2/3 - Inheritance

Tuesday, February 3, 2015

2:04 PM

* Interfaces are "contracts" for objects
  + The object classes are obligated to implement the methods in the interface
  + The interfaces have no implementation
  + Interfaces act as a data type : meaning that you can create an object that is of an interface type
  + A class that implements an interface creates an "is-a" relationship between class and interface
* Type casting is allowed across an is-a relationship. To do so, you have to type cast DOWNWARDS, meaning you type cast an object back into an interface. For example:

Song s = new Song();

Media m;

m = (Media)s;

* That last step casts the song into a media object

* Inheritance
  + Extending interfaces
    - After you make an interface, you can then extend it. Extending it makes it more complicated and adds on more features or changes old requirements. Any class that inherits from the extended interface has everything from the original interface as well.
    - Ex:

public interface CompressedMedia extends Media {

int getCompressedSize();

//more methods than just Media interface

}

* Now if a new class implements the extended interface, then this class must provide methods for BOTH interfaces
  + This is the idea of polymorphism: an object can be of many different types
* Extending interface is only necessary if you have to add functionality/requirements onto an existing interface that wouldn't make sense without the original requirements
  + Otherwise, you can just make a separate interface that is entirely unrelated
    - Then a class can possibly implement BOTH of these

Ex:

public class Song implements Compressed,Media{

//stuff

}

* Important!!!!
  + In order for interfaces to share methods, you have to type cast
    - If you have a Person interface that has a method getFirstName() and a student interface that doesn't, then in order to invoke the getFirstName method, you have to type cast the student into a person
    - Ex:

Public interface Student{…stuff}

Public interface Person{..stuff}

Student s1;

s1. firstName = "Mary";

String name = (Person)s1.getFirstName();

* When you look at it this way, the type casting seems a little unnecessary
  + What we should've done is made Student inherit from Person so that you don't have to typecast anymore. Rather than having 2 separate interfaces, you have one interface extend the other!

Ex:

Public interface Student extends Person{…..stuff}

Then you can do the same as original but no more type casting:

Student s1;

S1.firstName = "Mary"

String name = s1.getFirstName();