LECTURE 6

OBJECT-ORIENTED PROGRAMMING

SUBCLASSES AND INHERITANCE II

MCS 275 Spring 2021
Emily Dumas

LECTURE 6: SUBCLASSES AND INHERITANCE II

Course bulletins:

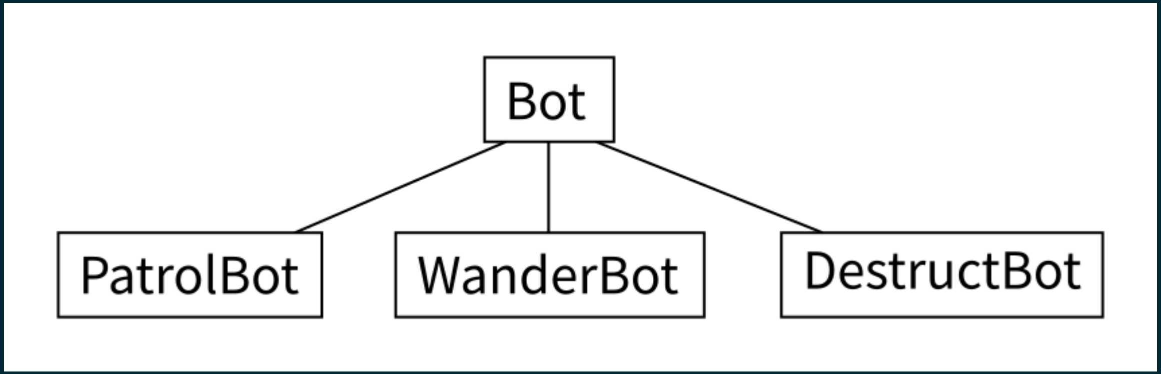
- Quiz 2 is due at Noon tomorrow (Tue Jan 26).
- Project 1 posted. Deadline 6pm CST on Fri Feb 5.
- Project 1 autograder opens on Mon Feb 1.
- Quiz 3 and Worksheet 4 will be lighter (so you can prioritize project work).

PLAN

Finish our robot simulation class hierarchy

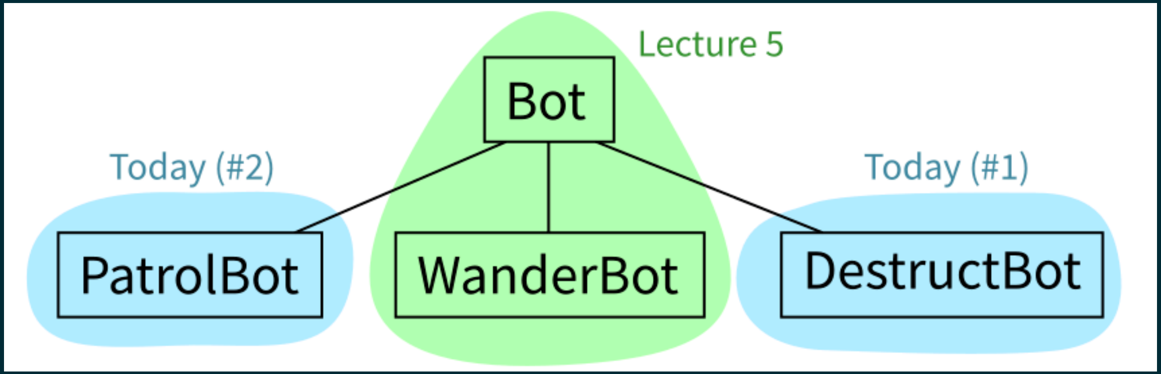
Discuss more OOP theory & practice

PLANNED BOT HIERARCHY



- `PatrolBot` walks back and forth.
- `WanderBot` walks about randomly.
- `DestructBot` sits in one place for a while and then self-destructs.

PLANNED BOT HIERARCHY



- `PatrolBot` walks back and forth.
- `WanderBot` walks about randomly.
- `DestructBot` sits in one place for a while and then self-destructs.

CLASS ATTRIBUTES

Attributes declared in the class definition, outside of any method, are **class attributes**.

Class attributes are shared by every instance of the class. Often used for constants.

Contrast with the **instance attributes** we have used thus far (e.g. `self.x = 1` in constructor) which exist separately for each instance.

FOUR PILLARS OF OOP

- **Encapsulation** - Classes manage their own private, internal state.
- **Abstraction** - Method calls express intent (independent of implementation).
- **Inheritance** - Distinct classes can share behavior.
- **Polymorphism** - Code using a class will also work on its subclasses.

EXTENDING THE SIMULATION

Beyond adding more robot types, how might me improve or extend the simulation?

EXTENDING THE SIMULATION

Might create a class `Arena` that manages the list of bots and the space in which they move. Would have a single `.update()` method that updates all bots.

`Arena` could have a metthod to render itself as a string for display (or as a PNG, HTML, ...).

EXTENDING THE SIMULATION

If we wanted to add robot interaction or movement constraints then the `Bot` class would need a way to access information about its surroundings.

We might make a parent `Arena` a required argument to the `Bot` constructor.

`Bot.update()` could call methods of `Arena` to learn about other robots, movement limits, etc.

e.g. in `Bot.update()`:

```
self.arena.bots_visible_from(self.position,self.sight_range)
```

REFERENCES

- I discussed inheritance in MCS 260 Fall 2020 Lecture 25, using "Square is a subclass of Rectangle" as an example in this geometric object module.
- See *Lutz*, Chapter 31 for more discussion of inheritance.
- *Lutz*, Chapters 26-32 discuss object-oriented programming.

REVISION HISTORY

- 2021-01-25 Fixed typo
- 2021-01-23 Initial publication