Application Programming

Week 3

Lecture 1

Deliverables

- More on Polymorphism
- Motivation for Arraylists and Java I/O

Polymorphism

- It is the provision of a single interface to entities of different types.
- Let’s start with an example:
  - A vehicle is a means of transportation between locations.
  - A car is a type of vehicle.
    - It has doors, wheels, a gas pedal, brake.
  - A 2016 Aston Martin Vulcan is a type of car.
    - It has unique features that differentiate it from other cars.
  - There are multiple 2016 Aston Martin Vulcans in production

Inheritance

- Enables new objects to take on the properties of existing objects.
- Superclass → a class that is used as the basis for inheritance.
- Subclass → a class which inherits from a superclass.

Inheritance

- Vehicle

Car

2016 Aston Martin Vulcan

Interface

Defines some functionality, which must be implemented in a subclass.

No implementation.

Abstract Class

Some implementation, but defines some functionality which must be implemented in a subclass.

Class

Implementation code.

Implements a superclass or extends an abstract class.

Vehicle Example

- In this example, we have 3 Java classes:
  - Vehicle is an interface, declared in `Vehicle.java` as
    - `public interface Vehicle { }`
  - Car is an abstract class, declared in `Car.java` as
    - `public abstract class Car implements Vehicle { }`
  - AstonMartinVulcan2016 is a class, declared in `AstonMartinVulcan2016.java` as
    - `public class AstonMartinVulcan2016 extends Car { }`
  - …. and objects can be created from it as

```
AstonMartinVulcan2016 mine = new AstonMartinVulcan2016();
AstonMartinVulcan2016 yours = new AstonMartinVulcan2016();
```

Payroll System Example

- We want to build a payroll system for a company.
- We can assume a few basics…
  - Employees get paid.
  - Employees were hired on some date.
  - Employees have names, a title
  - The company has a name

Payroll System Example

- From these assumptions, we can create a few basic classes.
  - `Employee.java` should have variables for name, title, hire date, and pay (since all of these are unique to individual employees).
  - `Company.java` should contain all of the employees, and have a name.
  - It would be useful to have `Date.java` which handles the hire dates for us.

Payroll System Example

- But what if some employees are hourly, and some are salaried?
  - A salaried employee works with our current code.
  - An hourly employee will need an hourly wage and the number of hours they work.

Payroll System Example

- But what if some employees are hourly, and some are salaried?
  - A salaried employee works with our current code.
  - An hourly employee will need an hourly wage and the number of hours they work.

Interfaces and Abstract Classes

- An interface is a means to describe a set of methods that can be called on an object.
  - Contains only constants and abstract methods.
  - Cannot specify any implementation details.
- An abstract class is conceptually the same, except that it can opt to specify implementation details.
- An interface is often used in place of an abstract class when there's no default implementation to inherit.
- *Should Employee be an interface or an abstract class?*