CS3500: Object-Oriented Design
Spring 2014

Class 6
1.24.2014

Northeastern University
College *of* Computer and Information Science

Today...

- Liskov - Chapter 3: Procedural Abstraction
- Liskov - Chapter 4: Exceptions
- Maps
- Liskov - Chapter 5: Data Abstraction

Office Hours This Week

- Tuesday, January 21: 12:30-1:25pm
- Thursday, January 23: 12:30-2:30pm
- **Friday, January 24: NO OFFICE HOURS**

Assignment 3

- MyList implementation
- Due ~~Friday, January 24, 2014 (TONIGHT)~~
  Saturday, January 25 (TOMORROW)

Chapter 3: Procedural Abstraction

[Liskov]

Procedural abstraction: "An abstraction that hides details associated with executing an operation or task." [Liskov]

Benefits of Abstraction

[Liskov]

- Abstraction by parameterization:
  - irrelevant: identity of the actual data
  - relevant: presence, number, and type of the actuals
- Abstraction by specification:
  - irrelevant: "how" is done
  - relevant: "what" is done

Specifications

- Formal
- Informal

Template for Procedural Abstraction

[Liskov]

```
return_type pname (...)
```

//REQUIRES: This clause states any constraints on use

//MODIFIES: This clause identifies all modified inputs

//EFFECTS: This clause defines the behavior

Template for Class Providing Standalone Procedures

[Liskov]

```
visibility cname {
```

//OVERVIEW: This clause defines the purpose of the //class as a whole

```
  visibility static p1 ...
  visibility static p2 ...
}
```

```
public class Arrays{
  //OVERVIEW: This class provides a number of standalone procedures that
  //   are useful for manipulating arrays of ints.

  public static int search (int[] a, int x)
    //EFFECTS: If x is in a, returns an index where x is stored;
    // otherwise, returns -1

  public static int searchSorted (int[] a, int x)
    //REQUIRES: a is sorted in ascending order
    //EFFECTS: If x is in a, returns an index where x is stored'
    //  otherwise, returns -1.

  public static void sort (int[] a)
    //MODIFIES: a
    //EFFECTS: Rearranges the elements of a into ascending order
    //  e.g., if a = [3, 1, 6, 1] before the call,
    //  on return a = [1, 1, 3, 6].
}
```

Procedures

- Total
- Partial

ICE: Liskov Exercise 3.2

Specify and implement a method with the header

```
public static int sum (int[] a)
```

that returns the sum of the elements of a

ICE: Liskov Exercise 3.4

Specify and implement a procedure that determines whether or not a string is a palindrome. (A palindrome reads the same backward and forward; an example is "deed.")

Chapter 4: Exceptions

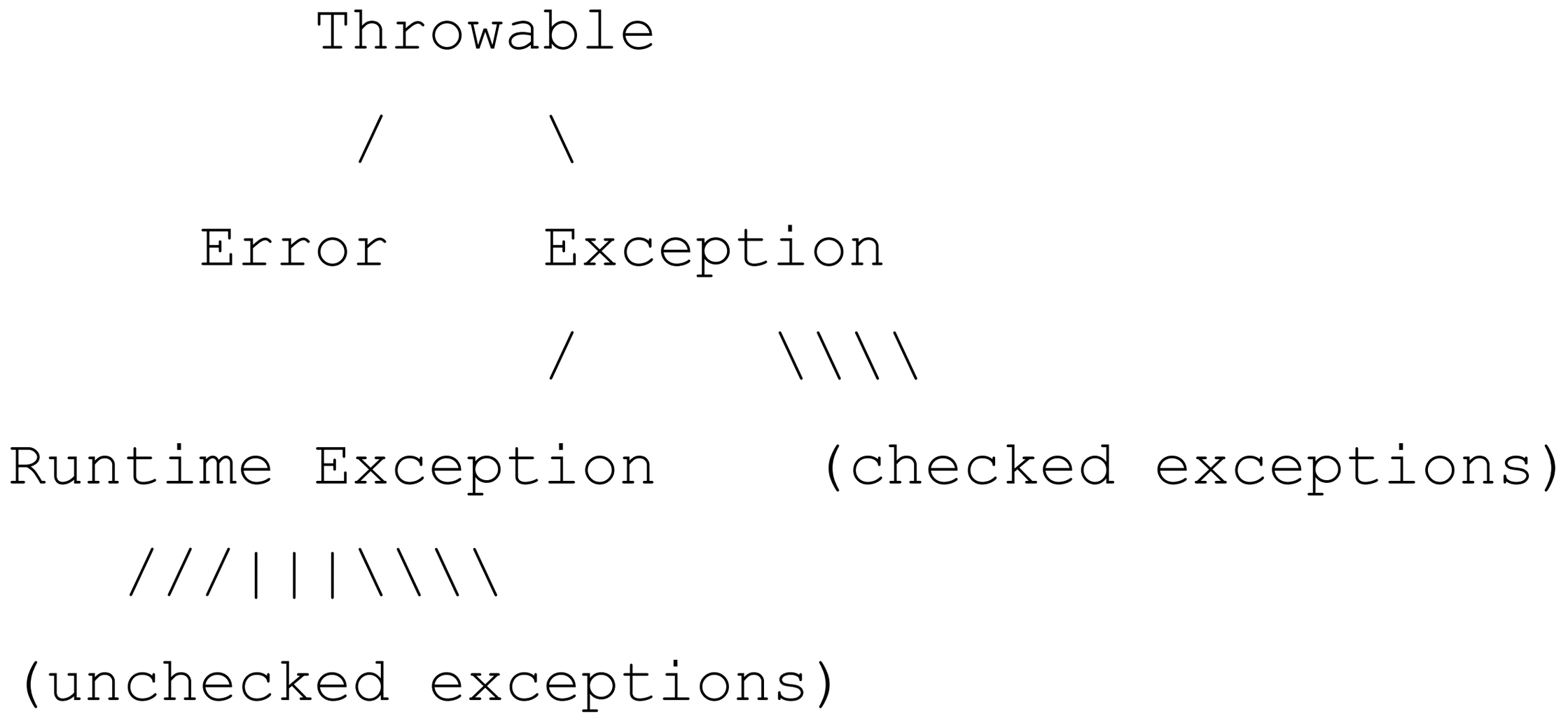
[Liskov]

```
public static int factorial (int n) throws NonPositiveException
  // EFFECTS: if n is non-positive throws NonPositiveException
  //          returns the factorial of n


public static int search (Vector<Integer> v, int x)
  throws NullPointerException, NotFoundException
  // REQUIRES: v is sorted
  // EFFECTS: if v is null throws NullPointerException
  //          if x is not found in v throws NotFoundException
  //          else returns i such that v.get(i) = x
```

Types of Exceptions

[Liskov]



Exceptions within Code

- `<method header> throws <exception>`
- `throw expression;`
- `try`
  `body`
  `catch (<exception> e1) <catch body>`
  `catch (<exception> e1) <catch body>`
  `...`
  `finally <finally body>`
- `@throws`

Exceptions within the Recipe

- If the algebraic specification does not contain any equations that describe the result of an operation f applied to an instance of C, then the body of the dynamic method f should throw a RuntimeException such as an IllegalArgumentException.
- Similarly, the body of the dynamic method f should throw a RuntimeException if there is no relevant equation (because the side conditions for all potentially relevant equations are false).

Exceptions: Items 57-65

[Bloch]

- Item 57: Use exceptions only for exceptional conditions
- Item 58: Use checked exceptions for recoverable conditions and runtime exceptions for programming errors
- Item 59: Avoid unnecessary use of checked exceptions
- Item 60: Favor the use of standard exceptions
- Item 61: Throw exceptions appropriate to the abstraction
- Item 62: Document all exceptions thrown by each method
- Item 63: Include failure-capture information in detail messages
- Item 64: Strive for failure atomicity
- Item 65: Don't ignore exceptions

Data Structures

- List
- Stack
- Queue
- Set
- Map

Comparison of Lists, Sets, and Maps [from Reges and Stepp]

Data Structure	Description/ Strengths	Weaknesses	Example Usages
List	A sequence of elements arranged in order of insertion	Slow to search, slow to add/remove arbitrary elements	List of accounts; prime numbers' the lines of a file
Set	A set of unique elements that can be searched quickly	Does not have indexes' user cannot retrieve arbitrary elements	Unique words in a book; lottery ticket numbers
Map	A group of associations between pairs of "key" and "value" objects	Not a general-purpose collection; cannot easily map backward from a value to its key	Word counting; phone book creation

Assignment 4

- Implement MyMap
- Due: Friday, January 31, 2014 at 11:59 pm