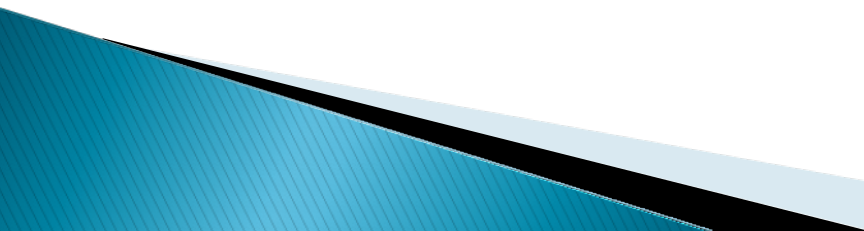


CSSE 220 Day 2

More course and Instructor Intro
Class, Objects, and Methods in Java
UML Class Diagram Basics

Your questions about ...

- ▶ Syllabus
 - ▶ Wiki
 - ▶ Eclipse
 - ▶ Java
 - ▶ etc.
 - ▶ My questions for you:
 - Did editing the Wiki work well for everyone?
 - Could everyone checkout and commit the HW1 project?
- 

Instructor Intro

On separate slides

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 the time the class period starts.

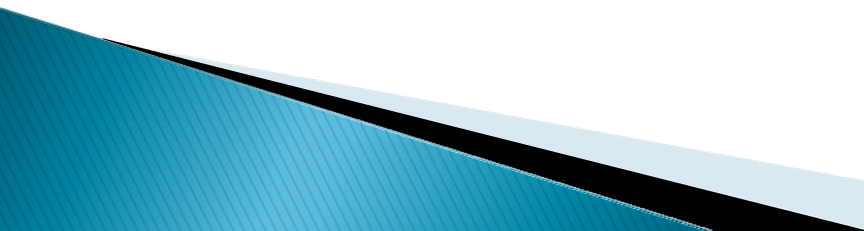
- Except perhaps for the four of you who have 2nd hour classes in Crapo.

Bonus points for reporting bugs

- ▶ In the textbook
- ▶ In any of my materials
- ▶ Use the Bug Report discussion forum on ANGEL
- ▶ More details in the Syllabus

- ▶ **Suggestions:**
 - Subscribe to the discussion forums on ANGEL.
 - Make your picture on ANGEL visible to other students in the class.

Some major emphases of 220

- ▶ Reinforce and extend OO ideas from 120
 - Major emphasis on inheritance
 - GUI programming using Java Swing
 - ▶ Object-oriented Design
 - ▶ Data Structures
 - Introduce algorithm efficiency analysis (big O)
 - Abstract Data Types
 - Specifying and using standard data structures
 - Implementing simple data structures (lists)
 - ▶ Recursion
 - ▶ Simple sorting and searching algorithms
 - ▶ A few additional Software Engineering concepts
- 

What will you spend your time doing for this course?

- ▶ 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 textbook
 - Some written exercises, mostly from the textbook
- ▶ Ideally, discussing the material with other students in the course
 - ▶ Study groups are an excellent idea

In all your code:

- ▶ Write appropriate comments:
 - Javadoc comments for public fields and methods.
 - Explanations of anything else that is not obvious.
- ▶ Give explanatory 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.

Identifiers (Names) in Java

- ▶ The rules:
 - Start with letter or underscore (_)
 - Followed by letters, numbers, or underscores
- ▶ The conventions:
 - variableNamesLikeThis
 - methodNamesLikeThis()
 - ClassNamesLikeThis
 - CONSTANT_NAMES_LIKE_THIS

Variables in Java

- ▶ Like C:

- `int xCoordinate = 10;`

- ▶ But Java catches some mistakes:

- `int yPosition;`

- `printf("%d", yPosition);`



What does this do in C?

- ▶ Java will detect that `yPosition` isn't initialized!

Some Terminology

- ▶ Spend 2 minutes talking with one or two nearby students. Try to come up with working definitions of the following terms:
 - class
 - object
 - instance
 - method (how is a method different than a function?)
 - field (a.k.a. instance variable)
 - constructor
- ▶ An object _____ things and can __ things.

Using Objects and Methods

- ▶ Works just like Python:

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

Implicit
argument



Explicit
arguments



- ▶ Java Example:

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

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

Separating Use from Implementation

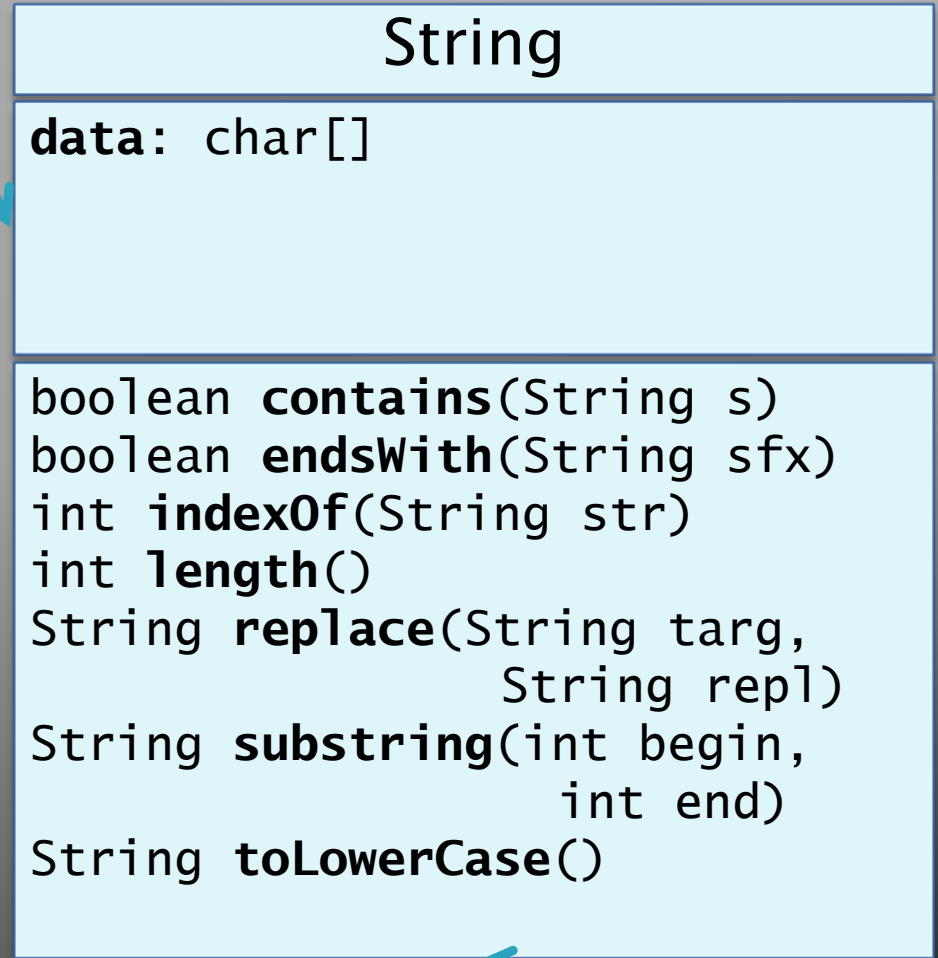
- ▶ Can use methods of an object without knowing how its implemented
 - Recall zellegraphics from 120: `Line.setWidth(5)`

UML Class Diagram

- ▶ Shows the data (fields) and operations (methods) of the objects of a class
- ▶ Does *not* show the implementation
- ▶ Not necessarily complete

Fields

Class name



For all of the details on String methods, go to

<http://java.sun.com/java/se/6/docs/api/java/lang/String.html>

Methods

Java Strings are immutable!

Quick 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 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

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

primitives vs. objects
boolean is special
char is an integer type
String and *char* are different

Most common number types in Java code

Q5,6

Quick 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)`

Exercise

Finish quiz

Continue working on homework