CS 367 - Introduction to Data Structures
Tuesday, February 2, 2016

Homework 1 due 10 pm tonight, February 2nd
Homework 2 assigned tonight, due 10 pm this Friday, February 5th

Program 1 due 10 pm Sunday, February 14th, GET STARTED NOW!

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 this 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

Iterators

- iterators and the Java API
- using iterators
- options for implementing iterators
- making a class iterable

Today

Handin using the 367 Forms
Exceptions Review

- throwing
- handling
- execution
- practice with exception handling
- throws and checked vs. unchecked
- defining

Next Time

Read: finish *Exceptions*, start *Linked Lists*
Java Primitives vs. References Review
Chains of Linked Nodes

- Listnode class
- practice with chains of nodes

Exception Throwing – Signaling a Problem

Java Syntax

```
throw exceptionObject;
```

Example

Exception Handling – Resolving a Problem

Java Syntax

```
try {
   // try block
   code that might cause an exception to be thrown

} catch (ExceptionType1 identifier1) {
   // catch block
   code to handle exception type 1

} catch (ExceptionType2 identifier2) {
   // catch block
   code to handle exception type 2

}
... more catch blocks

finally {
   // finally block - optional
   code always executed when try block is entered
}
```

Example

Exception Execution

Normal Execution

- Start: top of main()

- Execute:

- Skip:

- Switch to Exception Handling Execution

Exception Handling Execution

- Skip:

- Execute:

- Switch back to Normal Execution

Searching for a Matching Catch

1. Locally

2. Remotely

Checking a Match

1. Match Found

2. No Match Found

ExceptionTester Example

```
public class ExceptionTester {

   public static void main(String[] args) {
      System.out.print("main[");
      try {
         methodA( ); System.out.print("after A,");
         methodE( ); System.out.print("after E,");
      } catch (RedException exc) {
         System.out.print("main-red,");
      } catch (GreenException exc) {
         System.out.print("main-green,");
      } finally {
         System.out.print("main-finally,");
      }
      System.out.println("]main");
   }

   private static void methodA( ) {
      System.out.print("\nA[");
      try {
         methodB( );
         System.out.print("after B,");
      } catch (BlueException exc) {
         System.out.print("A-blue,");
      }
      System.out.println("]A");
   }

   private static void methodB( ) {
      System.out.print("\nB[");
      methodC( );
      System.out.print("after C,");
      try {
         methodD( );
         System.out.print("after D,");
      } catch (YellowException exc) {
         System.out.print("B-yellow,");
         throw new GreenException();
      } catch (RedException exc) {
         System.out.print("B-red,");
      } finally {
         System.out.print("B-finally,");
      }
      System.out.println("]B");
    }
```

What is Output When:

1. no exception is thrown

```
main[
A[
B[
```

2. `methodE` throws a `YellowException`?

```
main[
A[
B[
```

3. `methodC` throws a `GreenException`?

```
main[
A[
B[
```

4. `methodD` throws a `GreenException`?

```
main[
A[
B[
```

What is Output When:

5. `methodC` throws a `RedException`?

```
main[
A[
B[
```

6. `methodD` throws a `RedException`?

```
main[
A[
B[
```

7. `methodD` throws a `YellowException`?

```
main[
A[
B[
```

8. `methodD` throws a `OrangeException`?

```
main[
A[
B[
```

What is Output When:

9. `methodC` throws a `YellowException`?

```
main[
A[
B[
```

10. `methodC` throws a `BlueException`?

```
main[
A[
B[
```

11. `methodE` throws a `RedException`?

```
main[
A[
B[
```

`throws` clause – Passing the Buck

Checked vs. Unchecked

Java Syntax

```
... methodName(parameter list)
       throws ExceptionType1, ExceptionType2, ... {
    ...
}
```

Example

```
public static void main(String[] args) throws IOException { ...
```

Defining a New Exception Class

Checked

```
public class MyException extends _____________________ {


}
```

Unchecked

```
public class MyException extends _____________________ {


}
```

Example

```
public class EmptyBagException extends Exception {

    public EmptyBagException() {
        super();
    }

    public EmptyBagException(String msg) {
        super(msg);
    }

}
```