Lecture 10: Interfaces

AITI Nigeria Summer 2012
University of Lagos.
**Interfaces**

- An interface in Java is special type
- A class with only method signatures
  - Methods have no body
  - Can never create an instance of an interface
- Classes can *implement* the interface
  - A contract: the class will implement all the methods of an interface definition
Example:
A General Sorting Method

- Create a general sorting method that works on Arrays of any class
  - Each class implements the *MyComparable* interface

- The interface allows two objects to be compared

- `obj1.compareTo(obj2)`
  - return 1 if `obj1 > obj2`
  - 0 if `obj1 == obj2`
  - -1 if `obj1 < obj2`
Example: MyComparable

public interface MyComparable {
    public int compareTo(Object obj);
}

public static void sort(MyComparable[] array) {
    for (int i = array.length; i >= 1; i--) {
        // find the maximum index in the array [0..i-1]
        int maxIndex = i - 1;
        for (int j = 0; j < i; j++) {
            if (array[j].compareTo(array[maxIndex]) == 1) {
                maxIndex = j;
            }
        }
        // Replace last element with maximum value indexed at maxIndex
        MyComparable temporary = array[i - 1];
        array[i - 1] = array[maxIndex];
        array[maxIndex] = temporary;
    }
}
Example: Bank Account

• Compare bank accounts based on balance
• Bank account with greater balance is greater

public class BankAccount implements MyComparable {
    private double balance;
    ...
}
Example: Bank Account

```java
public class BankAccount implements MyComparable {

    public int compareTo(Object obj) {
        if (obj instanceof BankAccount) {
            BankAccount ba = (BankAccount) obj;
            if (this.balance > ba.balance)
                return 1;
            else if (this.balance < ba.balance)
                return -1;
            else return 0;
        } else //error
    }
}
```
Interfaces

• An interface type can be used just like any other type
  – return type of method
  – argument type of method
  – array of interface type

• One cannot create an object of an interface:
  `new Comparable();`

• All methods of an interface are public
Callbacks

• Executable code passed as argument to another class
• The class calls the code when an event happens
• Examples:
  – Call a method when an SMS message is received
  – Call a method when a user presses a button on a phone (J2ME, Android)
Callbacks

**Your class**

- **Constructor:** Register your method
- **callback method:** Process the event

**Library Code**

- Register callback
- Wait for event
  - Event: call callback method

---

1. Register your method
2. Process the event
Interfaces and Callback

• Interface defines the method that will be called when the event happens
  – Defines the arguments you are passed
• You create a class that implements the method
  – The code you want to execute when the event happens
• Must register the callback first
Callback Example

• A class is implemented that receives an SMS message and calls a callback
  – Class Name: Gateway.java
  – Register callback:
    void setInboundNotification(IInboundMessageNotification);

• Interface:
  public Interface IInboundMessageNotification {
    public void process(String message);
  }
public class ProcessMessage
    implements IInboundNotification {

    public ProcessMessage() {
        Gateway = new Gateway();
        gateway.setInboundNotification(this);
    }

    public void process(String message) {
        System.out.println(message);
    }
}
Differences from Inheritance

• The interface does not define any default behavior to inherit
  – Empty definitions in the interface

• The methods must be completed by the implementing class

• A class can implement multiple interfaces
Inheritance or Interface?

• Inheritance:
  – When you want to promote code reuse
  – A subclass is a refinement of superclass
  – A class can only have one superclass

• Interface
  – More general contracts than inheritance
    • Comparable, Writeable, process message, process button-press
  – When you want to define a method contract
  – When you cannot find any reuse in the methods
Relationships

• **has-a**: A class has reference to another class
  – Ex: ContactList *has a* list of Contact

• **is-a**: A class inherits from another class
  – Ex: Person *is a* Contact

• **implements**: A class defines the methods of an Interface
  – BankAccount *implements* Comparable