06-0: **Errors**

- Errors can occur in program
  - Invalid input / bad data
  - Unexpected situation
  - Logic error in code
- Like to handle these errors gracefully, not just halt the program
  - Running a web server, don’t want one piece of bad data to bring the whole thing down

06-1: **Error Checking**

- We could check for any conceivable error

```java
ArrayList<Integer> A;
// Initialize A
A.set(i, new Integer(x/y));
```

06-2: **Error Checking**

- We could check for any conceivable error

```java
if (i >= 0 && i < A.size())
    if (y != 0)
        A.set(i, new Integer(x / y));
    else
        // Handle division by zero case
    // Handle outside bounds of the array case
```

- Problems with this method?

06-3: **Exceptions**

- We can let the system catch all the errors for us
  ```java
  A.set(i, new Integer(x / y));
  ```
- Throws an exception if i ≥ 0, i < A.size(), y == 0. Program ends.
- Problems with this method?

06-4: **Exceptions**

- We can let the system catch the errors for us
- We can “catch” the errors ourselves

```java
try
    A.set(i, new Integer(x,y));
catch (Exception e)
    // do some work to clean up after the exception
```

06-5: **Try-Catch Block**
try
{
    // Any Java Code
}
catch (Exception e)
{
    // Any Java Code
}

- If an exception is raised inside the try block:
  - Stop immediately and execute the code in the catch block
  - Continue after the catch block as normal

- If no exception is raised inside the try block
  - Ignore the code in the catch block

06-6: Exceptions

```java
int x;
int y;
try
|
  x = 3;
  y = 0;
  x = x / y;
  System.out.println("Can't get here!");
} catch (Exception e)
|
  System.out.println("Exception caught!");
} System.out.println("Done with try block");
```

06-7: Exceptions

```java
int x;
int y;
try
|
  x = 3;
  y = 5;
  x = x / y;
  System.out.println("We will get here!");
} catch (Exception e)
|
  System.out.println("We won't get here!");
} System.out.println("Done with try block");
```

06-8: Exceptions

```java
ArrayList<Integer> A = new ArrayList<Integer>();
for (int i = 0; i < 10; i++)
|
  A.add(new Integer(i));
} try
|
  for (int i = 0; i < 10; i++)
      System.out.println(i);
  for (int i = 10; i > 0; i--)
      System.out.println(i);
} catch (Exception e)
|
  System.out.println("Error!");
} System.out.println("Done with try block");
```

06-9: Exceptions

```java
ArrayList<Integer> A = new ArrayList<Integer>();
for (int i = 0; i < 10; i++)
|
  A.add(new Integer(i));
} try
|
  for (int i = 0; i < 10; i++)
      System.out.println(A.get(i));
  for (int i = 10; i > 0; i--)
      System.out.println(A.get(i));
} catch (Exception e)
|
  System.out.println("Error!");
} System.out.println("Done with try block");
```
06-10: Exceptions

- Variables declared within a try block are not visible outside
- Actually, variables declared within *any* block are not visible outside the block

```java
try {
    ArrayList<Integer> A = new ArrayList<Integer>();
    for (int i = 0; i < 10; i++)
        A.add(new Integer(i));
}
catch (Exception e) {
    System.out.println("Error!");
    A.set(3, new Integer(5)); // ERROR!
}
```

06-11: Exceptions

- What went wrong?

```java
try {
    A.set(new Integer(x/y));
}
catch (Exception e) {
    System.out.println(e.getMessage());
}
```

- We’ll do more with this after Inheritance

06-12: Uncaught Exceptions

```java
int divide(int x, int y) {
    int result = x / y;
    System.out.println(result);
    return result;
}
void foo() {
    int x = 6;
    x = divide(x, 2);
    x = divide(x, 1);
    x = divide(x, 0);
    x = divide(x, 3);
}
```

06-13: Uncaught Exceptions

```java
int divide(int x, int y) {
    int result = x / y;
    System.out.println(result);
    return result;
}
void foo() {
    try {
        int x = 6;
        x = divide(x, 3);
        x = divide(x, 2);
        x = divide(x, 0);
        x = divide(x, 2);
    } catch (ArithmeticException e) {
        System.out.println("Error!");
    }
}
```

06-14: Uncaught Exceptions

```java
int divide(int x, int y)
    { int result = x / y;
    System.out.println(result);
    return result;
    }
void foo()
    { try
        { int x = 6;
        x = divide(x, 3);
        x = divide(x, 2);
        x = divide(x, 0);
        x = divide(x, 2);
        } catch (ArithmeticException e)
        { System.out.println("Error!");
        }
    }
void divideBySelf(ArrayList<Integer> A)
    { for (int i = 0; i < A.size(); i++)
        divideBySelf(A);
    }
```
```java
int res = divide(A.get(i), A.get(i));
A.set(res);
} catch (ArithmeticException e) {
    System.out.println("Excp. caught!");
}

06-15: Uncaught Exceptions

```
06-20: **In-Class Assignment**

- Go to lecture note website
- Get ListFun.java
- Fill in body of reverse2, so that it returns a reversed copy of the list passed in (without changing the list passed in)