CS 367 - Introduction to Data Structures
Thursday, February 4, 2016

Homework 2 due 10 pm tomorrow, February 5th
Homework 3 assigned by Monday, February 8th, possibly sooner

Program 1 due 10 pm Sunday, February 14th, TIME IS RUNNING OUT!

Assignment questions? Post on Piazza or consult with a TA during scheduled hours.

Use the 367 Forms to report any exam conflicts or McBurney exam accommodations.

Email your instructor by tomorrow, Friday, 2/5, if you participate in religious observances that might interfere with course requirements. Include your name, UW ID#, date and explanation.

Last Time

Handin using the 367 Forms
Exceptions Review

- throwing
- handling
- execution
- practice with exception handling

Today

Exceptions Review (from last lecture)

- throws and checked vs. unchecked
- defining

Java Primitives vs. References Review
Chains of Linked Nodes

- Listnode class
- practice with chains of nodes

Next Time

Read: continue *Linked Lists*
Chains of Linked Nodes

- more practice with chains of nodes

Java Visibility Modifiers
LinkedList Class

Primitive vs. Reference Types: Assignment

Primitives

assume code is in main() — Call Stack

```
int x, y, z;
x = 11;
y = x;
z = x;
z = 33;
y = 22;
```

→ What does each variable contain after the code above executes?

A.) x has y has z has
B.) x has y has z has
C.) x has y has z has

✵

References

assume code is in main() — Call Stack | Heap

```
ArrayList<String> x, y, z;
x = new ArrayList<String>();
y = x;
z = x;
y = new ArrayList<String>();
z.add("Computer");
y.add("Science");
```

→ What does each ArrayList contain after the code above executes?

A.) x's ArrayList has y's ArrayList has z's ArrayList has
B.) x's ArrayList has y's ArrayList has z's ArrayList has
C.) x's ArrayList has y's ArrayList has z's ArrayList has

→ What do x, y and z contain?

✵

Primitive vs. Reference Types: Parameter Passing

Primitives

Call Stack | Heap

Given:

```
void mod1(int x) {
   x = 42;
}
```

Execute code in `main()`:

```
int   x = 11;
int[] y = {11, 22, 33};
mod1(x);
mod1(y[2]);
```

→ What does variable `x` and array `y` in `main` contain after the code above executes?

A.) `x` has `y`'s array has
B.) `x` has `y`'s array has
C.) `x` has `y`'s array has

→ What happens if we call `mod1(y)` in main?

Primitive vs. Reference Types: Parameter Passing

References

Call Stack | Heap

Given:

```
void mod2(int[] x) {
   x[0] = 21;
}

void mod3(int[] x) {
   x = new int[x.length];
   x[0] = 42;
}
```

Execute code in `main()`:

```
int   x = 11;
int[] y = {11, 22, 33};
mod2(y);
mod3(y);
```

→ What does variable `x` and array `y` in `main` contain after the code above executes?

A.) `x` has `y`'s array has
B.) `x` has `y`'s array has
C.) `x` has `y`'s array has

→ What happens if we call `mod2(x)` in main?

Programmer's Memory Model for Java

Call Stack

contains

birth

death

Heap

contains

birth

death

Static Data

contains

birth

death

New Data Structure - Chain of Linked Nodes

The Data Structure

Array **vs.** **Chain of Nodes**

Goal

`Listnode` Class

```
class Listnode<E> {


   private E data;
   private Listnode<E> next;



   public Listnode(E d) {
      this(d, null);
   }

   public Listnode(E d, Listnode<E> n) {
      data = d;
      next = n;
   }



   public E getData()                  { return data; }

   public Listnode<E> getNext()        { return next; }



   public void setData(E d)            { data = d; }

   public void setNext(Listnode<E> n) { next = n; }
}
```

Practice: Using Listnodes

→ **Draw a memory diagram** corresponding to the given code:

assume code is in main() Call Stack | Heap

```
Listnode<String> n1 = null;
```

→ **Write the code that results in:**