

CSSE 220 Day 2

Class, Objects, and Methods in Java
UML Class Diagram Basics

Your questions about ...

- ▶ The syllabus
 - ▶ Java
 - ▶ etc.
-
- ▶ Could everyone checkout and commit the HW1 project?

Announcements

- ▶ Please do not sit in the back row or on the far right side of the room.
- ▶ Please consider making your picture on ANGEL visible to students in your courses.
 - ◻ Home → Preferences (wrench icon) → Personal info
- ▶ **If you want all of your ANGEL mail to also go to your regular mail, too, you can set it that way.**
 - Home → Preferences → System Settings
- ▶ You can subscribe to the ANGEL discussion forums. (From the Communicate menu)

More announcements

▶ Cell Phones

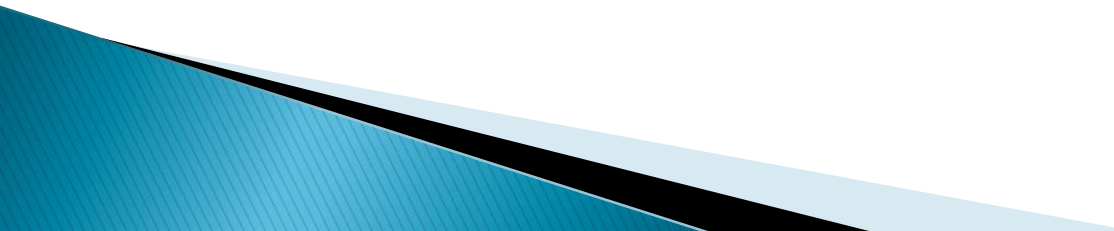
- please set ringers to silent or quiet.
 - Minimize class disruptions.
 - But sometimes there are emergencies.

▶ Personal needs

- If you need to leave class for a drink of water, a trip to the bathroom, or anything like that, you need not ask me. Just try to minimize disruptions.

- ▶ Please be here and have your computer up and running by class time as best you can.

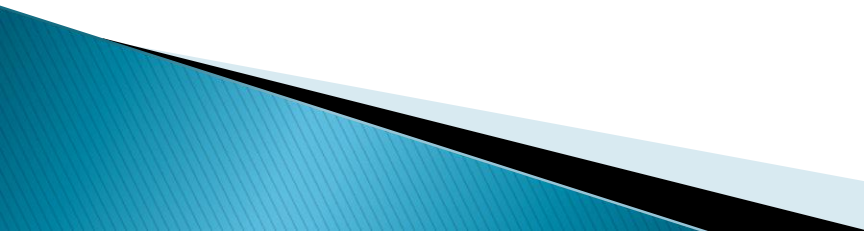
Bonus points for reporting bugs

- ▶ In the textbook
 - ▶ In any of my materials.
 - ▶ Use the Bug Report Forum on ANGEL
 - ▶ More details in the Syllabus
-
- ▶ Subscribe to the discussion forums on ANGEL
- 

Some major emphases of 220

- ▶ ***Reinforce from 120:***
 - Procedural programming (functions, conditionals, loops, etc)
 - Using objects
- ▶ ***Object-Oriented Design***
 - Major emphasis on interfaces
 - GUI programming using Java Swing
 - UML class diagrams
- ▶ ***Software Engineering concepts***
- ▶ ***Data Structures***
 - Introduce algorithm efficiency analysis
 - Abstract data types
 - Specifying and using standard data structures
 - Implementing simple data structures (lists)
- ▶ ***Recursion***
- ▶ ***Sorting and searching algorithms***
 - as examples for the above

What will I spend my time doing?

- ▶ Small programming assignments in class
 - ▶ Larger programming problems, mostly outside of class
 - Exploring the JDK documentation to find the classes and methods that you need
 - Debugging!
 - Reviewing other students' code
 - ▶ Reading (a lot to read at the beginning; less later)
 - Thinking about exercises in the textbooks
 - Some written exercises, mostly from the textbook
 - ▶ Discussing the material with other students
- 

Acrobat Games: An introduction to object-oriented design

▶ We need

11 volunteers

- BasicAcrobat x 3
- ProudAcrobat x 2
- DoublingAcrobat
- AcrobatWithBuddy x 2
- Choreographer x 2
- Curmudgeon

This exercise is adapted from Joe Bergin's page at <http://www.csis.pace.edu/~bergin/Java/RolePlay.html> that describes an idea presented by Steven K. Andrianoff and David B. Levine at SIGCSE-2002

Each volunteer gets:

- 1 White sheet with instructions for how to behave
- 1 Yellow sheet (blank): scratch pad
- 3 Red sheets (blank): for returning data

▶ Instructor:

- Write the 6 types on the whiteboard
- Beside each type, write the names of the actors for that type
 - Or just refer to them as basicAcrobat1, etc.
- Announce instructions per the next slide

Acrobat Games (continued)

The instructor announces these commands (using students' names or *basicAcrobat1*, etc). When paper returns to the instructor after a *count* command, she will announce what is on the paper.

- ▶ `basicAcrobat1.clap(2);`
- ▶ `basicAcrobat2.twirl(1);`
- ▶ `basicAcrobat1.twirl(2);`
- ▶ `basicAcrobat1.count();`
- ▶ `basicAcrobat3.count();`
- ▶ `basicAcrobat3.sing();`

- ▶ `proudAcrobat1.clap(3);`
- ▶ `proudAcrobat2.twirl(1);`
- ▶ `proudAcrobat1.bow();`

- ▶ `curmudgeon.twirl(3);`
- ▶ `curmudgeon.clap(3);`

- ▶ `acrobatWithBuddy1.clap(4);`
- ▶ `acrobatWithBuddy1.clap(2);`
- ▶ `acrobatWithBuddy1.twirl(1);`
- ▶ `acrobatWithBuddy1.nameBuddy();`

- `acrobatWithBuddy2.nameBuddy();`
- `acrobatWithBuddy2.clap(3);`
- `acrobatWithBuddy2.jump(4);`
- `acrobatWithBuddy2.count();`

- `acrobatWithBuddy1.twirl(1);`
- `acrobatWithBuddy1.nameBuddy();`
- `acrobatWithBuddy2.nameBuddy();`

- `doublingAcrobat.clap(3);`
- `doublingAcrobat.count();`
- `doublingAcrobat.twirl(100);`
// just kidding!
- `choreographer1.clap(3);`
- `choreographer1.clap(3);`
- `choreographer2.clap(2);`
- `choreographer1.count();`

- `basicAcrobat1.count();`

Acrobat Games (continued)

- ▶ Consider the following command:
 - `acrobatWithBuddy1.clap(2);`
- ▶ When
 - `acrobatWithBuddy1`'s buddy is `acrobatWithBuddy2`
 - `acrobatWithBuddy2`'s buddy is `acrobatWithBuddy1`
- ▶ When
 - `acrobatWithBuddy1`'s buddy is herself

This idea has a name – *recursion!*

Instructions for the actors

- ▶ The next six slides give the instructions that tell each actor how to behave.
- ▶ There is a single set of instructions for each type of actor:
 - BasicAcrobat
 - ProudAcrobat
 - DoublingAcrobat
 - AcrobatWithBuddy
 - Choreographer
 - Curmudgeon

You are a BasicAcrobat

If you are asked to do anything else, say (as dramatically as you can) “I refuse”.

When you are asked to:

- ▶ *clap*, you will be given a number.
Clap your hands that many times.
- ▶ *twirl*, you will be given a number.
Turn completely around that many times.
- ▶ *count*, write on your Red paper how many actions (claps and twirls) you have performed so far (reading that number from your Yellow paper).
 - For example, after *clap 2* and *twirl 1* you would write *3* on the paper.

On your yellow scratch pad, keep track of how many actions you have performed so far.

Give that Red paper to the person who asked you to count.

You are a Proud Acrobat

When you are asked to:

- ▶ *clap*, you will be given a number.

Clap your hands that many times.

Say “Thank you.” Then take a bow (as dramatically as you like).

- ▶ *twirl*, you will be given a number.

Turn completely around that many times.

Say “Thank you.” Then take a bow (as dramatically as you like).

- ▶ *bow*,

Say “Thank you.” Then take a bow (as dramatically as you like).

- ▶ *count*, write on your Red paper how many actions (claps and twirls and bows) you have performed so far (reading that number from your Yellow paper).

- For example, after *clap 2* and *twirl 1* you would write *3* on the paper.

Give that Red paper to the person who asked you to count.

Say “Thank you.” Then take a bow (as dramatically as you like).

If you are asked to do anything else, say (as dramatically as you can) “I refuse”.

On your yellow scratch pad, keep track of how many actions you have performed so far.

You are a Doubling Acrobat

If you are asked to do anything else, say (as dramatically as you can) “I refuse”.

When you are asked to:

- ▶ *clap*, you will be given a number.

Clap your hands *twice* that many times.

- ▶ For example, if you are told to *clap 3* then you should clap *6* times.

- ▶ *twirl*, you will be given a number.

Turn completely around *twice* that many times.

- ▶ *count*, write on your Red paper how many actions (claps and twirls) you have performed so far (reading that number from your Yellow paper).

- For example, after *twirl 2* and *clap 3* you would have twirled 4 times and clapped 6 times, so you would write *10* on the paper.

Give that Red paper to the person who asked you to count.

On your yellow scratch pad, keep track of how many actions you have performed so far.

You are an AcrobatWithBuddy

When you are asked to:

- ▶ *clap*, you will be given a number.

Pass the same instruction to your Buddy.
Then clap your hands that many times.

- ▶ *twirl*, you will be given a number.

Pass the same instruction to your Buddy.
Then turn completely around that many times.

- ▶ *nameYourBuddy*, say (loudly) the name of your Buddy

- ▶ *count*, first ask your Buddy to count. Your Buddy should then give you a Red paper with a number written on it. Add that number to the number of actions (claps and twirls) you have performed so far, and write that sum on your Red paper.
 - For example, after *clap 2* and *twirl 1* your own count would be *3*. So if your Buddy gives you Red paper with (say) 7 on it, you would write 10 on your Red paper.

Give your own Red paper to the person who asked you to count.

Throw away the piece of paper from your Buddy.

When you are given this card, before we start the role play, you should (mentally) choose another actor (anyone except a Curmudgeon) to be your Buddy. That person will be your Buddy for the rest of the exercise.

If you are asked to do anything else, say (as dramatically as you can) "I refuse".

On your yellow scratch pad, keep track of how many actions you have performed so far.

You are a Choreographer

When you are given any instruction, such as twirl, clap or count, pass it on to two other actors.

- For example, if you are told to **clap 3** then you might respond by saying **John, clap 3** and when John has finished, saying **Mary, clap 3** assuming that John and Mary are names of two of the actors. You should not clap.
- Do not directly refuse any command. However, if either of your two actors says “I refuse,” then you say “I refuse.”
- If the command is **count**:
 - Both of your actors will eventually hand you a Red piece of paper (unless they refuse, in which case you refuse).
 - Add the two numbers from their papers and write the sum on your own Red paper.
 - Then give your own Red paper to the person who asked you to count.

Pick your two actors at random each time (but never pick a Curmudgeon).

You can pick the same actor twice (instead of two different actors) or you can even pick yourself as one (or both) of the actors. Try these!

But don't do these fancy tricks the first time a Choreographer is given an instruction.

You are a Curmudgeon

- ▶ When given any instruction (such as **twirl**, **clap**, or **count**), ignore it. Stand up, cross your arms over your chest, smirk, and say (as smugly and dramatically as you can) **“I refuse.”**

Then sit down again if you were originally sitting.

Acrobat Games – Debriefing:

Classes and *Objects*

- ▶ What are the names of some *classes* represented in Acrobat Games?
 - See below
- ▶ What are the names of some *objects* ?
 - *john*, *mary*, ... (names of the actors in your classroom)
 - Or perhaps you think of them as named by numbers:
 - *basicAcrobat1*, *basicAcrobat2*, *basicAcrobat3*,
proudAcrobat1, *proudAcrobat2* etc.
- ▶ Can there be more than one object from the same class?
 - Yes!

BasicAcrobat

Choreographer

Curmudgeon

ProudAcrobat

DoublingAcrobat

AcrobatWithBuddy

Acrobat Games – Debriefing:

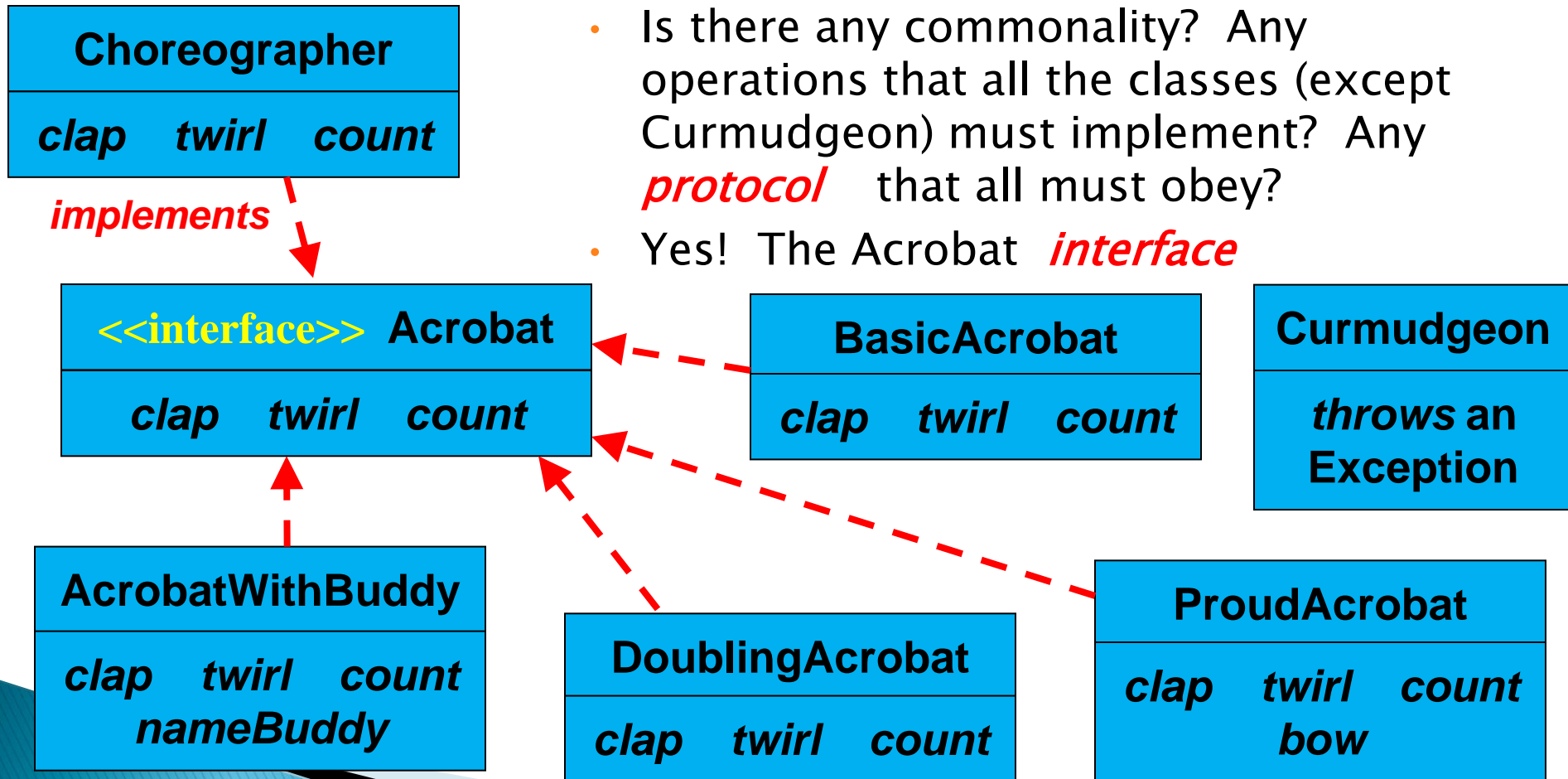
Operations

- ▶ What are the names of some *operations* that the objects in Acrobat Games can do?
 - *clap*, *twirl*, *count*, *bow*, *nameBuddy*
 - We call these operations *methods* in Java
- ▶ Are all objects able to do the same operations?
 - No. For example, only an *AcrobatWithBuddy* can *nameBuddy*
- ▶ Are all objects of the same class able to do the same operations?
 - Yes. For example, all *ProudAcrobats* can *bow*

Acrobat Games – Debriefing:

Interfaces

- ▶ For each type of object, what operations can that object do?



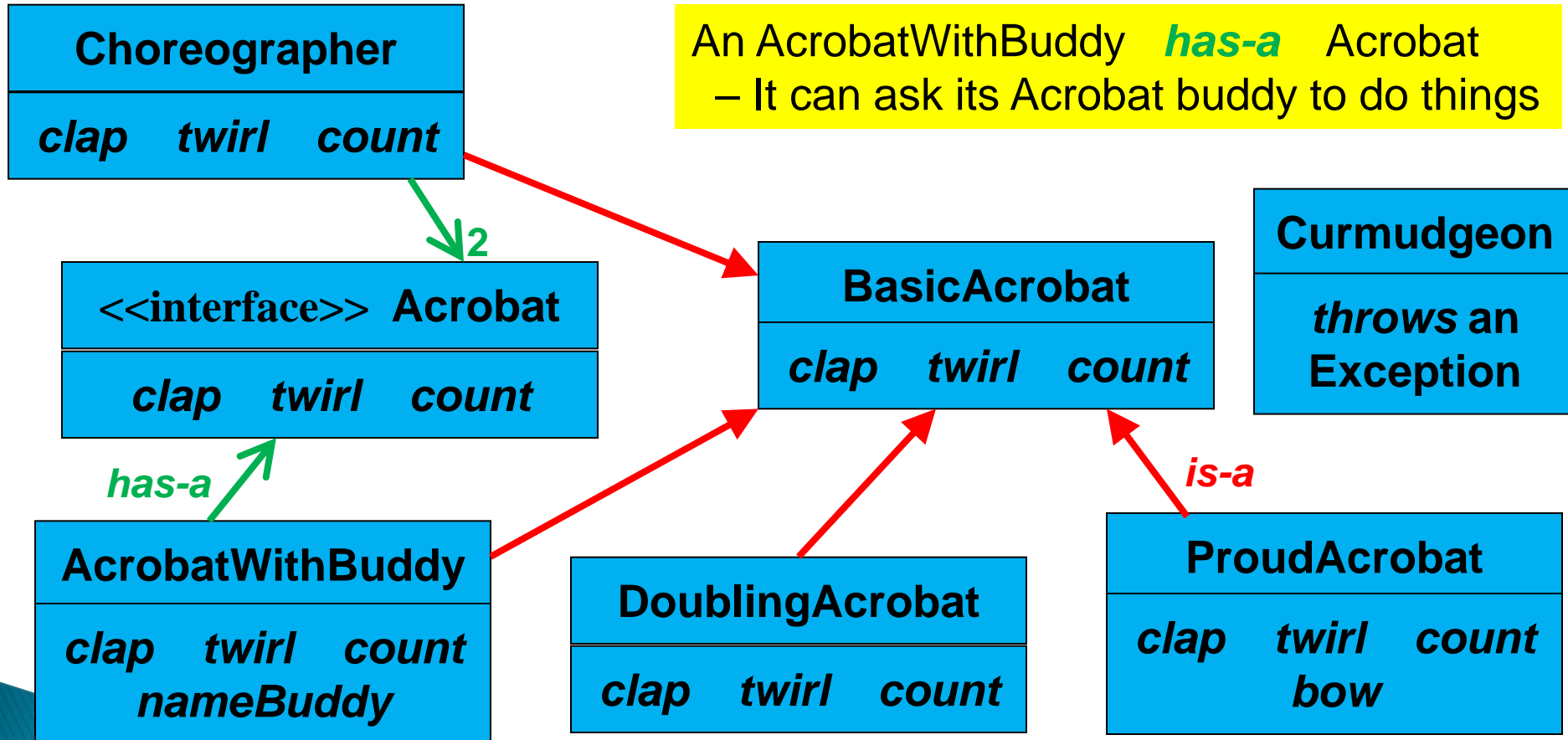
- Is there any commonality? Any operations that all the classes (except Curmudgeon) must implement? Any **protocol** that all must obey?
- Yes! The Acrobat **interface**

Acrobat Games - Debriefing

is-a and *has-a*

A ProudAcrobat *is-a* BasicAcrobat
– It *inherits* all the attributes and operations of a BasicAcrobat

An AcrobatWithBuddy *has-a* Acrobat
– It can ask its Acrobat buddy to do things

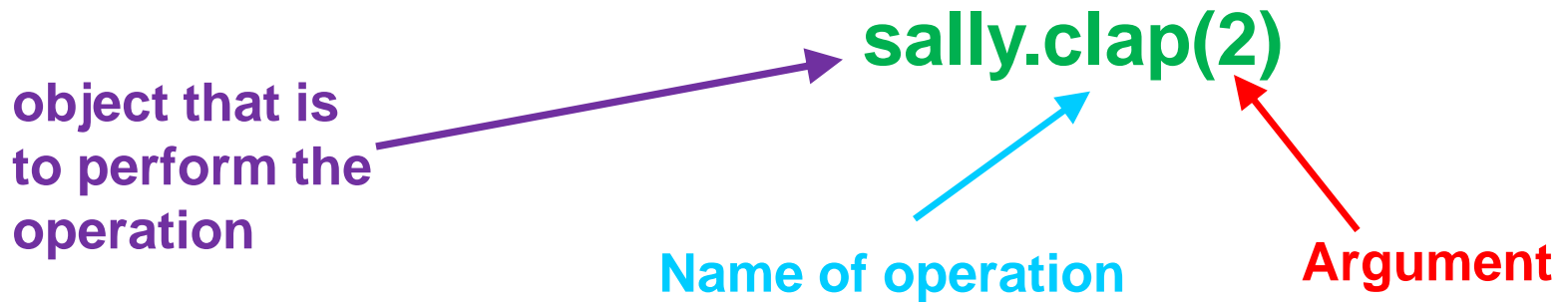


What other *is-a* relationships exist above? *has-a*?

Acrobat Games – Debriefing:

arguments and *returned values*

- ▶ When asking an object to perform an operation, we say three things. What are they?



- What determines how a particular object performs the `clap` operation?
 - Answer: What class it is an instance of. For example, a `BasicAcrobat` claps one way, while a `ProudAcrobat` claps another way. Also, what argument it is given. `sally.clap(2)` claps twice, while `sally.clap(6)` claps six times.
- Do objects have to act alone when performing?
 - No. For example, a `Choreographer` asks others to act on its behalf.
- How are the `clap` and `count` operations fundamentally different?
 - Answer: `clap` does something, while `count` returns a value.

Acrobat Games – Summary

- ▶ In Acrobat Games, we saw many of the ideas of Object–Oriented Programming (OOP), including:
 - **Classes** (`BasicAcrobat`, `ProudAcrobat`, ...)
 - **Objects** (`proudAcrobat1`, `proudAcrobat2`, ... – **instances** of the classes)
 - **Methods** (the operations `clap`, `twirl`, `count`, ...)
 - **Arguments:** `clap(2)`
 - **Returned values:** from `count`
 - **Encapsulation** in classes and in methods
 - Internal **state** (each object keeps track of its count)
 - **Inheritance** (e.g., `ProudAcrobat` **is-a** `BasicAcrobat`)
 - **Associations** (e.g., an `AcrobatWithBuddy` **has-a** `Acrobat`)
 - Implementing to an **interface** (... implements the `Acrobat` interface)
 - **Exceptions** (e.g., from `Curmudgeons`)
 - **UML class diagrams** (that show relationships between classes)
 - **Recursion** (when an `AcrobatWithBuddy` has itself as its Buddy)

Interlude – From worldmag.com

WORLD
MAGAZINE

Archives *1996 to the Present*

Quick Takes

ODDBALL OCCURRENCES

Bovine force

» If Linda and Charles Everson Jr. had been driving just a bit faster, they may not have celebrated another anniversary. While celebrating their first anniversary, the Michigan couple was driving on Highway 150 alongside a cliff near Manson, Wash., when something fell from above and crushed the hood of their minivan. Instead of falling rocks, it was a falling cow. The 600-pound cow, which had fallen from 200 feet up, crushed the front of their minivan, but the couple escaped unscathed. The 1-year-old bovine wasn't so fortunate.

wenatchee**w**orld.com
THE FIERCELY INDEPENDENT VOICE OF NORTH CENTRAL WASHINGTON

The cow heard around the world

Media milking falling-heifer story to death

The couple were driving back to their Manson hotel on Highway 150 Sunday after attending a church service in Chelan when the 600-pound heifer named Michelle dropped from above and landed on the hood of their Buick Terraza and bounced off onto the road. Everson said he was stunned and kept on driving, repeating to himself "I don't believe it. I don't believe it."



Identifiers (Names) in Java

- ▶ The rules:
 - Start with letter or underscore (_)
 - Followed by letters, numbers, or underscores
- ▶ The conventions:
 - `variableNamesLikeThis`
 - `methodNameLikeThis (...)`
 - `ClassNameLikeThis`

Variables in Java

- ▶ Like C:

- `int xCoordinate = 10;`

- ▶ But Java catches some mistakes:

```
int width, height, area;  
area = width * height;
```



What does this do in C?

- Java will detect that `width` and `height` aren't initialized!

Using Objects and Methods

- ▶ Works just like Python:

- `object.method(argument, ...)`

Implicit
argument

Explicit
arguments

“Who does What,
With What?”

- ▶ Java Example:

```
String name = "Bob Forapples";  
PrintStream printer = System.out;
```

```
int nameLen = name.length();  
printer.printf("'%s' has %d characters", name, nameLen);
```

The dot notation is
also used for *fields*

Separating Use from Implementation

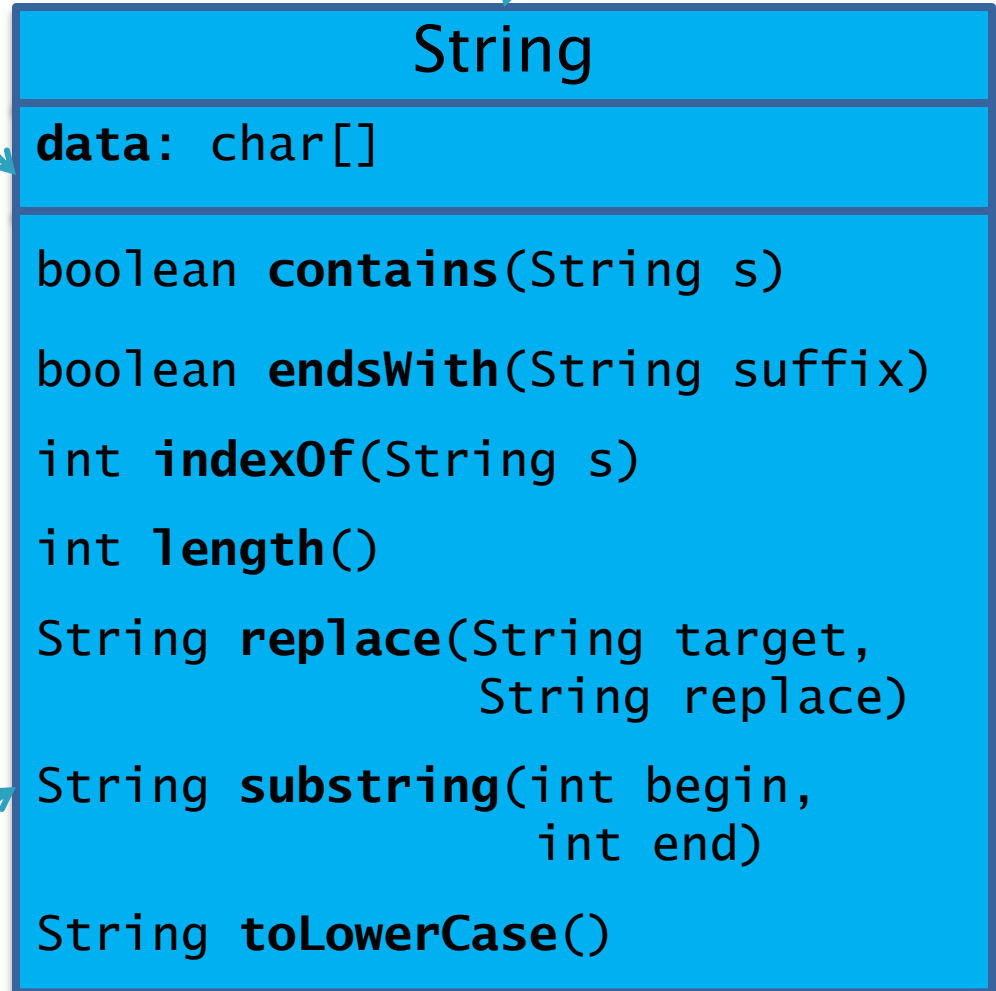
- ▶ Can use methods of an object without knowing how they are implemented
 - Recall zellegraphics from csse 120:
`line.setWidth(5)`

UML Class Diagram

- ▶ Shows the:
 - **Attributes** (data, called **fields** in Java) and
 - **Operations** (functions, called **methods** in Java) of the objects of a class
- ▶ Does *not* show the implementation
- ▶ Is *not* necessarily complete

Fields

Class name



Methods

String methods are *immutable* - if the method produces a String, the method *returns* that String rather than mutating (changing) the implicit argument

Exercise

- »» Checkout ObjectsAndMethods from SVN
Work on UsingStrings.java

Passing Parameters

- ▶ Arguments can be any expression of the “right” type
 - See example...
- ▶ What happens if we try to give `substring()` an explicit argument that isn't a number?
 - How does the compiler know that `rhit.length()` evaluates to a number?
 - What's the return type of `length()`?
- ▶ Static types help the compiler catch bugs.
 - Important in large programs

```
String rhit = "Rose-Hulman";
System.out.println("Rose");
System.out.println(rhit.substring(0, 4));
System.out.println(rhit.substring(0, 2+2));
System.out.println(rhit.substring(0, rhit.length() - 7));
System.out.println("Rose-Hulman".substring(0, 4));
```

Primitive types

Primitive Type	What It Stores	Range
byte	8-bit integer	-128 to 127
short	16-bit integer	-32,768 to 32,767
int	32-bit integer	-2,147,483,648 to 2,147,483,647
long	64-bit integer	-2^{63} to $2^{63} - 1$
float	32-bit floating-point	6 significant digits (10^{-46} , 10^{38})
double	64-bit floating-point	15 significant digits (10^{-324} , 10^{308})
char	Unicode character	
boolean	Boolean variable	false and true

figure 1.2

The eight primitive types in Java

Most common
number types in
Java code

Exercise

»» Work on SomeTypes.java

Constructing Objects

left, top, width, height

▶ Example:

```
Rectangle box = new Rectangle(5, 10, 20, 30);
```



▶ Several steps are happening here:

1. Java reserves space for a *Rectangle* object
2. *Rectangle*'s *constructor* runs, filling in slots in object
3. Java reserves a variable named *box*
4. *box* is set to refer to the object

Accessors and Mutators

▶ *Accessor* methods

- Get a value from an object
- Examples:
 - `box.getHeight()`
 - `box.getWidth()`

▶ *Mutator* methods

- Change the *state* of an object (i.e., the value of one or more fields)
- Examples:
 - `box.translate(10, 20)`
 - `box.setSize(5, 5)`

Tip: Use mutators with care!

Reminder: In all your code:

- ▶ **Write appropriate comments:**
 - Javadoc comments for public fields and methods.
 - Explanations of anything else that is not obvious.
- ▶ **Give self-documenting variable and method names:**
 - Use name completion in Eclipse, Ctrl-Space, to keep typing cost low and readability high.
- ▶ **Use Ctrl-Shift-F in Eclipse to format your code.**
- ▶ **Take care of all auto-generated TODO's.**
 - Then delete the TODO comment.
- ▶ **Correct ALL compiler warnings.**
 - Quick Fix is your friend!



Exercise

- »» Finish quiz
- Continue working on homework