

CSCI 136:
Data Structures
and
Advanced Programming
Lecture 4
Exceptions and classes

Instructor: Dan Barowy
Williams

Topics

- Study tip: growth mindset
- Exceptions
- Classes and objects

Study tip #1: **growth mindset**

Have you ever thought:
“I’m not good at [x]”



Study tip #1: **growth mindset**

If you are motivated and study effectively,
there is nothing you cannot learn.

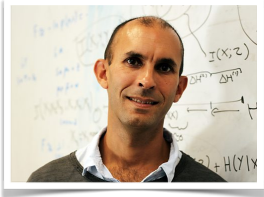
In fact, **you learn whether you want to or not.**

Proof (demo). Again. Ungarbled. One more time.

Notice that you can understand the garbled sound!

Study tip #1: **growth mindset**

Every brain is an **amazing learning machine**.



Anil Seth,
Professor of Cognitive and
Computational Neuroscience,
University of Sussex

Your brain is capable of rewiring
itself in **milliseconds**.

Learning how to use your brain is
a **skill** that requires **practice**!

Your to-dos

1. Lab 1, **due Tuesday 2/15 by 10pm**.
2. Read **before Mon**: Bailey, Ch 3-3.1 & Ch 4-4.2.2.
Suggestion: read *actively*.

Announcements

- Colloquium **today @ 2:35pm in Wege** (TCL 123):
Senior Thesis Proposals



Nim

- Game starts with **random** piles.
- Each player removes **one or more** objects from **ONE** pile.
- The last player to remove the **last object wins**.

Exceptions

Exceptions

A **software exception** is a mechanism **for signaling errors**. When an exception is **thrown** in a program, the program will cease running ("crash") unless the program **catches** and **handles** the error.

More precisely, an uncaught exception unwinds the call stack until a matching exception handler is found. If an exception handler is not found, the program halts, printing a stack trace.

Example in Nim using Scanner

Java is Object-Oriented

- OO is a **system** for writing code that has properties **highly valued** by software engineers.
- Those properties are:
 - **Code reuse**
 - **Modularity**
 - **Data abstraction**
- It is sometimes said (**incorrectly**) that OO is about “modeling the real world.”
- OO is a very big topic, and it takes awhile to master all the pieces.
- For now, we are going to focus on **data abstraction**.

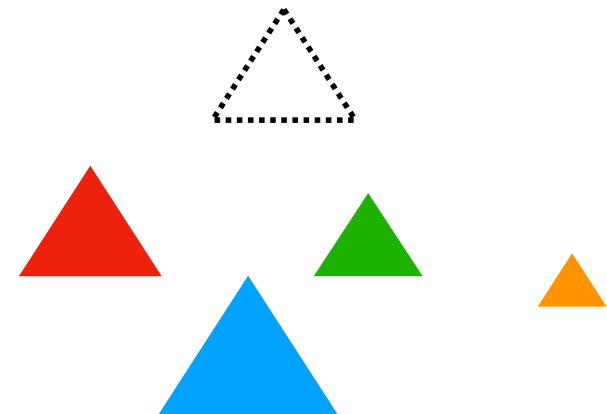
If you don't understand all these words just yet, **don't worry**.

Classes and objects

Classes

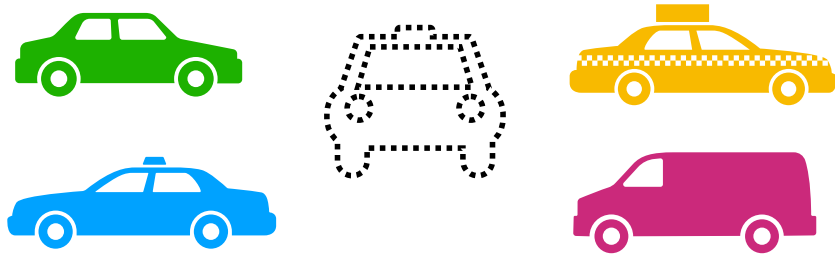
A **class** is a mechanism for **data abstraction**. The purpose of a class is to separate the details that are important to the programmer (the **interface**) from the details that are important to the computer (the **implementation**). *Classes are a key building block in designing data structures.*

Classes are **prototypes**.
Objects are **copies** (“instances”).



“Car” is a **prototype**.

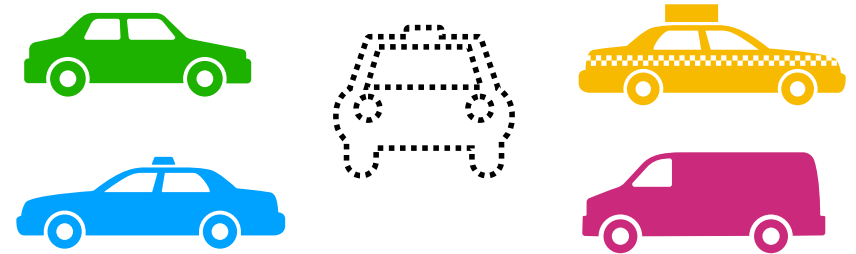
There are many **instances** of cars.



All cars have the **same interface**.
(wheels, doors, steering wheel, etc.)

“Car” is a **prototype**.

There are many **instances** of cars.



But most cars vary in the details
(wheels, doors, steering wheel, etc.)

```
public static void main(String[] args) {  
    System.out.println("I'm static!");  
}
```

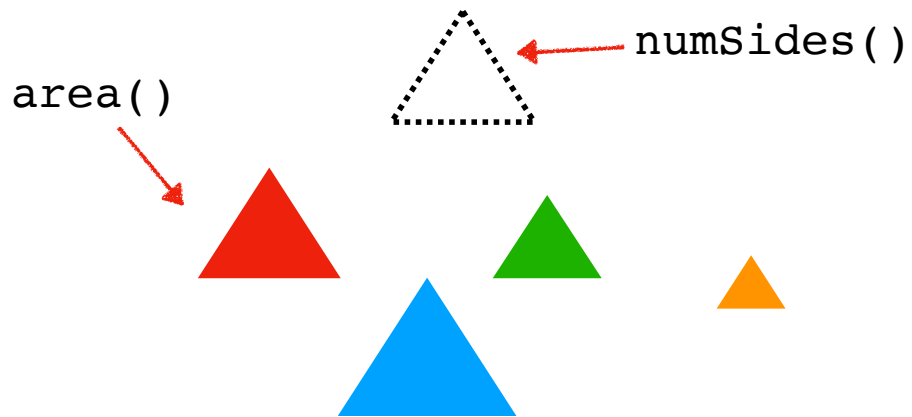
Methods are **functions** that are tied
to either:

1. a **class**, or
2. an instance of a class (an **object**).

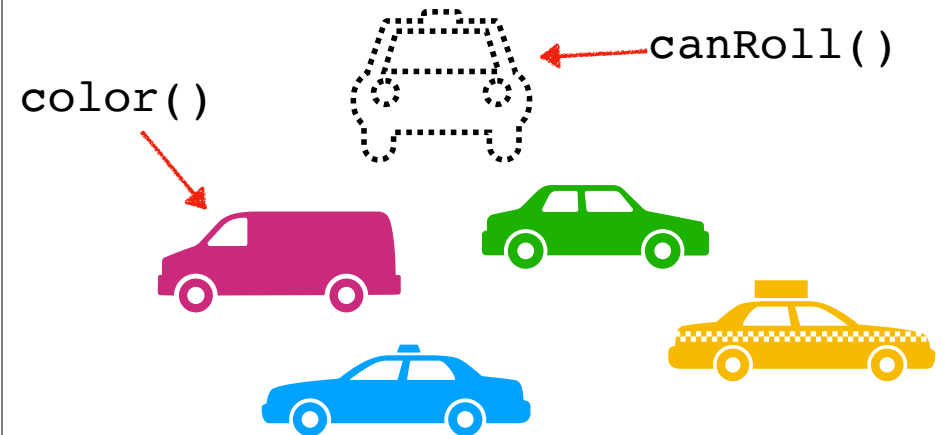
instance method

static method

`static` methods are “attached” to class.
instance methods are “attached” to object.

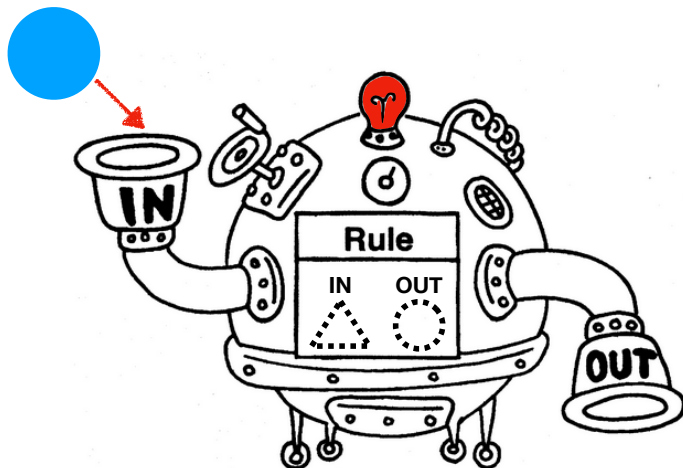


`static` methods are “attached” to class.
instance methods are “attached” to object.



A class also defines a `type`.

Using object incorrectly yields a `type error`.



Let's convert Nim to use objects.

Recap & Next Class

Today:

- Scanner
- Exceptions
- OO

Next class:

- Vectors and generics