CS 251 - Lab 004

TA: Kage Weiss
Office Hours: R 2-2:50 FEC 2000, or by appointment.
Contact: mmweiss@unm.edu
Website: **http://cs.unm.edu/~kageweiss/TA/cs251.html** **-- SLIDES POSTED**

- Sign in sheet located on desk by TA
- Today we are working on MIDTERM REVIEW and Postfix (Lab 5)
  - **We are in the process of catching everyone's grades up as there is a backlog, please be watching your grades for updates and let us know if you have ANY questions going into the midterm. The plan is to have all Labs 3/4 graded by tonight and the remaining Lab 2s post-midterm.**
  - **What is the best thing you can do to prepare for TOMORROW's MIDTERM? Study the previous midterms! Seriously!**

Midterm Review

Week 1 – Can you code?
- Java Basics
- loops, types, methods, classes
- JavaDoc Comments

Week 2,3 – Classes Extended
- Enums, Extension, Interface
- Inheritance: Extend, Implement

Week 4 – Nested Classes
- When/Why/How

Week 5 – Exceptions, KEYWORDS, Class<Generic>
- throws/throw
- KEYWORDS
- Interfaces: Generic classes

Week 6 –Collections, Collection
- Collection frames

Midterm Review

- Week 2,3 – Classes Extended
  - Enums, Extension, Interface
    - What are the basic methods for Enums?
  - Inheritance: Extend, Implement
    - What does inheritance do?
      - What is and isn’t shared from your parent?
    - What is the difference between:
      - Override
      - Overload
      - Hide
    - Interface v. Abstract class?

Midterm Review

- Week 2,3 – Classes Extended
  - Enums, Extension, Interface
    - What are the basic methods for Enums?
  - Inheritance: Extend, Implement
    - What does inheritance do?
      - What is and isn’t shared from your parent?
    - What is the difference between:
      - Override – same signature as parent non-static method
      - Overload – same name different signature as another method
      - Hide – same signature as parent static method
    - Interface v. Abstract class?
      - Interfaces are limited to signatures and constants
      - Abstract classes are not, but must be extended (limit 1)

Midterm Review

- Week 4 – Nested Classes
  - When/Why/How
    - Privacy?
      - When should it be public?
    - Permanency?
      - Is this class immutable or a member object?
    - Referencing:
      - this.n
      - this('n')
      - super.n
      - super('n')

Midterm Review

- Week 4 – Nested Classes
  - When/Why/How
    - Privacy?
      - When should it be public?
    - Permanency?
      - Is this class immutable or a member object?
    - Referencing:
      - this.n – instance value named n in **this** class instance
      - this(“n”) – referential call to a constructor of **this** class
      - super.n – instance value named n in this’s **parent** class instance
      - super(“n”) – call to a constructor for this’s **parent** class

Midterm Review

Week 5 – Exceptions, KEYWORDS, Class<Generic>

- throws/throw
  - A method ***throws*** an Exception when a line inside will ***throw*** it.
  - Always be sure to ***try*** and ***catch*** our *(Exception e)*
  - and then properly handle it, and ***finally***, do what we need to.
- KEYWORDS
  - Seriously, you just must learn these.
- Interfaces: Generic classes and collection frames
  - How do generic classes help us?
    - How is leaving undefined the type we're handling useful?

Midterm Review

Week 5 – Exceptions, KEYWORDS, Class<Generic>

- throws/throw
  - A method ***throws*** an Exception when a line inside will ***throw*** it.
  - Always be sure to ***try*** and ***catch*** our *(Exception e)*
  - and then properly handle it, and ***finally***, do what we need to.
- KEYWORDS
  - Seriously, you just must learn these.
- Interfaces: Generic classes and collection frames
  - How do generic classes help us?
    - <T> "type", <E> "element", <K, V> "key, value", etc.
    - How is leaving undefined the type we're handling useful?

Midterm Review

- Week 6 – Collections, Collection
  - Interfaces: collection frames
    - What collections do we know?
      - Array
      - List (ooh, lookie, an Interface)
        - LinkedList, ArrayList, Deque, Queue
      - Map (another pesky Interface)
        - HashMap
        - How do we use maps?
      - Set (who wants to guess whether this is an Interface?)
        - HashSet
    - What are some useful Collection**s** methods?

Midterm Review

```
public class SuperClass{
        public var x;
}
```

“sysou” Ctrl + Space is the Eclipse shortcut for **System**.**out**.println();

MainClass.method1(x) will print out:

A) x
   x
   x

B) x
   x
   x

C) x
   x
   x

```
public class MainClass extends SuperClass{
        var x;

        void method1(var x){
                sysou(x)
                sysou(this.x)
                sysou(super.x)
        }
}
```

Midterm Review

```
public class SuperClass{
        public var x;
}
```

```
public class MainClass extends SuperClass{
        var x;

        void method1(var x){
                sysou(x)
                sysou(this.x)
                sysou(super.x)
        }
}
```

"sysou" Ctrl + Space is the Eclipse shortcut for **System**.**out**.println();

MainClass.method1(x) will print out:

B) **x**
x
x

Always refer to the closest version of a variable with the same name (and check privacy).
"x" exists in method1, so that's the one it looks at.
"this.x" refers to MainClass.x, and "super.x" refers to the class above MainClass, so SuperClass.x

Midterm Review

```
public class SuperClass{
        public var x;
}
```

“sysou” Ctrl + Space is the Eclipse shortcut for **System**.**out**.println();

MainClass.method2(x) will print out:

```
A) x        B) x        C) x
   x           x           x
   x           x           x
```

```
public class MainClass extends SuperClass{
        var x;
        void method1(var x){
                sysou(x)
                sysou(this.x)
                sysou(super.x)
        }
        void method2(var y){
                sysou(x)
                sysou(this.x)
                sysou(super.x)
        }
}
```

Midterm Review

```
public class SuperClass{
        public var x;
}
```

"sysou" Ctrl + Space is the Eclipse shortcut for **System**.**out**.println();

MainClass.method2(x) will print out:

A) x
   x
   x

Always refer to the closest version "x" does not exist in method2, so the closest "x" is in MainClass

```
public class MainClass extends SuperClass{
        var x;
        void method1(var x){
                sysou(x)
                sysou(this.x)
                sysou(super.x)
        }
        void method2(var y){
                sysou(x)
                sysou(this.x)
                sysou(super.x)
        }
}
```

Midterm Review

```
public class SuperClass{
        var x;
        String printOut(){
                sysou(x)
                return(this.x)
        }
}
```

MainClass.method1(x) will print out:

A) x
   x
   x
   x

B) x
   x
   x

C) x
   x
   x
   x

```
public class MainClass extends SuperClass{
        var x;

        void method1(var x){
                sysou(x)
                sysou(this.x)
                sysou(super.printOut())
        }
}
```

Midterm Review

```
public class SuperClass{
        var x;
        String printOut(){
                sysou(x)
                return(this.x)
        }
}
```

```
public class MainClass extends SuperClass{
        var x;

        void method1(var x){
                sysou(x)
                sysou(this.x)
                sysou(super.printOut())
        }
}
```

MainClass.method1(x) will print out:

A) x
x
x
x

Always refer to the closest version of a variable with the same name (and check privacy).
"x" exists in method1, so that's the one it looks at.
"this.x" refers to MainClass.x, and "super.printout()" refers to the printOut() method in the class above MainClass, so SuperClass.printOut() which prints **and** returns SuperClass.x