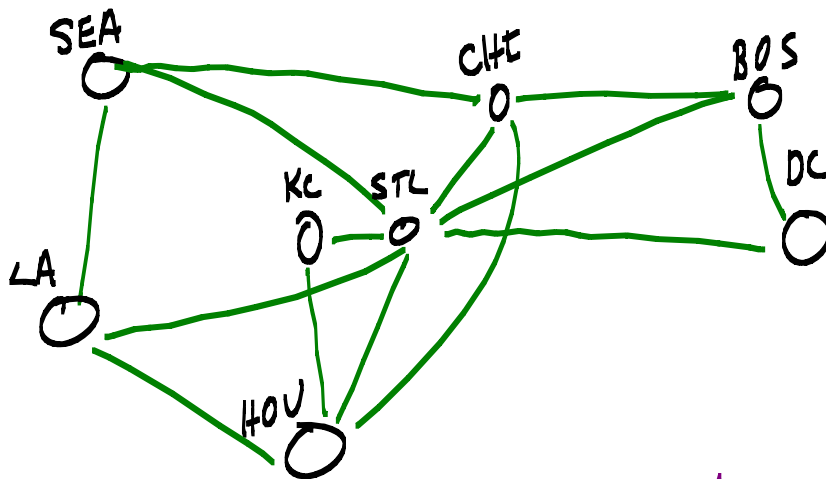


# Representation Invariants, Abstraction Functions, & Exceptions

Note Title

1/30/2007

- Today:
- Representation Invariants
  - Abstraction Functions
  - Iterator vs. Visitor Pattern
  - Exception Handling
- } Graph Example



Graph = Vertices + Edges

$$G = (V, E)$$

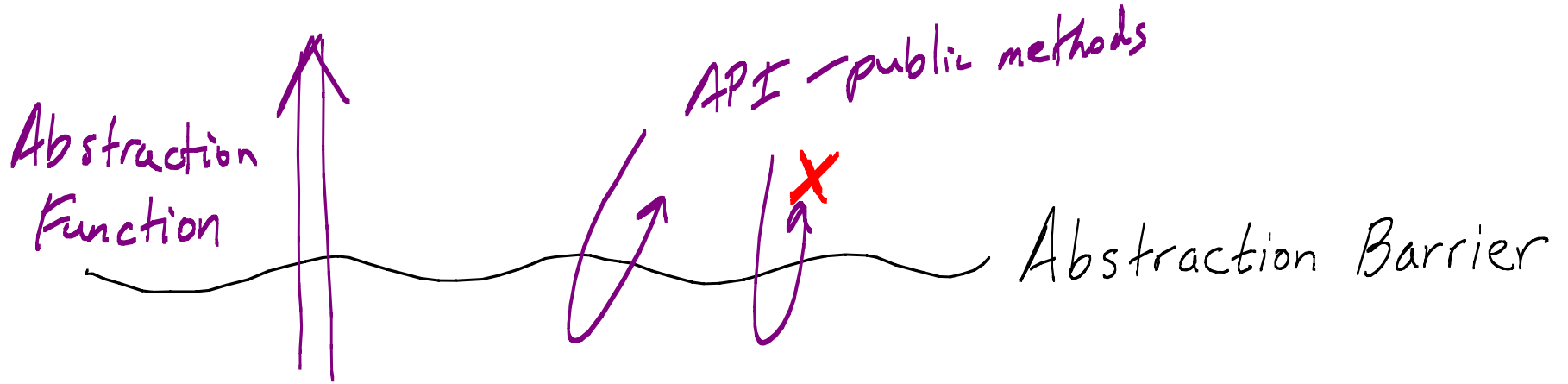
Here:  $V$  = set of cities

$E$  = flights

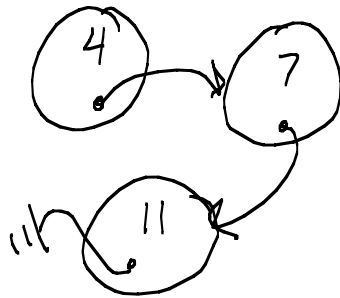
Graphs can be Directed or Undirected

User  
[4, 7, 11]

void remove( $E x$ ) throws  
NoSuchElementException

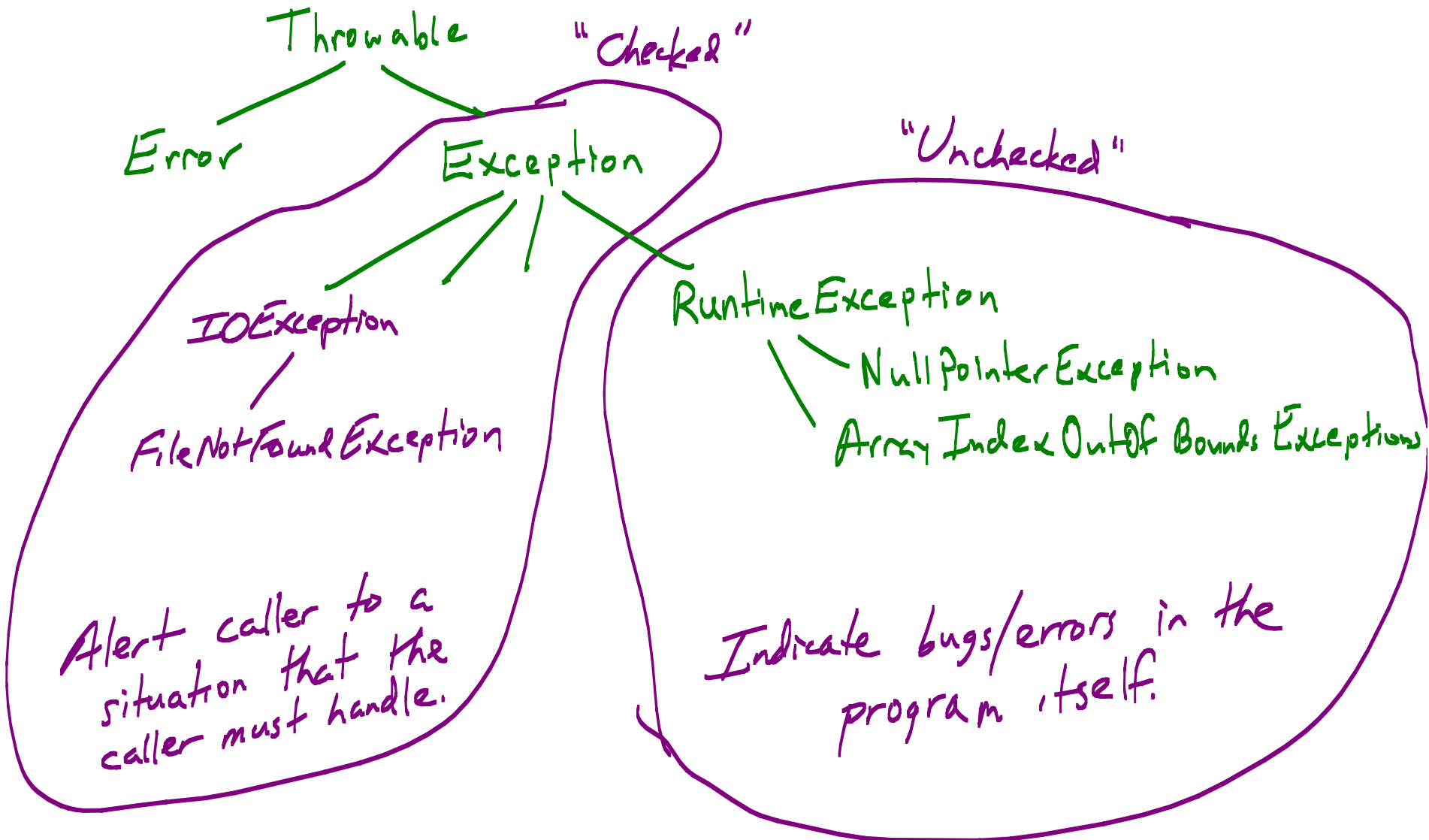


Size  
**3**



Representation Invariants  
help guarantee self-consistency  
repOK - checks these

Internal Representation



throws MyFooException  
void foo() {

{ try {

== x ← exception here

○ ← won't execute

} catch (MyException me) {

==

} catch (FileNotFoundException fnfe) {

==

} finally {

==

}

}

in here,  
can return  
- throw  
- neither

in here,  
can return

has the  
last word  
- throw  
- neither

```
while (fileNotOpened) {  
    try {  
        get file name from user  
        open the given file  
    } catch (FileNotFoundException fnfe) {  
        complain to user  
    }  
}
```

setPixel(int x, int y) throws BadPixelException {

try {

set the pixel

} catch (ArrayIndexOutOfBoundsException aioobe) {

throw new BadPixelException(x, y);

}

}



setPixel(int x, int y) throws ArrayIndexOutOfBoundsException {

set the pixel;

}

Stack data structure Stack(E)

```
E pop() {  
    if (head == null)  
        throw new NoSuchElementException();  
  
    E temp = head.value;  
    head = head.next;  
    return temp;  
}
```

E pop() E

```
try E  
    return head.value;  
} X {  
    catch (NullPointerException n) {  
        throw new  
            NoSuchElementException();  
    }  
} finally  
    try E  
        head = head.next;  
    } catch (NullPointerException npe) {  
        throw new NoSuchElementException();  
    }  
}
```