1.00 Lecture 15

Interfaces

Reading for next time: Big Java: sections 14.1-14.6, review 9.3

Interfaces

- **Interface is a specification for a set of methods a class must implement**
  - **Interfaces specify but do not implement methods**
  - **A class that implements the interface must implement all its methods**
  - **You may then invoke methods on this class that rely on the interface. Examples:**
    - **If your class implements a Printable interface that has a printData() method, you can put objects of your class into arrays or ArrayLists and call that method**
    - **You will use interfaces frequently in Swing (GUI) and numerical methods**

Interfaces, p.2

- Interfaces are like an abstract class but:
  - If they were implemented as an abstract class, a subclass could only inherit from one superclass
  - Multiple interfaces can be inherited (e.g., Drawable and Rotatable) in your class
  - Interfaces cannot be instantiated

```
Drawable shape1= new Drawable();        // Error
```

  - You can declare objects to be of type interface

```
Drawable shape1;                        // OK
```

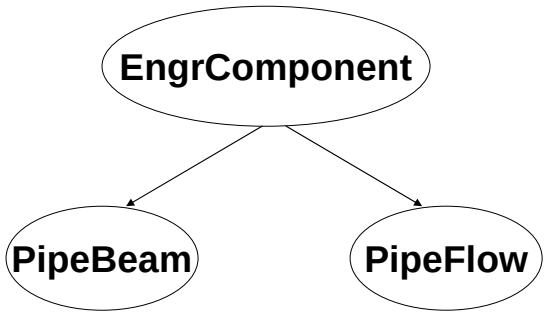
  - They can be names for objects of a class that implements the interface. Assume Rectangle implements Drawable:

```
Drawable shape1= new Rectangle();       // OK
```

  - Interfaces may contain methods and constants

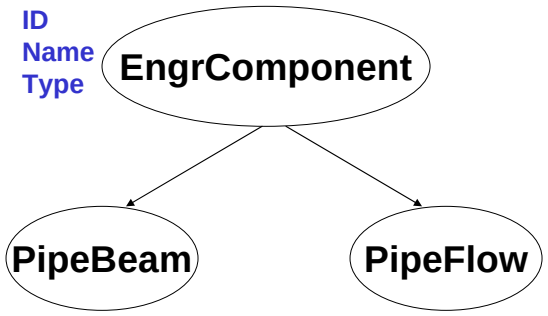
```
public interface Rotatable {
    void rotate(double theta);        // Required method(s)
    double MAX_ROTATE= 360; }         // Implicitly final
    // Methods and fields default to be public
```

Interfaces and multiple inheritance

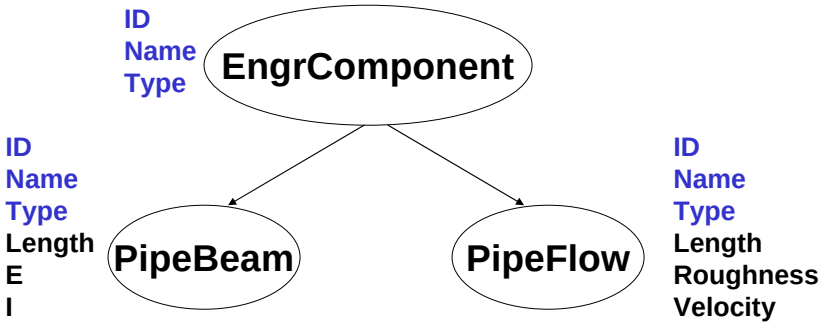


Assume we have an engineering analysis system that does materials, thermo, fluids, etc. on many components

Interfaces and multiple inheritance

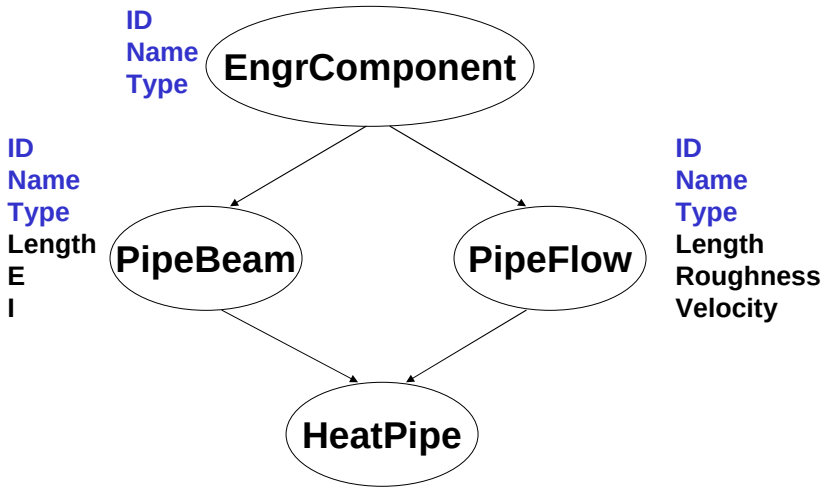


Interfaces and multiple inheritance



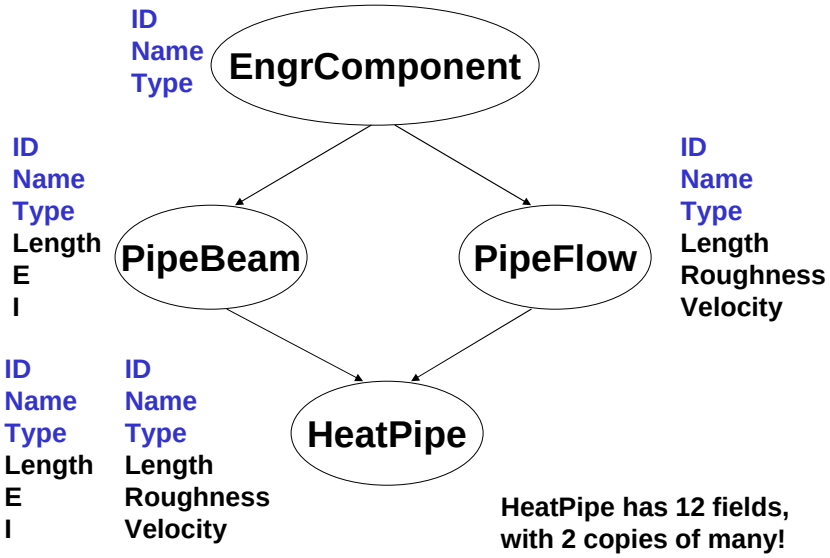
We now want a component that is a pipe with flow and is a circular beam. We want to get its heat transfer rate through the fluid and metal (e.g. to cool an ice rink) and its material strength. We want *multiple inheritance* from PipeFlow (heat transfer methods) and PipeBeam (materials methods)

Interfaces and multiple inheritance



What member data fields will HeatPipe have *(in C++)*?

Interfaces and multiple inheritance

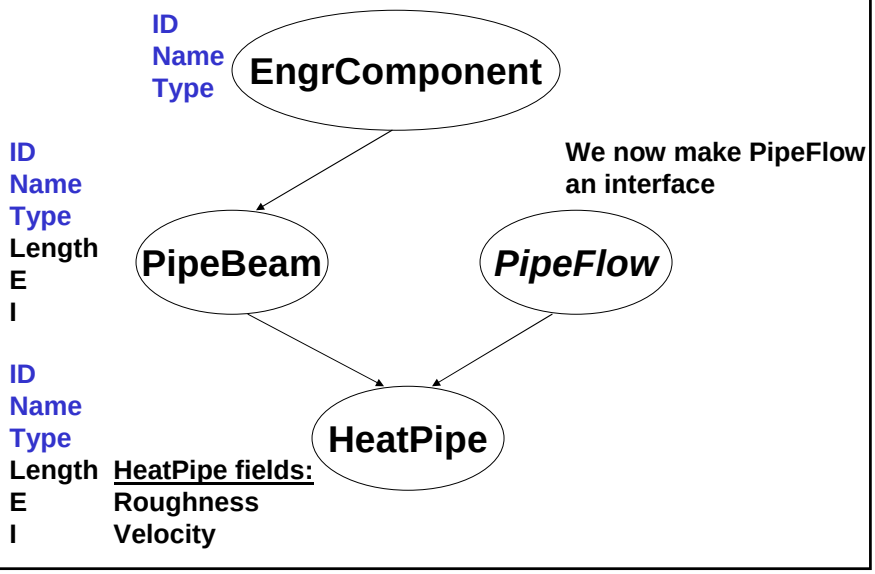


HeatPipe has 12 fields, with 2 copies of many!

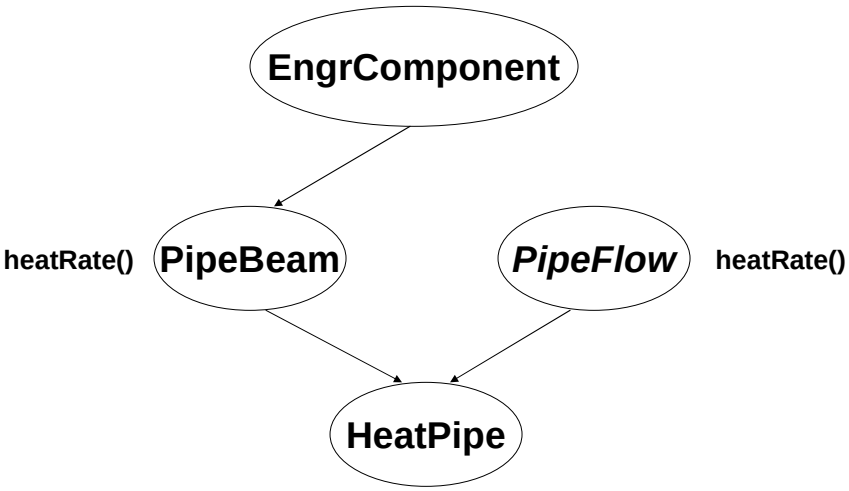
Interfaces and multiple inheritance

- So, fields in classes with multiple inheritance are a bad idea
  - Java implements multiple inheritance with interfaces, a very restricted abstract class, to avoid this difficulty
  - Java allows no instance fields in interfaces, only final (constant) fields that don't have this difficulty
- Now, let's look at methods…
  - What if interfaces allowed non-abstract methods (methods with bodies)?

Interfaces and multiple inheritance, part 2



Interfaces and multiple inheritance, part 2



Which heatRate() is invoked by heatRate() or super.heatRate() in HeatPipe?

Interfaces and multiple inheritance, part 2

- **So, non-abstract methods in classes with multiple inheritance are a bad idea**
  - **Java implements multiple inheritance with interfaces, a very restricted abstract class, to avoid this difficulty (Yes, we're repeating ourselves…)**
  - **Java allows no method bodies in interfaces, only abstract methods that don't have this difficulty**
  - **This forces you to implement the method in the subclass, so there is no ambiguity when the method is called**
- **Interfaces are Java's way of letting a class be two or more 'types' or things, which is multiple inheritance**
  - **Interfaces are very restricted compared to full multiple inheritance, but they are much safer and easier to understand. And they're useful enough to be worth it, even though they don't allow as much reuse as we'd like**
  - **Even if you implement an interface twice by accident in a set of subclasses, it's still unambiguous (and this happens!)**

Interface exercise

- Write an interface
  - In Eclipse: File->New->Interface
  - Call it Printable
  - Define one method, printData()
- Save Printable
- Write a Student1 class (File->New->Class, as usual)
  - Private data: String name, int year
  - Write a constructor
  - Implement Printable
    - Use 'implements Printable' in class declaration
    - Write a printData() method. Eclipse will try to help you!
- Save/compile Student1

Interface exercise, p.2

- Write a bogus class:

```
public class Bogus implements Printable {
   public void printData() {
     System.out.println("Bogus");
   }       // Java will write a constructor automatically
}          // for any class; nothing to do with interfaces
```

- Write an InterfaceTest class, with just a main method:
  - `import java.util.*;` at line 1 to be able to use ArrayLists
  - Create two new Student1s and a new Bogus
  - Create an ArrayList (remember ArrayLists?)
    - `ArrayList arr= new ArrayList();`
  - Add the two Student1s and one Bogus to the ArrayList
    - `arr.add(s);`
  - Loop through the ArrayList and invoke `printData()` on each element
    - Guaranteed to be there because Printable implemented
    - Remember `arr.size()` gives ArrayList size
    - `arr.get(i)` gives `Object` at slot i
    - Must cast `Object` to `Printable` as you get it from the ArrayList

Inheritance- key points

- Inheritance allows a programmer to extend objects that she did not write
  - Access restrictions still hold for the super class
    - If the base class changes private data or members, the sub classes should be unaffected
  - Protected members in super class allow direct access by sub classes
    - Must not change in super class; must be designed with intent to allow use by sub classes
  - Sub class has all data (private, protected and public) of the super class. Each object has all this data.
    - Sub class can use only public and protected methods and data of the super class, not private methods or data
  - All Java objects inherit implicitly from class Object
    - Java libraries, Java documentation use Object frequently
- Interfaces are like an abstract base class, but allow classes to inherit more than one interface
  - Restricted form of multiple inheritance allowed in Java

Inheritance review questions

1. What is the difference between an abstract class and an interface?
   - When do you use an interface? An abstract class?
2. Why does Java not support multiple inheritance?
3. Can an object be of more than one data type?
   - If so, give an example.
4. What is polymorphism?
5. What is the base class for all Java classes?
6. Can an abstract class also be defined as final?
   - Try it!