Representation Invariants, Abstraction Functions, & Exceptions

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

Graph Example

Graph = Vertices + Edges

Here: $V = \text{set of cities}$
$E = \text{flights}$

Graphs can be Directed or Undirected
User

\[4, 7, 11\]

Abstraction Function

Size 3

4 → 7

11

Internal Representation

Representation Invariants help guarantee self-consistency repOK — checks these

API - public methods

void remove(E x) throws No SuchElementException
Throwables

```
Exception

IOException
FileNotFoundException
```

Alert caller to a situation that the caller must handle.

```
Checked

Unchecked

RuntimeException
NullPointerException
ArrayIndexOutOfBoundsException
```

Indicate bugs/errors in the program itself.
```java
void foo() {
    try {
        x <- exception here
        // non-executable code

        3 Catch (MyException me) {
        }

        3 Catch (FileNotFoundException fnfe) {
        }

        3 Finally {
            has the last work
        }
    }
}
```
while (fileNotOpened) {
    try {
        get file name from user
        open the given file
    } catch (FileNotFoundException fnfe) {
        Complain to user
    }
}
```java
setPixel(int x, int y) throws BadPixelException {
    try {
        set the pixel
    } catch (ArrayIndexOutOfBoundsException e) {
        throw new BadPixelException(x, y);
    }
}
```
Stack data structure $Stack(E)$

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

```java
E pop() {
    try {
        return head.value;
    } catch (NullPointerException npe) {
        throw new NoSuchElementException();
    }
    finally {
        head = head.next;
        catch (NullPointerException npe) {
        }
    }
} else {
    // handle other exceptions
}
```