Announcements

1. HW1 posted, tutor hours start today! Expect to be a little confused, we’re here to help
2. A tour of the website
3. Post midterm/final conflicts on Piazza
4. Ways of working remotely...
Today’s Topics

1. Introduce Abstract Data Types
2. Intro to UML
   - Inheritance vs Composition
3. ADT’s, Classes, and Interfaces
4. Java Collections Framework (JCF)
public static void main( String[] args )
{
    if ( ___________________ ) {
        System.out.println( "Too few arguments" );
        return;
    }
}

What goes in the blank to make the program exit if the user does not enter at least one command line argument?
A. args[0] < 1
B. args.length < 1
C. System.in < 1
D. System.in( args ) < 1

This will help with problem 2 on the HW
For problem 1, do Friday's reading, and use Google
public static void main( String[] args )
{
    if ( args.length < 1 ) {
        System.out.println( "Too few arguments" );
        return;
    }

    A String[] strArray = new String[ args[0] ];
    B System.out.println( strArray[0] );
}

Which line above causes a compile error, A or B?
What does this code print, when called with the command line argument 42?

A. null
B. "" (the empty string)
C. 0
D. Nothing, there is a null pointer exception
E. Nothing, there is an array index out of bounds error
Abstraction

- Abstraction means:
  - Hiding irrelevant details to focus on the essential features needed to understand and use a thing
- Abstraction is an essential tool for managing complexity
  - Designing, implementing, and using complex systems would be impossible without abstraction
Abstraction example: car brakes
Group discussion

- What examples of abstraction have you experienced in CSE 8B/CSE 11?

Java
ArrayList
IDE
Critters - simulation
ADT Implementers and Users

Implementers

“We can implement the ADT however we want!”

ADT Interface:
sets the rules of interaction

Users

“We can use the ADT however we want!”
Example: a String ADT

- You might define a String abstract data type along these lines:

- Values: a String is a sequence of zero or more Unicode 1.0 characters

- Operations:
  - Create a String
  - Add a character to the end of a String
  - Say what character is at a particular position in a String
  - ...

- What other operations might be good to specify for a String ADT?
ADTs are language-neutral

- In CSE 12 we will concentrate on implementing ADT’s in Java
- And we will consider many features of Java that are useful for implementing ADT’s
- But always keep in mind that the basic principles of ADT design and analysis are language-neutral!

APIs are like ADTs, but specify the interface in a particular language

- Users and implementers still do things as they please, but both within a common language
Which of the following would be elements of an **ADT** specification, and which would be elements of an **API** specification?

I. `int getValueAtIndex(int index) throws IndexOutOfBoundsException`

II. Get operation: given an index, returns the value at that index

**A.** I is part of an **API** and II is part of an **ADT**

**B.** II is part of an **API** and I is part of an **ADT**

**C.** Both I and II could be part of an **API** or **ADT**

**D.** Other
Inheritance vs. Composition

Design patterns using UML
Data, data, everywhere
Presents, presents everywhere
Collections

- A collection is an ADT that contains data elements, and provides operations on them.

There are different ways that elements can be “collected”:
- Set
- List
- Sorted list
- ...

All collections implement the interface Collection.
Java Generics

- Key to Java’s Collections are **generics**
  - Generics answer the question: “**what is the collection a collection of?**”
  - If this question isn’t enforced with generics, there can be runtime errors when the what comes out of the collection is not what you expected
  - `Set<E>` means a set of things that are of type E.
Which choice is the most reasonable model of our bag of presents?

A. `public class Present<Bag> implements Collection<Bag>`

B. `public class Bag<Collection> implements Present<Collection>`

C. `public class Present<Collection> implements Bag<Collection>`

D. `public class Bag<Present> implements Collection<Present>`

E. Other/none/more than one
Placing limits on what kind of thing our collection can store

- This is a kid, so we ONLY want the bag to hold presents that are:
  - **Wrapable** (can be wrapped in paper)
    - Implements a “wrap()” method
  - **Playable** (can be played with)
    - Implements a “play()” method

- How can we represent these requirements in Java?
Java Interfaces
ADTs in the Java language
Actual Google interview question (given to a UCSD student)

- Given an **interface** called Set with the functions union, intersect and subtract, implement that interface in a class.
Actual Google interview question (given to a UCSD student)

- Given an **interface** called Set with the functions union, intersect and subtract, implement that interface in a class.

- **What would our implementation start with?**
  
  A. public class Set<E> {
  
  B. public class MySet<E> extends Set<E> {
  
  C. public class FancySet<E> implements Set<E> {
  
  D. Other/none/more than one