CSE 21
Intro to Computing II

Lecture 5 – Object Oriented Programming

Announcement

- Lab 4 due before start of next lab
  - Type your answers in a text file and submit it as an attachment
- Reading assignment
  - Chapter 7.5 to 7.8 of textbook

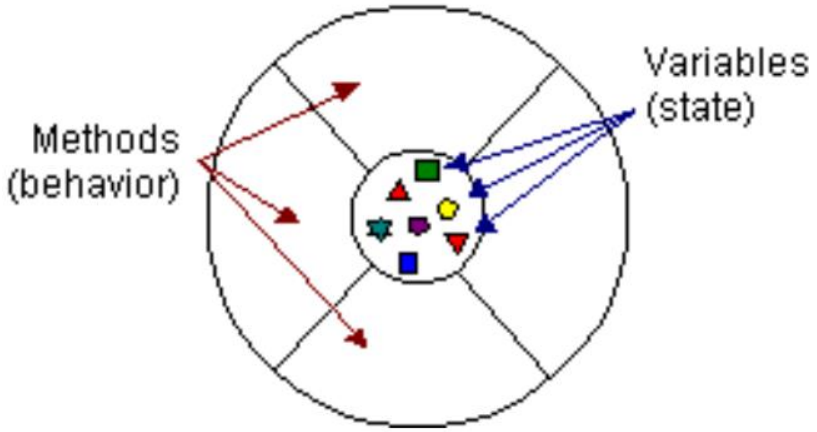
Object-Oriented Programming

- Our new programming metaphor is multiple independent intelligent agents (Objects)
- An object can…
  - ask other objects to do things
    - this is called “message passing”
  - remember things about its own past history
    - this is called “local state”
  - behave just like another except for a few differences
    - this is called “inheritance”
- Many people find this way of thinking and modeling the world more intuitive
  - The world is made up of objects! people, desks, chairs, etc.

What is an Object?

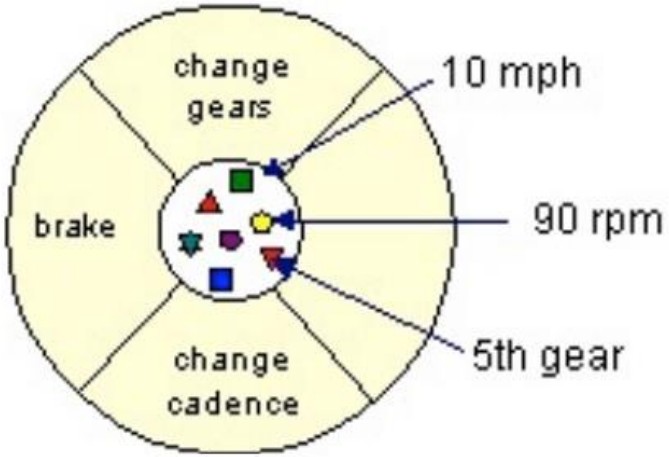
- Real-world objects share states and behaviors
  - E.g., cats have states (name, color, breed, hungry) and behaviors (meowing, sleeping, shredding rugs)
  - E.g., bikes have states (gear, # wheels, # gears) and behaviors (braking, changing gears)
- Software objects are modeled after real-world.
- A software object…
  - maintains its states in one or more variables
  - implements its behaviors with methods
  - An object is a software bundle of variables (what it knows) and related methods (what it can do)
- *Classes* are **"factories"** for generating objects

How can we visualize objects?



- A particular object is called an instance
- Its variables are called instance variables

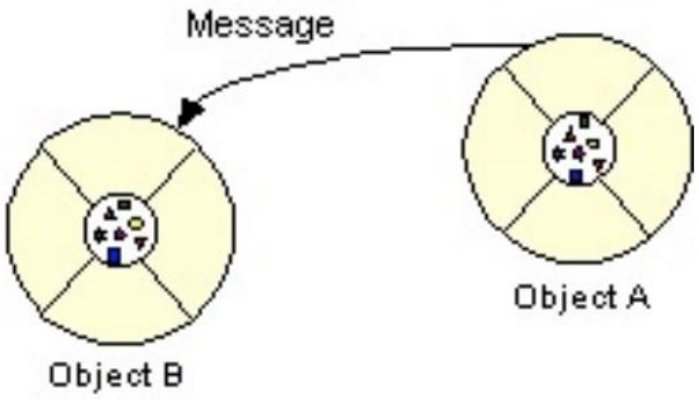
- A SW object modeling a bike:
  - States (in variables): speed: 10mph, cadence: 90rpm, gear: 5th
  - Methods: (brake, change gears, change cadence)
  - Note: no method to change speed directly, it's a side-effect of the gear and how fast you're pedaling!

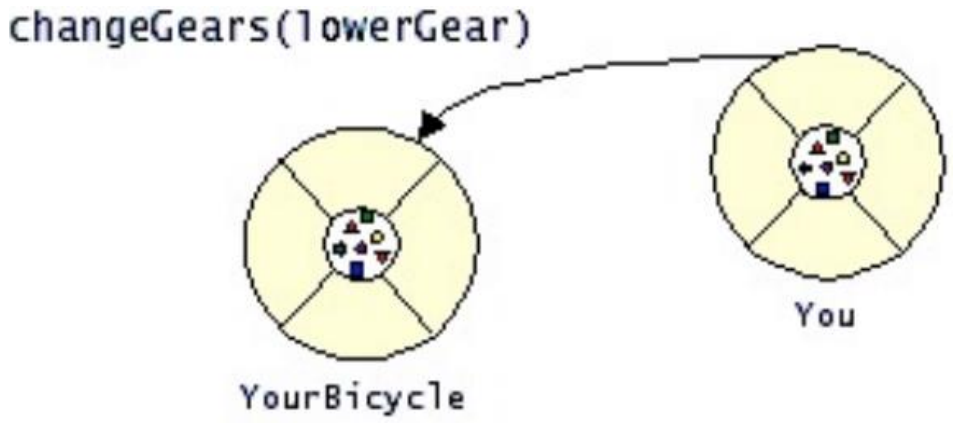


A `Bike` instance

What Is a Message?

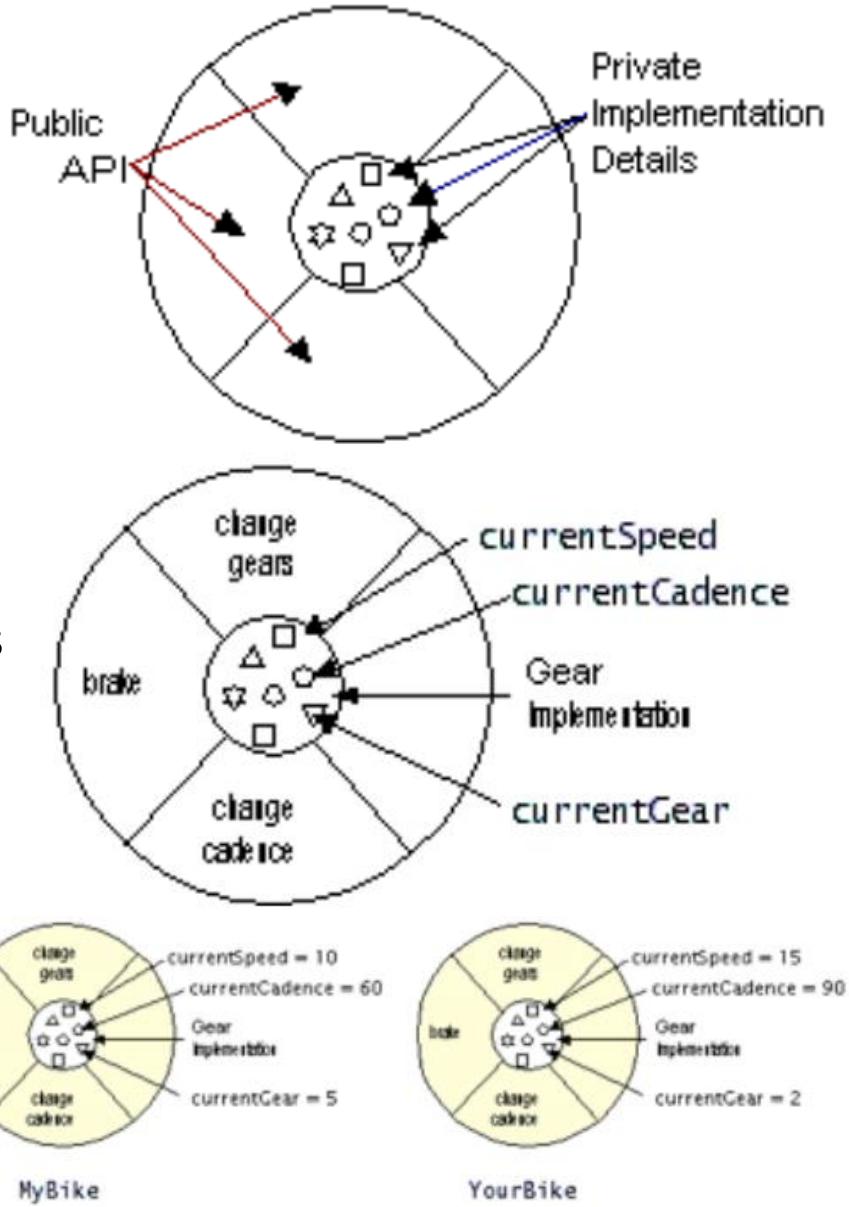
- A single object alone is not very useful…
- An object as a component of a program that has object-object interaction is powerful.
- If object A wants object B to perform one of B's methods, A *sends a message* to B (sometimes with parameters)
- Here, you are asking `yourBicycle` to `changeGears` to `lowerGear`





What is a Class?

- A class is the basic unit of Java. It's the **"blueprint"** or **"factory"** that defines the variables and methods common to all objects of a certain kind.
- Methods isolate, or ***encapsulate*** the data inside from the outside.
  - Other objects ask about this object's state via methods.
- After you have your `Bike` class, you can create any number of (instances of) bike objects!



The Member Access Operator (.)

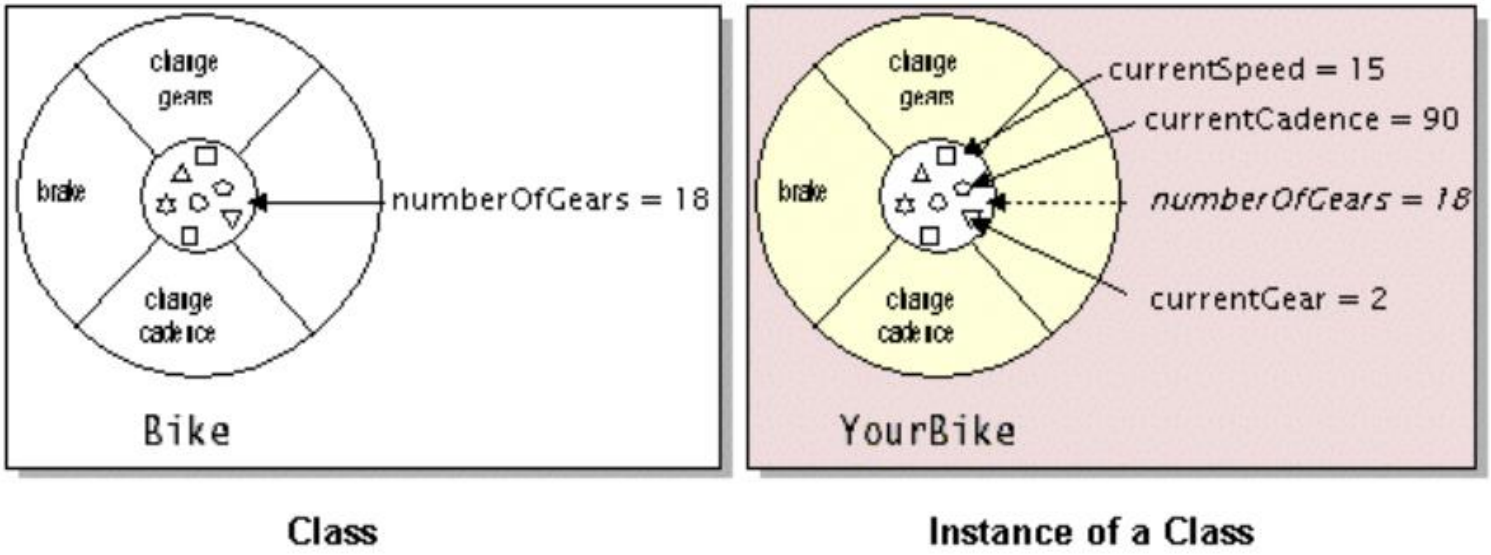
- The members of an object (instance variables and methods) are accessed using the member access operator, or dot operator (.)

- As in…

```
mybike.speed
mybike.changeGears()
```

- Note: methods and variables can have the same name
  - Use () to disambiguate!

Instance vs. Class (static) variables



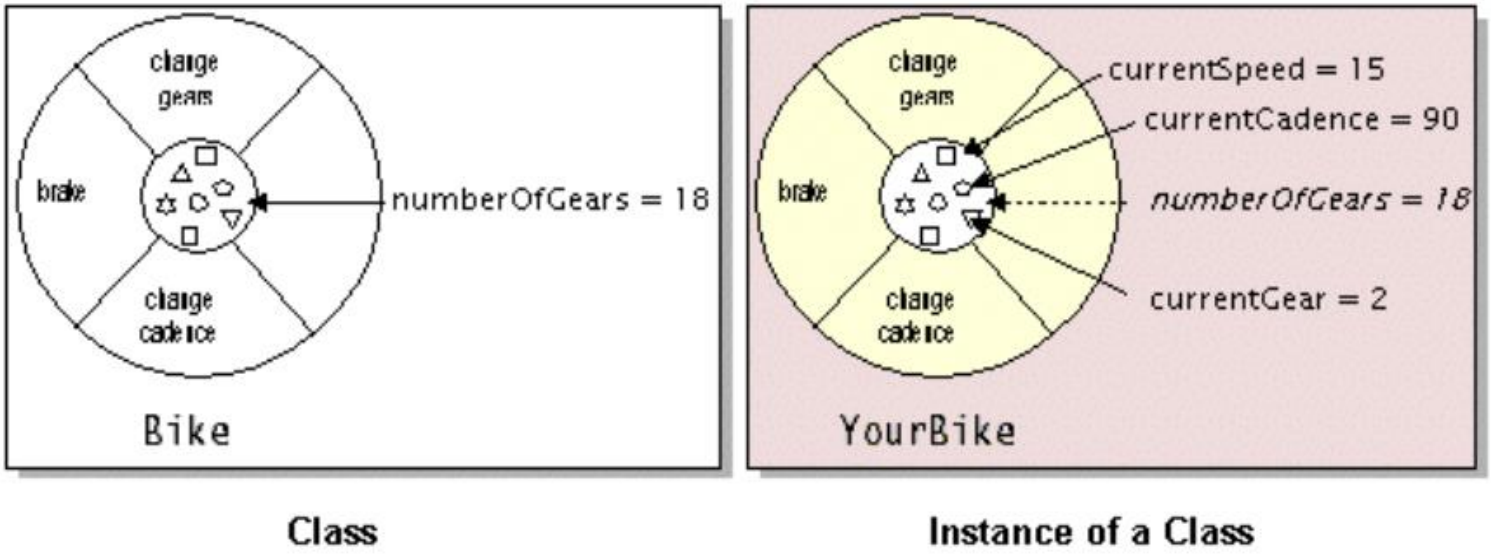
Class

Instance of a Class

- A ***class variable*** (aka ***static variable***) is shared by all instances of a class.
  - Unlike ***instance variables*** that can be different for each instance.
  - E.g., suppose all bikes had the same number of gears. If we made this a class variable, and we wanted to change it, it would change for ALL bikes.

```
static int numGears;
```

Instance vs. Class (static) variables



Class **Instance of a Class**

- Access static variables from the class, not from an instance.

```
Bike mybike = new Bike();
System.out.println(mybike.speed);            // OK
System.out.println(mybike.numberOfGears);    // NO!
System.out.println(Bike.numberOfGears);      // Good
```

Common Methods in a Class

- Methods common to many classes
  - ***Constructors*** are called if you ask for a ***new*** object
    - Java provides a ***default*** constructor (with no arguments)
  - ***Accessors***, or "get methods", or "getters" are used to read/retrieve the values of instance variables
    - Including predicate methods returning `booleans`
  - ***Mutators***, or "set methods", or "setters" are used to set the values of instance variables
  - **`toString`** method creates an important String representation of the contents of the object
    - `System.out.println(obj)` calls object's `toString`

Designing a Class

- To design a class, think about what the objects in that class should do
  - Determine the set of variables (your state)
    - inside each object (instance variables)
    - shared by all objects in a class (class variables)
  - Determine methods (your API, or “behavior”)
    - Constructors (these build an instance)
    - Accessors (these query info of your state)
    - Mutators (if any) (these change the object)

Constructors

- Constructors are called when you request a new object
  - Method Signature:

```
public <Class> (args…) { … }
  public Bike(double s) {
     speed = s;
  }
```

  - Called by:

```
<Class> var = new Class(args…)
  Bike myBike = new Bike(3.5);
```

  - Java provides a ***default*** constructor (with no arguments)