Homework 2:

Program 1:

Assignment questions? Discuss within your CS367 Team, post on Piazza, or consult with a TA during scheduled hours.

Report any exam conflicts or McBurney exam accommodations by this Friday 2/3.

Email your instructor by this Friday, 2/3, if you participate in religious observances that might interfere with course requirements. Include your name, UW ID#, date and explanation.

Last Week
Using lists via the ListADT, implementing ListADT using an array (SimpleArrayList), Java’s List interface, Iterators: concept, using, options for implementation, making a class Iterable

This Week
Read: Exceptions, start Linked Lists
Teams
Exceptions Review
• throwing
• handling
• execution
• practice with exception handling
• throws and checked vs. unchecked
• defining
Java Primitives vs. References Review, p.14
Chains of Linked Nodes
• Listnode class
• practice with chains of nodes

Next Week
Read: continue Linked Lists:
Chains of linked nodes, more practice with chains of nodes, Java Visibility Modifiers, LinkedList class, LinkedListIterator class, LinkedList variations, tail reference, header node, double linking, circular linking
Exception Throwing – Signaling a Problem

Java Syntax

    throw exceptionObject;

Example
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
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`?

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

10. `methodC` throws a `BlueException`?

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

11. `methodE` throws a `RedException`?

```java
main[
A[
B[
```
throws clause – Passing the Buck

Checked Exceptions vs. Unchecked Exceptions

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 (if you want to support an optional message)

public class EmptyBagException extends Exception {

    public EmptyBagException() {
        super();
    }

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

}
What is Output When SOLUTION:

1. no exception is thrown

```java
main[
 A[
 B[after C,after D,B-finally,]B
 after B,]A
 after A,after E,main-finally,]main
```

2. methodE throws a YellowException?

```java
main[
 A[
 B[after C,after D,B-finally,]B
 after B,]A
 after A,main-finally,Exception in thread "main" YellowException
 at ExceptionTester.methodE(ExceptionTester.java:145)
 at ExceptionTester.main(ExceptionTester.java:37)
```

3. methodC throws a GreenException?

```java
main[
 A[
 B[main-green,main-finally,]main
```

4. methodD throws a GreenException?

```java
main[
 A[
 B[after C,B-finally,main-green,main-finally,]main
```
What is Output When SOLUTION:

5. methodC throws a RedException?

    main[
    A[
    B[main-red,main-finally,]main

6. methodD throws a RedException?

    main[
    A[
    B[after C,B-red,B-finally,]B
    after B,]A
    after A,after E,main-finally,]main

7. methodD throws a YellowException?

    main[
    A[
    B[after C,B-yellow,B-finally,main-green,main-finally,]main

8. methodD throws a OrangeException?

    main[
    A[
    B[after C,B-finally,main-finally,Exception in thread "main"
OrangException
     at ExceptionTester.methodD(ExceptionTester.java:129)
at ExceptionTester.methodB(ExceptionTester.java:80)
at ExceptionTester.methodA(ExceptionTester.java:60)
at ExceptionTester.main(ExceptionTester.java:34)
What is Output When SOLUTION:

9. methodC throws a YellowException?

```java
main[
A[
B[main-finally,Exception in thread "main" YellowException
   at ExceptionTester.methodC(ExceptionTester.java:109)
   at ExceptionTester.methodB(ExceptionTester.java:76)
   at ExceptionTester.methodA(ExceptionTester.java:60)
   at ExceptionTester.main(ExceptionTester.java:34)
]
```  

10. methodC throws a BlueException?

```java
main[
A[
B[A-blue,]A
after A,after E,main-finally,]main
```  

11. methodE throws a RedException?

```java
main[
A[
B[after C,after D,B-finally,]B
after B,]A
after A,main-red,main-finally,]main
```
Primitive vs. Reference Types: Assignment

Primitives

assume code is in main()

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()

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

Given:

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

Execute code in `main()`:

```java
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

Given:

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

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

Execute code in `main()`:

```java
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
dead

Heap
contains

birth
dead

Static Data
contains

birth
dead
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: