Object-Oriented Programming

Dr. Sarah Abraham
University of Texas at Austin
Computer Science Department

Elements of Graphics
CS324e
Objects in Code

❖ Objects are:
  ❖ A grouping of related functions and variables
❖ This assists programmers by:
  ❖ Providing code structure and organization
  ❖ Allowing for more modular, higher level considerations
Classes

- Defines a group of related methods (functions) and fields (variables)
- Defines the behaviors and interactions of these methods and fields
- Outside classes do not need to consider implementation — just expected behavior
Object Instances

- Constructed based on the parent class’s specifications
- Multiple objects from the same class are independent
  - Can act (and be acted upon) in individual ways
- But objects still have same expected behavior even if they occupy different states
# Class Versus Object

<table>
<thead>
<tr>
<th>Car Class</th>
<th>Car Object</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Name:</strong> Car</td>
<td><strong>a_car</strong></td>
</tr>
<tr>
<td><strong>Fields:</strong> make, model, color, speed</td>
<td><strong>make:</strong> Honda</td>
</tr>
<tr>
<td><strong>Methods:</strong> accelerate(), brake()</td>
<td><strong>model:</strong> Civic</td>
</tr>
<tr>
<td></td>
<td><strong>color:</strong> black</td>
</tr>
<tr>
<td></td>
<td><strong>speed:</strong> 0</td>
</tr>
</tbody>
</table>
class Spot {
    float x, y, radius;
    void display() {
        ellipse(x, y, radius, radius, radius);
    }
}

What’s Missing?

- Write a method that will “complete” our Spot class!
Constructors

- Block of code that is activated upon object instantiation
- Method always shares class name
- Can assign values to object fields
Multiple Constructors

Spot() {
    x = 50;
    y = 50;
    radius = 30;
}

Spot(float x, float y, float _r) {
    this.x = x;
    this.y = y;
    radius = _r;
}
Using Objects

❖ Each object from a class must be created using keyword new:

```java
sp1 = new Spot();
sp2 = new Spot(75, 80, 15);
```

❖ Now we can display each object individually in `draw()`:

```java
void draw() {
    sp1.display();
    sp2.display();
}
```
Class Files

- A single file can contain all of a program’s classes BUT please use separate files for each class
- Multiple files:
  - Provide modularity
  - Are easier for groups to coordinate
Using Multiple Files

1. Create main program (setup() and draw() functions) in a sketch folder

2. Select “New Tab”

3. Give the file the name of the class it contains

4. Reuse class files by copying them to other sketch folders

Note: Each Processing sketch can only have one setup and draw function call
Extending Class Functionality

❖ Fields represent meaningful object values
❖ What might speed represent in Spot?
❖ What might direction represent in Spot?
❖ Methods represent meaningful object behaviors
❖ How could we use a move() method in Spot?
Hands-on: Creating Classes

Today’s activities:

1. Implement the Spot class in a Processing sketch. Be sure that it is within its own file
2. Add a speed field and a move() method, so the spot’s position can update
3. Create at least two Spot objects that start out with different positions and speeds
Spot sp;
void setup() {
    size(100, 100);
    sp = new Spot();
    sp.x = sp.y = 50;
    sp.radius = 15;
}
void draw() {
    sp.display();
}

class Spot {
    float x, y, radius;
    Spot() {...};
    void display() {
        ellipse(x, y, radius, radius);
    }
}
What does the keyword \texttt{this} mean?

\begin{verbatim}
Spot(float x, float y, float r) {
    this.x = x;
    this.y = y;
    this.r = r;
}
\end{verbatim}
Referring to an Instance

- Keyword `this` refers to the instance calling on the class functions or fields
- Same thing as `self` in Python
- Every instance knows who they are (`this` is implicit to all function calls and fields!)
- Must explicitly use `this` if a field is hidden by a local variable
Using Objects in Objects

- Objects can be fields of other objects
  - Allows for better code reuse and cleaner division between concepts
- `PVector` is a class that provides support for vectors
  - Stores x, y, z values as fields
  - Provides methods with useful mathematical functionality
Designing Classes

❖ What should be stored in fields?
❖ Data that creates a meaningful representation of the object in question

❖ What methods should be implemented?
❖ Functionality that has a clear purpose and is likely to be called multiple times
❖ Helper methods are smaller methods that can assist in building out clean functionality
Designing Classes

❖ There are no hard rules for when and how to build classes!
❖ Take problem into consideration before starting the design
❖ Use naming conventions for both fields and methods that express the purpose of that variable or function
❖ If possible, avoid writing the same functionality out in multiple places
Where to Call “new”

❖ Calling `new` in `draw` will instantiate an object that is local to the `draw` call
❖ Possible to save the object into a global array to make it accessible between frames
  ❖ Must be done with great care!
  ❖ `new` (the allocation of memory) is expensive
❖ Try to create objects as infrequently as possible
  ❖ Create objects upfront during `setup`
  ❖ Create objects based on user input in mouse/key callbacks
  ❖ Create objects using timers (will be discussed later)
Instapoll Question: Classes

Given this code and assuming all Spot methods have been implemented:

```java
void setup() {
    size(100, 100);
}

void draw() {
    Spot sp = new Spot(50, 50, 15);
    sp.display();
    sp.move();
}
```
What is the Output?

- Code will not compile
- Code will compile, object `sp` will move but not display
- Code will compile, object `sp` will not move but will display
- Code will compile, object `sp` will move and display
Hands-on: Complete Spot Class

- If you have not done so already, make sure your Spot class is working
- You do not need to resubmit, but please come to office hours if you are still uncertain how something works