Interfaces and Abstract Classes

1 Feb 2010
CMPT166
Dr. Sean Ho
Trinity Western University

TRINITY WESTERN UNIVERSITY

Quiz 2: 10min, 20pts

- Explain in detail each part of the declaration: **[5]**
  - **public static void main(String args[])**
- Create a constant array of integers, called fibInts, with values 1, 1, 2, 3, 5. **[4]**
- Write Java code to iterate through fibInts, calculating the sum of its entries. **[4]**
- Sketch a class hierarchy with at least one superclass and at least two subclasses, with attributes and methods appropriate for each class. No code is needed, but the design should make sense. **[7]**

Quiz 2: answers #1

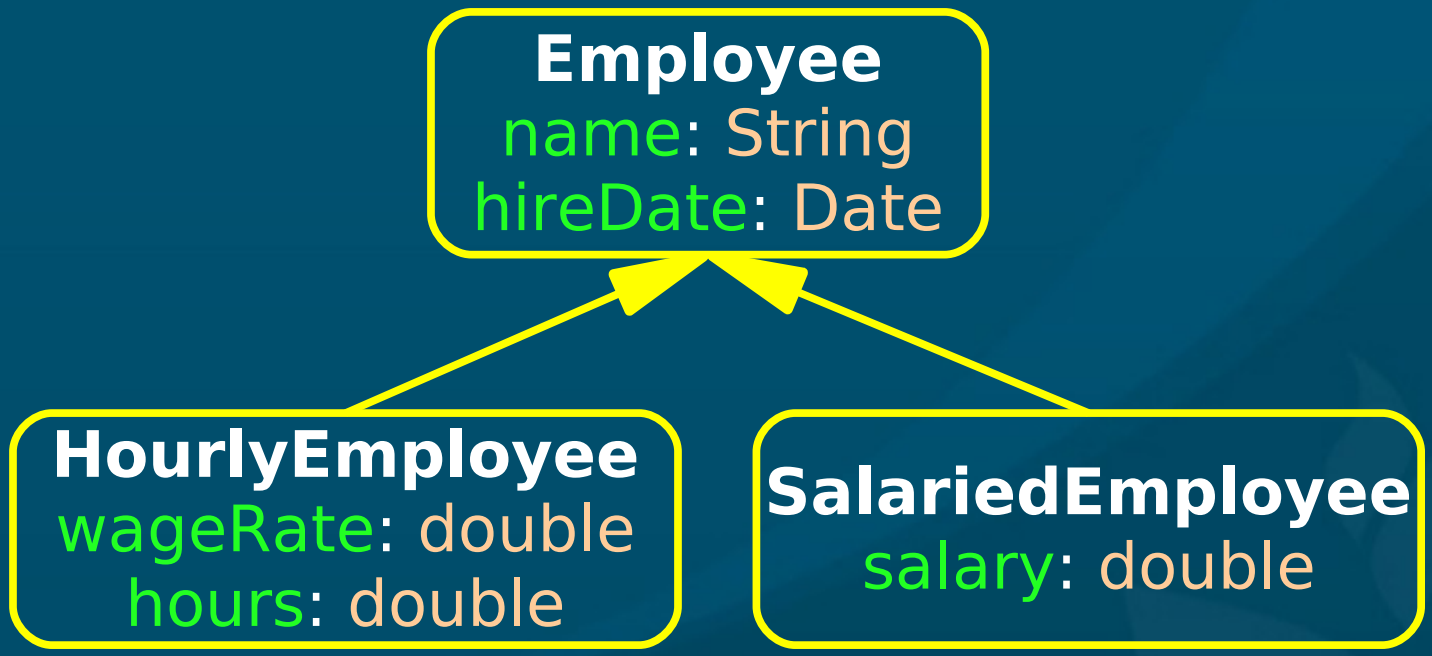
- Explain in detail each part of the declaration: **[5]**
  - **public static void main(String args[])**
  - public: accessible everywhere this class is imported
  - static: class method, not instance method, so VM can call main() without creating an instance of the class
  - void: return type: doesn't return anything
  - main(): name of the method
  - String[]: array of Strings

Quiz 2: answers #2-3

- Create a constant array of integers, called fibInts, with values 1, 1, 2, 3, 5. [4]
  - **final int fibInts = {1, 1, 2, 3, 5};**
- Write Java code to iterate through fibInts, calculating the sum of its entries. [4]
  - **int sum = 0;**
  - **for (int i=0; i<fibInts.length; i++)**
    - **sum += fibInts[i];**
- Or:
  - **for (int elt : fibInts)**
    - **sum += elt;**

Quiz 2: answers #4

- Sketch a class hierarchy ... [7]



Subclasses, instances, attributes

- Recall classes are user-defined container types
- A subclass inherits attributes and methods from the superclass
- Subclasses should be seen as specializations of the superclass: “A is a kind of B”
- Instances should be seen as examples of a class: “A is a B”
- Attributes should be seen as components or parts of a class: “A has a B”

Example

- `class Mammal { Heart h; }`
- `class Dog extends Mammal { void bark(); }`
- `class Cat extends Mammal { void meow(); }`
- `Dog fido = new Dog();`
- `Cat smokey = new Cat();`

- "A Dog is a kind of Mammal."
- "fido is a Dog."
- "fido is a Mammal."
- "fido has a Heart."
- "smokey can meow()."

Multiple inheritance (arity)

- Some languages (C++) allow a subclass to inherit from more than one superclass:

```
class Horse { public void eat(); }
class Donkey { public void eat(); }
class Mule : public Horse, Donkey {}// it's both!
```

- How do disambiguate name collisions?

```
myMule.eat();     // which one?
```

  - Specify superclass name:

```
myMule.Horse::eat();
```

- In C++, Python: arity is multiple.
- In Java: arity is single.

Abstract vs. concrete classes

- Abstract classes:
  - Too generic to define a real object
    - e.g., TwoDimensionalShape
  - Not intended to be directly instantiated
    - Java can enforce this: use abstract keyword
    - abstract classes can have abstract methods:
      - No body defined; each subclass must implement
- Concrete classes:
  - Subclass of an abstract class, meant to be instantiated
    - e.g., Square, Circle, Triangle

e.g: TwoDimensionalShape

- Abstract superclass: TwoDimensionalShape
  - Abstract method: draw()

```
abstract public class TwoDimensionalShape {
    abstract public void draw();        // no body
```

- Concrete subclasses: Circle, Square, Triangle
  - Each provide own implementation of draw()

```
public class Circle extends TwoDimensionalShape {
    public void draw() { drawOval( x, y, r, r ); }
}
public class Square extends TwoDimensionalShape {
    public void draw() { drawRect( x, y, w, h ); }
}
```

Interfaces

- Define a set of abstract methods

```
public interface drawableShape {
    public abstract void draw();
    public abstract double area();
}
```

- Classes implement these methods

```
public class Circle implements drawableShape {
    public void draw() { drawOval( x, y, r, r ); }
    public double area() { return 2 * Math.PI * r * r; }
```

- e.g., Java Swing programs that handle events implement the actionListener interface

Abstract classes vs. interfaces

- Abstract superclasses declare identity:
  - "Circle is a kind of TwoDimensionalShape"
  - Each class can have only one superclass
    - No multiple inheritance in Java
  - Inherit methods, attributes; get protected access
- Interfaces declare capability:
  - "Circles know how to be drawableShapes"
  - May implement multiple interfaces
  - Interfaces are not ADTs (abstract data types)