Abstract Classes
Interfaces

CSCI 201
Principles of Software Development

Jeffrey Miller, Ph.D.
jeffrey.miller@usc.edu
Outline

• Abstract Classes
• Interfaces
Abstract Classes

- An abstract class is a way for parent classes to guarantee that child classes provide an implementation for a specific method
  - Consider the Shape example. Even though a Shape does not know how to find the area of a Triangle or Rectangle, it could require that both of those classes implement a getArea() method

- Abstract methods only contain declarations but no implementations
  - Any class that contains an abstract method must be declared abstract

- Abstract classes cannot be instantiated since not all of the methods have implementations

- Any class that inherits from an abstract class must implement all of the abstract methods or declare itself abstract
  - When a class implements an abstract method, it is said to override that method
Abstract Class Example

1 abstract class Parent {
2    public abstract int meth1();
3    public int meth() {
4        return 10;
5    }
6 }
7
8 class Child extends Parent {
9    public int meth1() {
10       return 20;
11    }
12 }
13
14 public class Test {
15   public static void main(String [] args) {
16      Child c = new Child();
17      System.out.println(c.meth());
18      System.out.println(c.meth1());
19      Parent p = new Parent();
20      System.out.println(p.meth());
21      System.out.println(p.meth1());
22   }
23 }
Abstract Class Example

abstract class Parent {
    public abstract int meth1();
    public int meth() {
        return 10;
    }
}

class Child extends Parent {
    public int meth1() {
        return 20;
    }
}

public class Test {
    public static void main(String [] args) {
        Child c = new Child();
        System.out.println(c.meth());
        System.out.println(c.meth1());
        Parent p = new Child();
        System.out.println(p.meth());
        System.out.println(p.meth1());
    }
}
Outline

• Abstract Classes
• Interfaces
Interfaces

- An interface is similar to a class, but there are no method implementations in it (not even inherited)
- When a class implements an interface, it must implement all of the methods in the interface
  - If it doesn’t implement all of the methods, it has then inherited an abstract method, so the class must be declared abstract
- A class can implement as many interfaces as it wants
  - This is how Java deals with supporting something similar to multiple inheritance
  - This is different than multiple inheritance though. How?
    - If the same method is inherited from more than one interface, there is no implementation, so there is no confusion
- If interfaces inherit from other interfaces, they will extend them
In C++, we had a problem with multiple inheritance when the same function was implemented in two different branches of the hierarchy.

It doesn’t matter which one gets called if the Telephone and Email Reader classes only contained function definitions and the only implementation was in IPhone.

```cpp
IPhone ip;
ip.send_email();
```
Interface Example

```java
1  interface Parent {
2     public abstract int meth1();
3     public int meth();
4  }
5
6  abstract class Child implements Parent {
7     public int meth1() {
8         return 20;
9     }
10 }
11
12 class GrandChild extends Child {
13    public int meth() {
14        return 30;
15    }
16 }
17
18 public class Test {
19    public static void main(String [] args) {
20       GrandChild gc = new GrandChild();
21       System.out.println(gc.meth());
22       System.out.println(gc.meth1());
23       Child c = new Child();
24       System.out.println(c.meth());
25       System.out.println(c.meth1());
26    }
27 }
```
interface Parent {
    public abstract int meth1();
    public int meth();
}

abstract class Child implements Parent {
    public int meth1() {
        return 20;
    }
}

class GrandChild extends Child {
    public int meth() {
        return 30;
    }
}

public class Test {
    public static void main(String[] args) {
        GrandChild gc = new GrandChild();
        System.out.println(gc.meth());
        System.out.println(gc.meth1());
        Child c = new GrandChild();
        System.out.println(c.meth());
        System.out.println(c.meth1());
    }
}
interface Parent {
  public abstract int meth1();
  public int meth();
}

abstract class Child implements Parent {
  public int meth1() {
    return 20;
  }
}

class GrandChild extends Child {
  public int meth() {
    return 30;
  }
}

public class Test {
  public static void main(String [] args) {
    GrandChild gc = new GrandChild();
    System.out.println(gc.meth());
    System.out.println(gc.meth1());
    Parent p = new GrandChild();
    System.out.println(p.meth());
    System.out.println(p.meth1());
  }
}