a's value is mem. loc. 0x0034BEEF
```

More on references

When setting the contents of an array or the fields of an object, Java first goes to the memory location specified by the variable and then sets the appropriate value.

Example:
```java
int[] a = {1,2};
int[] b = {3,4};
a[0] = 10;
a = b;
a[0] = 100;
b = a;
```

We'll work out what's happening in memory on the blackboard.

Example Midterm Problem

Suppose the following class has been defined:
```java
public class IntRef {
    public int data;
    public IntRef(int data) {
        this.data = data;
    }
}
```

What does the following code print out when run?
```java
IntRef var1 = new IntRef(1);
IntRef var2 = new IntRef(2);
IntRef var3 = new IntRef(3);
var2 = var1;
var2.data = 4;
var1 = var3;
var1.data = 5;
var1.data = 6;
System.out.println("1: " + var1.data);
System.out.println("2: " + var2.data);
System.out.println("3: " + var3.data);
```

Method calls vs. assignment

• Calling a method is equivalent to assigning the method parameters to their given arguments:

```java
public static void main(String[] args) {
    int a = 7;
    int b = 35;
    // swap a with b? (Doesn't work!)
    swap(a, b);
    System.out.println(a + " " + b);
}
```
```java
public static void swap(int x, int y) {
    int temp = x;
    x = y;
    y = temp;
}
```
Method calls vs. assignment

• Calling a method is equivalent to assigning the method parameters to their given arguments:

```java
public static void main(String[] args) {
    int a = {7};
    int b = {35};
    // swap a with b? (Doesn’t work!)
    int x = a;
    int y = b;
    int temp = x;
    x = y;
    y = temp;

    System.out.println(a + " " + b);
}
```

Method calls vs. assignment

• Calling a method is equivalent to assigning the method parameters to their given arguments:

```java
public static void main(String[] args) {
    int[] a = {7};
    int[] b = {35};
    // swap a with b? (Doesn’t work!)
    int[] x = a;
    int[] y = b;
    int[] temp = x;
    x = y;
    y = temp;

    System.out.println(a[0] + " " + b[0]);
}
```

Method calls vs. assignment

• Calling a method is equivalent to assigning the method parameters to their given arguments:

```java
public static void main(String[] args) {
    int[] a = {7};
    int[] b = {35};
    // swap a with b? (Works!)
    int[] x = a;
    int[] y = b;
    int[] temp = x[0];
    x[0] = y[0];
    y[0] = temp;

    System.out.println(a[0] + " " + b[0]);
}
```