Computer Science 112
Data Structures

Lecture 07:
ArrayLists
Stacks

Review: Exceptions

When an error occurs, an exception is thrown.

- **An exception is an object**
  - **Its class is a descendant of Exception**
  - **Its class tells you what error has occurred**
    - **ArrayIndexOutOFBoundsException**
    - **NumberFormatException**

When an error occurs ...

E.g, when code tries to access a field of a null pointer

- **An exception that is an instance of the appropriate class is created**
- **This exception (instance) is "thrown"**
  - **The throw is caught by a try-catch statement waiting on the stack, or else**
  - **the throw causes the program to crash**

To throw an exception

- **Use the throw statement:**

```
throw new NoSuchElementException(j+" ");
```

Checked vs Runtime Exceptions

- **Checked Exceptions**
  - **Classes are descendants of Exception but not of RuntimeException**
  - **Require throws clauses in method headers**

```
public void foo(int x, int y)
  throws IOException{
```

  - **Represent user or environmental errors**
    - **FileNotFoundException**

Checked vs Runtime Exceptions

- **Runtime Exceptions**
  - **Classes are descendants of RuntimeException**
  - **Do not use throws clauses in method headers**
  - **Represent program errors**
    - **ArrayIndexOutOfBoundsException**

To catch a throw

```
try{
    <statements>
} catch (<class of exceptions> <variable>){
    <statements>
}
```

See main in DriveLLE.java

Finding a Catch

- **A catch is active during the time its try statements are executing**
  - **Including any methods they call**

```
try{ foo( )} catch (FileNotFoundException e)
    {…};
void foo( ){ … fie( ); …}
void fie( ){ … }
```

Finding a Catch

When an exception is thrown, java finds

- **The innermost active try**
  - **innermost = most recently entered**
  - **Active = not exited**
- **Where the exception being thrown is a subclass of the class in the catch**

Once catch is found

- **Skip rest of the try;**
- **Go immediately to the statements in the catch**

Bad uses of try-catch

- **Don't use it where an if, break, return, etc. would be simpler**
- **Don't use it to simply ignore an error**

Review: Generics

- **Consider ReadOnlyPairString:**
  - **Cf: ReadOnlyPairInteger.java**
    - **Class declarations and methods are almost identical**
  - **Solution "generics" (Java 5 & later)**
    - **Class & method definitions parameterized by type**
  - **See ReadOnlyPairInteger.java, ReadOnlyPairString.java, ReadOnlyPair.java**

Generic List

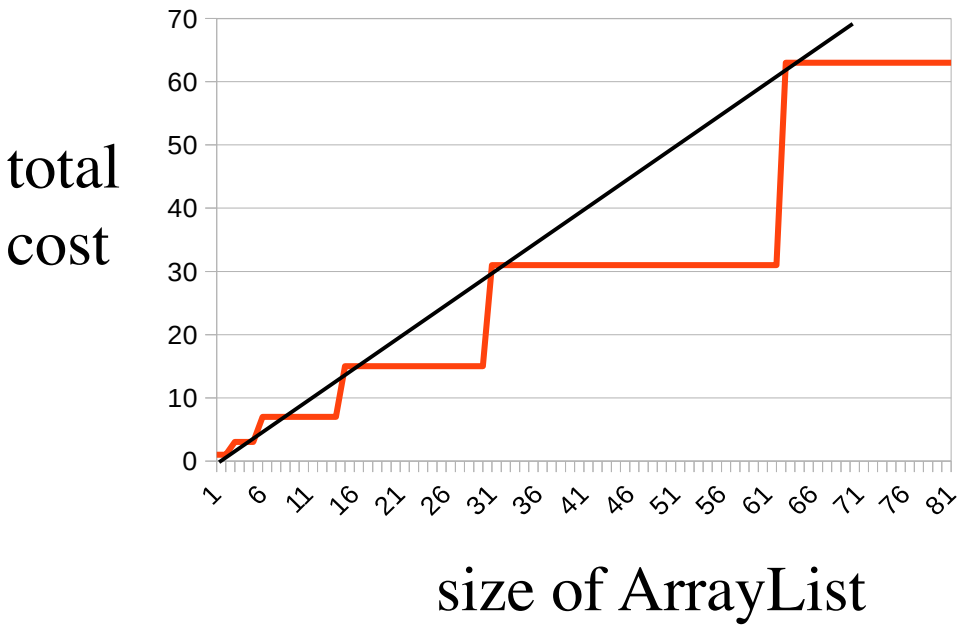
- **See LL.java**
- **Note use of wrapper class Integer**

New: ArrayLists

- **Motivation**
  - **Commonly used => part of standard library**
- **Implementation**
  - **size vs capacity**
- **operations on ArrayLists**
  - **constructor, add, add at index, get, set, remove**
  - **see DriveAL.java on Sakai**

Amortized big-O

- **Increasing size is very cheap**
- **Expanding capacity to n costs O(n)**
  - **but we do it increasingly rarely**



Total cost for n = k*n
Average cost for n
= k*n/n = k

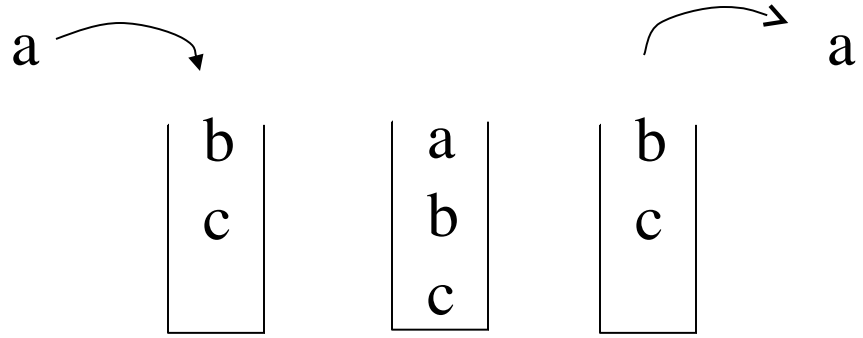
Amortized O(1) to add one element

New: Stacks

- **Motivation: Last In First Out**
  - **Metaphor: stack of trays in cafeteria**
- **Operations**
- **Example use: match parens**
- **Implementations:**
  - **ArrayList**
  - **Stack**
  - **big-Os**

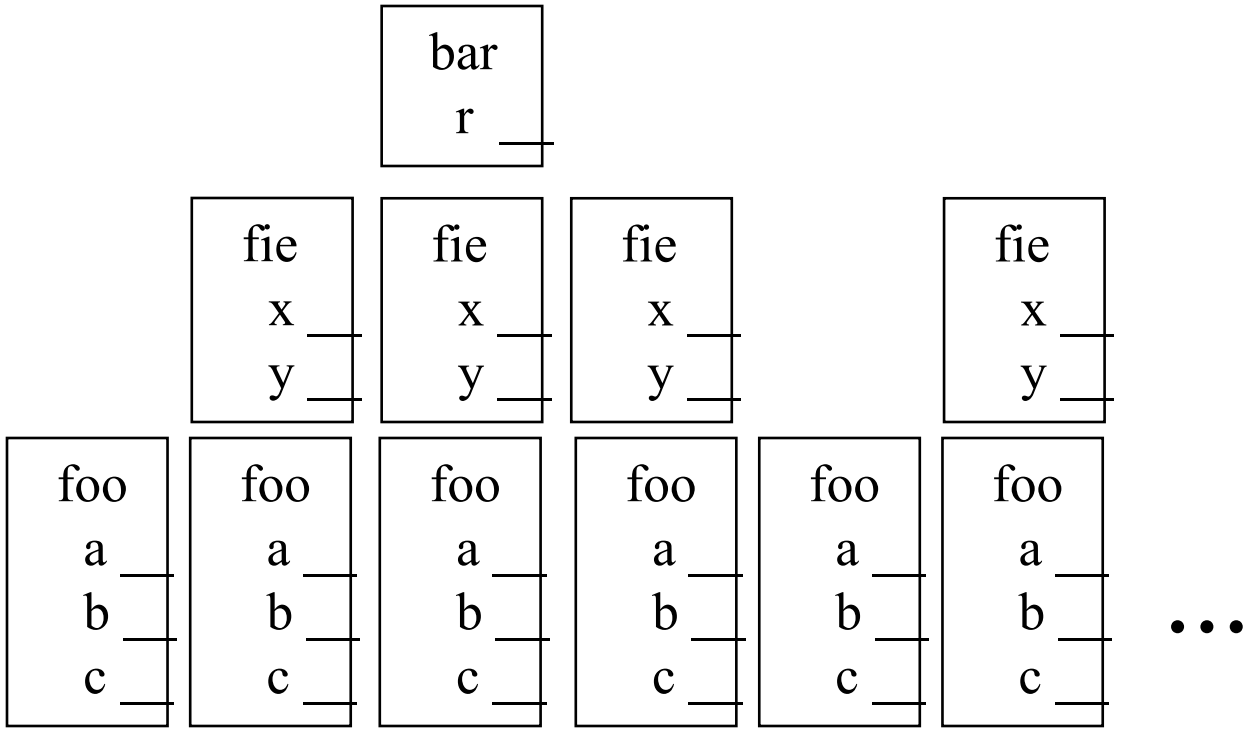
Stacks

- **Last in first out: Stack**



Stack of Invocation Records

```
public foo(int a)
  … int b, c;
  … fie(b);
  … fie(c );
public fie(int x)
  …int y;
  …bar(y);
public bar(int r)
  …
```



Operations

- **Stack**
  - **push**
  - **pop**
  - **isEmpty, size**
  - **clear**