Class 14: Object-oriented programming II

Programming for VR I

Patrick Mineault

Using other people's classes

- Copy paste at the top of your main file
- Add it as a separate file and import
- Install a package and import (common outside of Processing)

Second method

(illustration)

Using the import statement for your class files

- the imported file has to be in the same directory as the importing file
- The import name matches the filename: for file MyFile.py, use `import MyFile`

Interfaces

- The power of classes comes through when we create classes with a consistent interface
- Interface: a set of methods and attributes
- If we keep a common interface we can use a bunch of classes interchangeably

Different interfaces

```
class Turtle:
  def __init__(self):
    self.x = 0

  def walk(self):
    self.x += 1

class Hare:
  def __init__(self):
    self.x = 0

  def run(self):
    self.x += 5
```

Same interface

```
class Turtle:
  def __init__(self):
    self.x = 0

  def move(self):
    self.x += 1

class Hare:
  def __init__(self):
    self.x = 0

  def move(self):
    self.x += 5
```

Keeping things consistent

- Makes it easy to use two classes interchangeably
- In Python, interface definition by convention works because of what's called duck typing: if it looks like a duck and it quacks like a duck, it's a duck.

Other ways of defining consistent interfaces

- Hierarchy of classes
- For instance, Turtle and Hare could derive from a class called Mammal.

```
class Mammal:
  def number_of_limbs(self):
    return 4

class Hare(Mammal):
  def __init__(self):
    self.x = 0

  def move(self):
    self.x += 5

class ThreeLeggedDog(Mammal):
  def number_of_limbs(sef):
    return 3

hare = Hare()
dog = ThreeLeggedDog()
print(dog.number_of_limbs())
print(hare.number_of_limbs())
```

We won't use this much

- However, other languages use the inheritance model a lot.
- e.g.: https://docs.unity3d.com/ScriptReference/SphereCollider.html

Let's exercise together!

Let's make a paint tool with multiple tools. Our generic tool will look like this:

```
class Tool:
  def mouse_down(self):
    # When the mouse is first down

  def mouse_drag(self, pmouseX, pmouseY, mouseX, mouseY):
    # When the mouse is still down
    pass

  def mouse_up(self):
    # When the mouse is first up

  def get_icon(self):
    return "filename_32x32.png"
```

Sample tool

```
class Pen:
    def mouse_up(self, mouseX, mouseY):
        print("mouse up")

    def mouse_down(self, mouseX, mouseY):
        print("mouse down")

    def mouse_drag(self, pmouseX, pmouseY, mouseX, mouseY):
        line(pmouseX, pmouseY, mouseX, mouseY)

    def get_icon(self):
        return "pen.png"
```

Generic tool project

https://github.com/patrickmineault/programming-course/tree/master/class13-14/tool

Tools you will make

- Sticker tool (on mouse down, draw a sticker)
- Rectangle tool (on mouse down, remember position; on mouse up, draw)
- Spray can tool (on mouse drag, draw a circle at a random position close to the mouse cursor)
- Airbrush (on mouse drag, draw a smoothed circle)

End point

5-tool paint program