

Lecture 31: Collections

Last time:

1. Exceptions

Today

1. Project #6 due
2. Midterm 4/18
3. Exceptions (continued)
4. Collections in Java: Stack, ArrayList



Project #6 Assigned!

- Project due Monday, 4/16 at 11 pm
- Project is **closed**
 - You must complete the project by yourself
 - Assistance can only be provided by teaching assistants (TAs) and instructors
 - You must not look at other students' code
- You should be done now!

Midterm Exam: Wed. 4/18

- Test will be given in discussion section
- Go to your own section!
- Test will be:
 - Closed notes / book / neighbor / etc.
 - Cover all material since beginning of course, with special emphasis on topics since last midterm
- Study!
 - Review notes, projects, quizzes
 - Use study questions on web-site

Exception Handling: Example

- `DateReader.java`
 - Prompts user for a date in `mm/dd/yyyy` format
 - Prints year
- Program uses:
 - `substring` method
May throw `IndexOutOfBoundsException`
 - `Integer.parseInt` method
May throw `NumberFormatException`
- How do we know about these exceptions? Javadoc!
<http://java.sun.com/j2se/1.5.0/docs/api/java/lang/package-summary.html>

Javadoc Documentation Standard



- When documenting a method, list exceptions that method can throw
 - Use `@exception` tag
 - Be sure to include unhandled exceptions that operations in method may throw

- Example:

```
/**
 * Returns the year part of a date string
 * @param d date string in mm/dd/yyyy format
 * @return an integer representing the date
 * @exception IndexOutOfBoundsException
 * @exception NumberFormatException
 */
public static int getYear(String d) {
    ...
}
```

Collections in Java

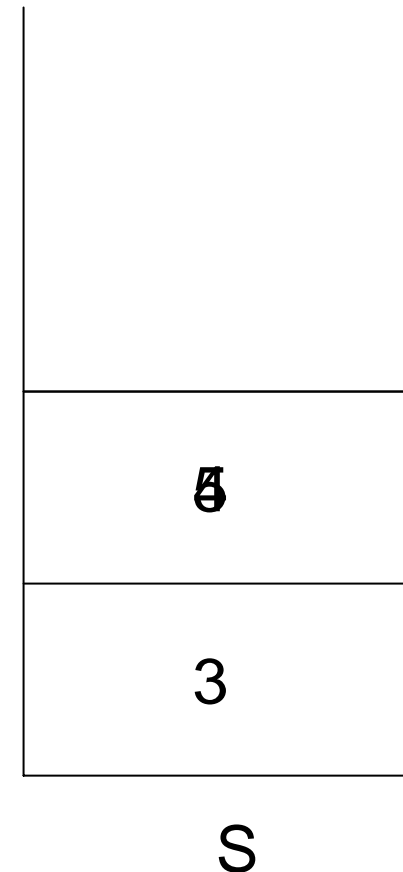
- Arrays are **collections**
 - Arrays are objects
 - Arrays are sequences of elements in base type
 - These elements are collected together in one object: the array
- Java includes many other collection mechanisms
 - Arrays good for some applications (fixed-length sequences), not others (varying-length sequences)
 - Other collections tuned for different purposes
 - General observation holds, however:
 - Collections are objects ...
 - ... that contain other objects in a given type
- We'll study two (more in 132): `Stack`, `ArrayList`

Stacks in Java

- Recall: a **stack** is a data structure (“device” for holding values)
- Three operations on a stack
 - **push**: add a new value into the stack
 - **pop**: remove the most recently added value still in stack
 - **top**: return the most recently added value in stack
Note: Java calls this “peek”
- Think: stack of plates in a restaurant
 - push = put new plate on top
 - pop = remove top plate
 - top = look at top plate (i.e., “peek”)

Example

- S.push (3);
- S.push (4);
- S.peek == ??
4
- S.pop ();
- S.push (5);
- S.peek == ??
5



Stacks in Java (cont.)

- Java includes a **generic** class for stack objects
 - Stack objects contain other objects
 - All objects in stack must have same type
 - Only objects may be stored in stacks (no primitive-type values)
- Syntax: `Stack<E>`
 - `Stack<E>` is a generic class
 - `E` is a class variable representing the base type
 - Replace `E` by a specific type to get a stack of that type of elements
 - Class is in `java.util` package
- Documentation:
<http://java.sun.com/j2se/1.5.0/docs/api/java/util/Stack.html>
- See example: `StackExample.java`

```
Stack<String> stack = new Stack<String>();
```

Creates a stack of strings

ArrayList Collection

- Like arrays ... but support for inserting/deleting new elements
 - Sequences of elements
 - All elements must be in same (base) type
- Syntax: `ArrayList<E>`
- Documentation:
<http://java.sun.com/j2se/1.5.0/docs/api/java/util/ArrayList.html>
- See example: `ArrayListExample.java`
 - `ArrayList<String> a = new ArrayList<String>();`
Creates an ArrayList of strings
 - `Collections.sort` may be used on `ArrayList<String>` objects?
 - Reason
 - `String` implements `Comparable` interface
 - `ArrayList<E>` implements `List<E>` interface