

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

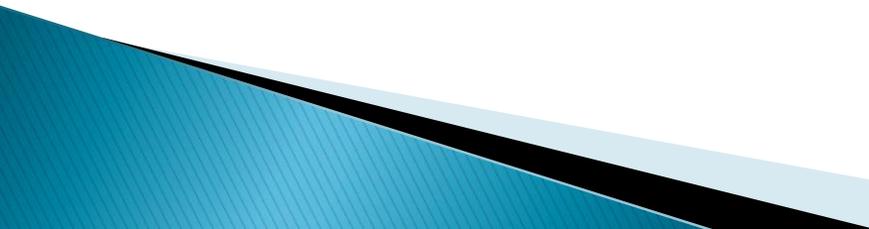
- please set ringers to silent or quiet.
 - Minimize class disruptions.
 - But sometimes there are emergencies.

▶ 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.

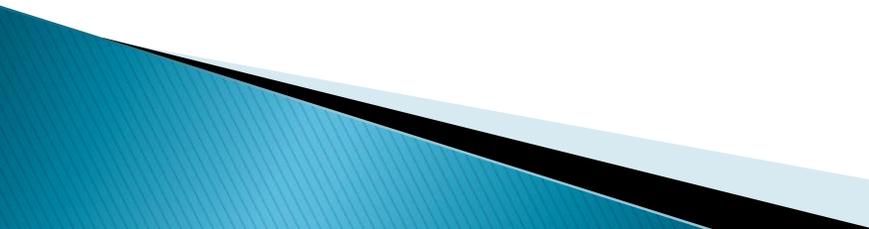
Bonus points for reporting bugs

- ▶ In the textbook
 - ▶ In any of my materials.
 - ▶ Use the Bug Report Forum on ANGEL
 - ▶ More details in the Syllabus
 - ▶ New edition of the text, so I'll probably mess up some page numbers...
 - ▶ 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
- 

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

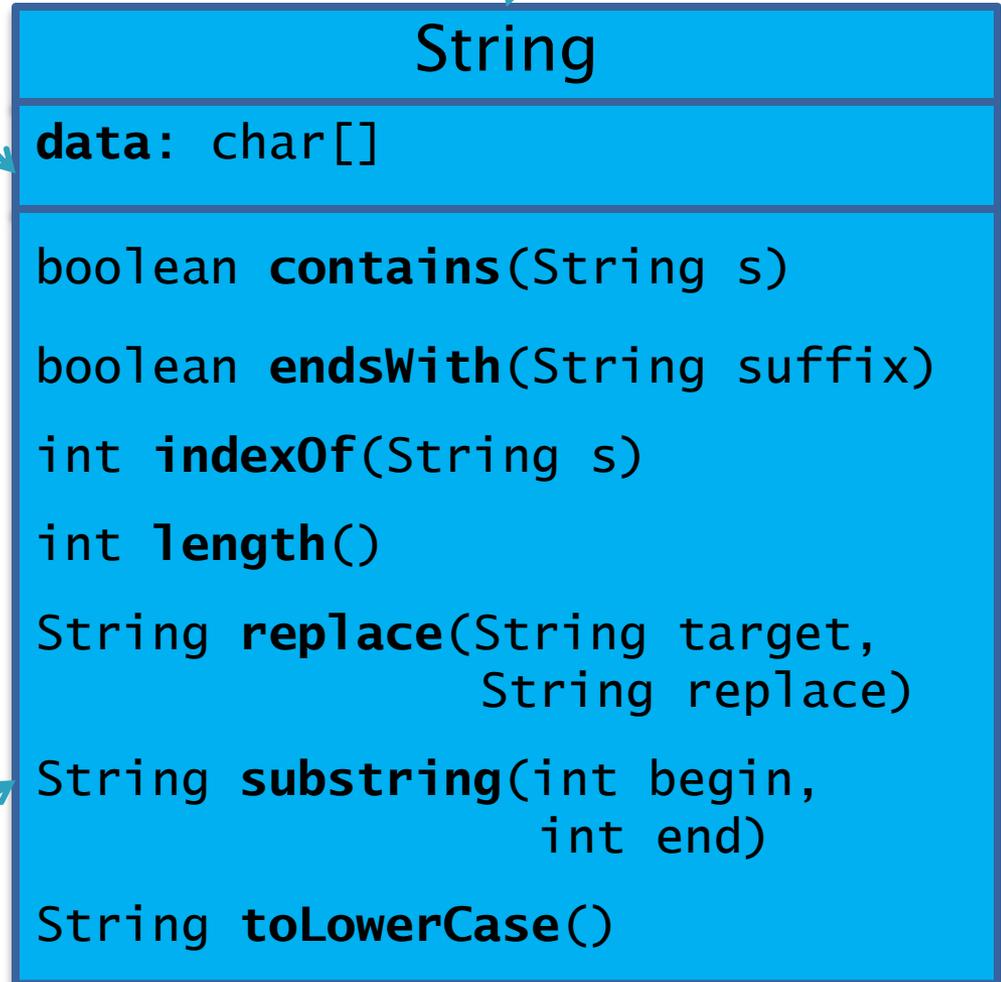
- ▶ Can use methods of an object without knowing how they are implemented
 - Recall zellegraphics from csse 120:
`line.setWidth(5)`

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()`

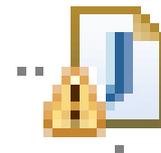
▶ *Mutator* methods

- 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 and pass it in
Continue working on
homework