Object-Oriented Programming

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

Car Class

Name: Car
Fields: make, model, color, speed
Methods: accelerate(), brake()

Car Object

a_car
make: Honda
model: Civic
color: black
speed: 0
Class Code Example

class Spot {
    float x, y, radius;
    void display() {
        ellipse(x, y, 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();
  }
  ```
Spot sp;
void setup() {
    size(100, 100);
    sp = new Spot();
    sp.x = sp.y = 50;
    sp.radius = 15;
}
void draw() {
    sp.display();
}
Question

- What does the keyword `this` mean?

```java
Spot(float x, float y, float r) {
    this.x = x;
    this.y = y;
    this.r = r;
}
```
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
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?
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
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();
}
```

- 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: Using 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