Inheritance: Part 1

Today's Objectives

- Identify subclasses and superclasses
- Define and utilize subclasses and superclasses
- Constructor chaining

Specializations: "is-a"

- Derived classes are specializations of a base class
- A ReserveAuction or BuyoutAuction ***is a*** specific type of Auction
- A GraphingCalculator is a more specific type of Calculator

Inheritance

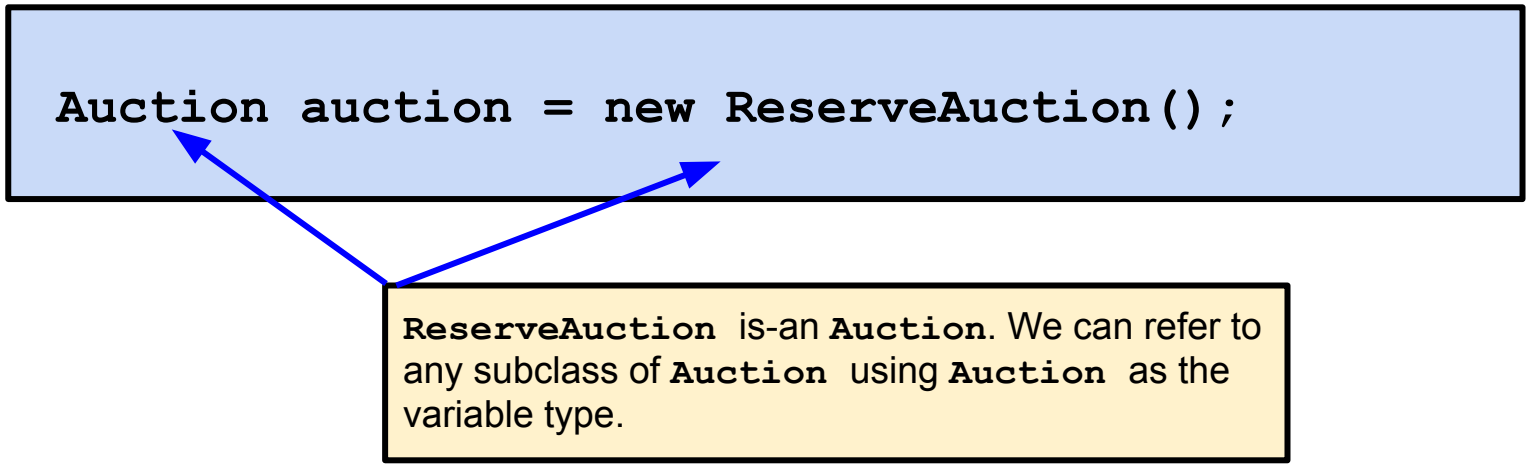
- Allows a class to take on the properties and methods defined in another class.
  - A **subclass** is the derived class that inherits the data and behaviors from another class.
  - A **superclass** is the base class or parent class whose data and behaviors are passed down.
  - All classes are actually subclasses of the `java.lang.Object` class.
  - You may hear superclass referred to as the **parent** class and subclass referred to as the **child** class.
- A class can inherit from another class using the `extends` keyword.
- Subclasses must implement superclass constructors if not using the default constructor (use `super` keyword).
- `private` vs. `protected` access modifiers
  - `protected` acts as `private` to all other classes but every class that extends the class will still have access as if defined with the `public` access modifier.

Inheritance: Overriding Methods

- A subclass can **override** a method from the superclass by redefining the method.
    - When a subclass method is called, the subclass method will be called if defined, otherwise the superclass method will be.
    - Method **signature must match** the signature being overridden **exactly**.
    - Java provides the `@Override` annotation to make it clear a method overrides the original method.
        - If you use the `@Override` annotation on a method you intend to override, you will get a compiler error if your signature does not match the signature of any signatures in the superclass. This is very useful to ensure your method **WILL** actually override as intended.
- If a subclass overrides a superclass method, that class can always call the superclass method by using the `super.` prefix to access the super version of the method.

Inheritance As Polymorphism

- Specialization classes can be referred to by their base class



- This promotes polymorphic code
    - Classes can only inherit from one class

Inheritance: A Few More Notes

- A class can only inherit from one class.
- Inheritance is transitive:
  - If class B inherits from class A,
    - class B "is-a" class A
    - classes that inherit from class B, they still have an "is-a" relationship with class A
- Constructors are **not** inherited and must always be invoked using **`super`**
  .
- Classes can chain constructors by using **`this`** to call another overloaded constructor:
  - `this("Hi", 1125);`
- The **`protected`** keyword has a caveat you should be aware of:
  - It is actually not only classes that inherit from the subclass that have access to **`protected`** properties or methods, but ALL classes in the same package. This can causes unintended consequences so avoid using **`protected`** properties unless absolutely necessary..

Let's Code!