More on Abstraction in Java

More on Interfaces

Produced by: Ms. Mairead Meagher
Dr. Siobhán Drohan
Siobhán Roche

Ollscoil Teicneolaíochta an Oirdheiscirt
South East Technological University

Department of Computing and Mathematics
http://www.setu.ie/

Topic List

– Interfaces define Types.

– Recap of Social Network V8.0 – Abstract Classes and Methods

– An Interface Example – Social Network V9.0.

– Naming Conventions for Interfaces.

Interfaces Define Types

- Interfaces define Types
  - They define common behaviour i.e. methods
  - Can be used to promote design to a higher level of abstraction
  - Can be used where multiple implementations of one abstraction are envisaged
  - Implementing classes are subtypes of the interface type.

- Classes can implement one or more Interfaces as appropriate i.e. have more than one type.

Interfaces Impose Types

- If a variable is declared as an Interface type:

  IMammal dog;

```
interface IMammal
{
    void eat();
    void travel();
}
```

Interfaces Impose Types

- If a variable is declared as an Interface type:

  IMammal dog;

- Then any instance of any class that implements that Interface can be assigned to that variable.

  IMammal dog = new Mammal();

```
interface IMammal
{
    void eat();
    void travel();
}
```

```
public class Mammal implements IMammal{
   public void eat(){
       System.out.println("Mammal eats");
   }

   public void travel(){
       System.out.println("Mammal travels");
   }
}
```

Interfaces Impose Types

- When you define a new interface, you are defining a new **Reference Data Type**.
- You can use interface names anywhere you can use any other data type name.
- If you define a **Reference** variable whose type is an interface e.g.

  IMammal dog;
- any object you assign to it *must* be an instance of a class that implements the interface e.g.

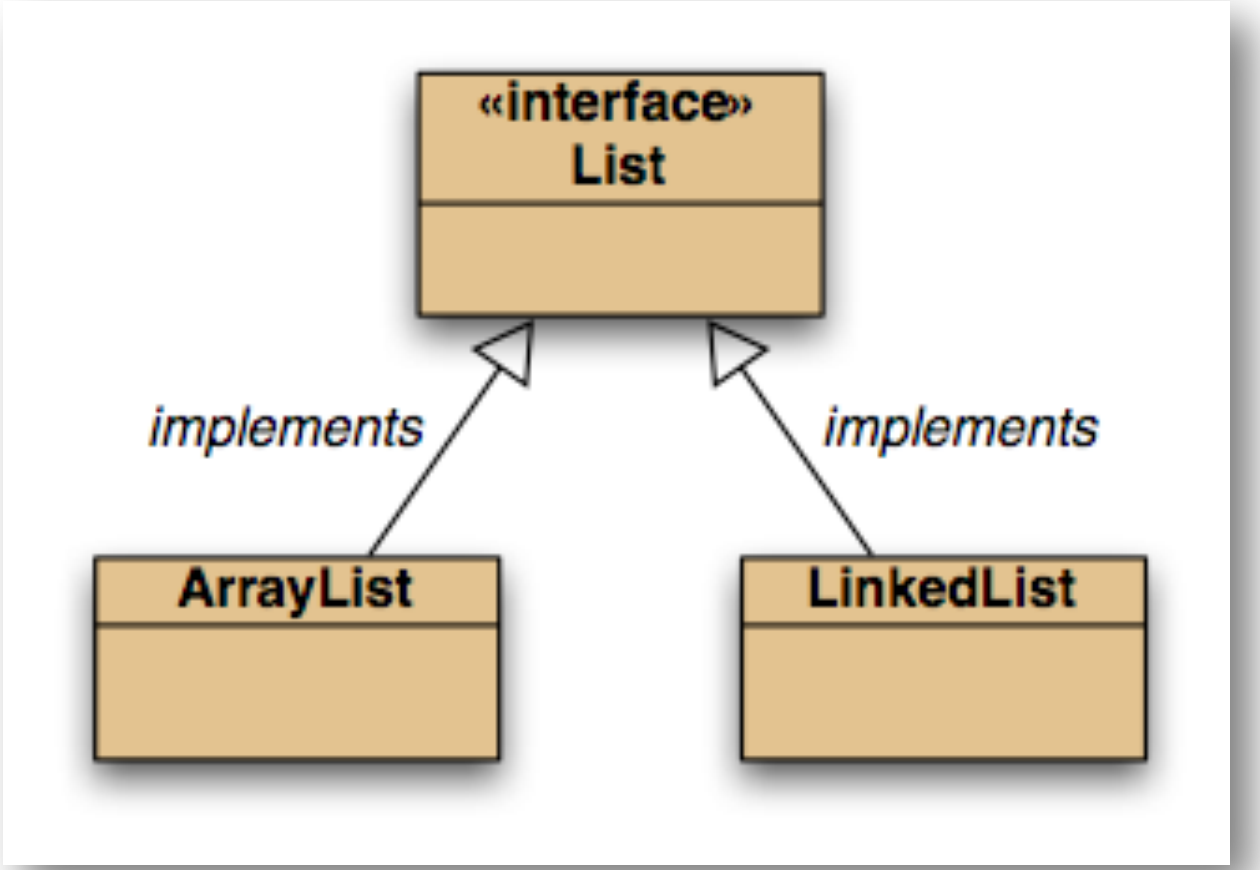
  dog = new Mammal();

```
interface IMammal
{
    void eat();
    void travel();
}
```

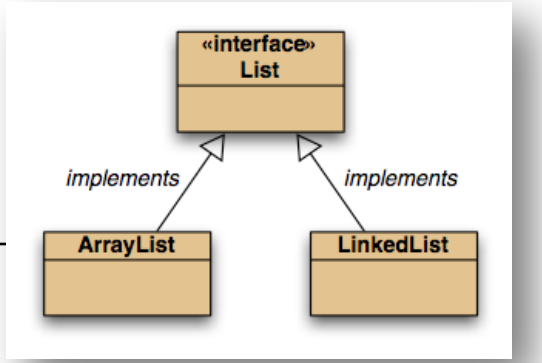
```
public class Mammal implements IMammal{
   public void eat(){
       System.out.println("Mammal eats");
   }

   public void travel(){
       System.out.println("Mammal travels");
   }
}
```

https://docs.oracle.com/javase/tutorial/java/IandI/interfaceAsType.html

Interfaces in Collections Framework



Interfaces in Collections Framework



- ArrayList implements the List interface.

OVERVIEW MODULE PACKAGE CLASS USE TREE DEPRECATED INDEX HELP

PREV CLASS NEXT CLASS FRAMES NO FRAMES ALL CLASSES

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

Module java.base

Package java.util

Class ArrayList<E>

java.lang.Object
  java.util.AbstractCollection<E>
    java.util.AbstractList<E>
      java.util.ArrayList<E>

Type Parameters:

E - the type of elements in this list

All Implemented Interfaces:

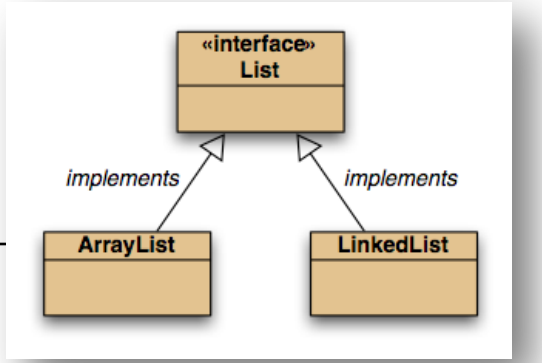
Serializable, Cloneable, Iterable<E>, Collection<E>, List<E>, RandomAccess

Direct Known Subclasses:

AttributeList, RoleList, RoleUnresolvedList

```
public class ArrayList<E>
extends AbstractList<E>
implements List<E>, RandomAccess, Cloneable, Serializable
```

Interfaces in Collections Framework



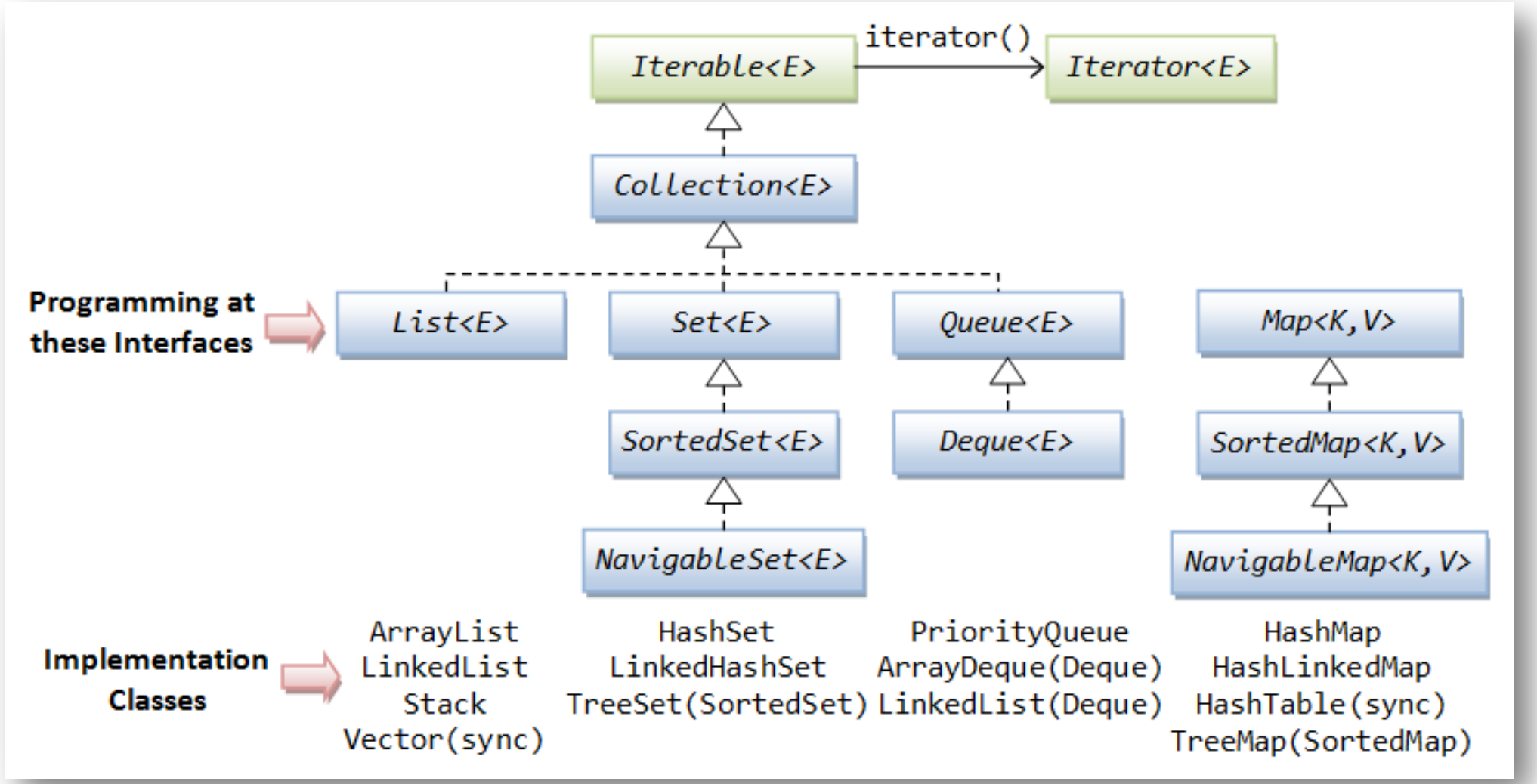
- ArrayList implements the List interface.

- Recall this rule:
  - If you define a reference variable whose type is an interface, any object you assign to it *must* be an instance of a class that implements the interface.

- Applying this rule to a List:

```
List<Product> products = new ArrayList<Product>();
```

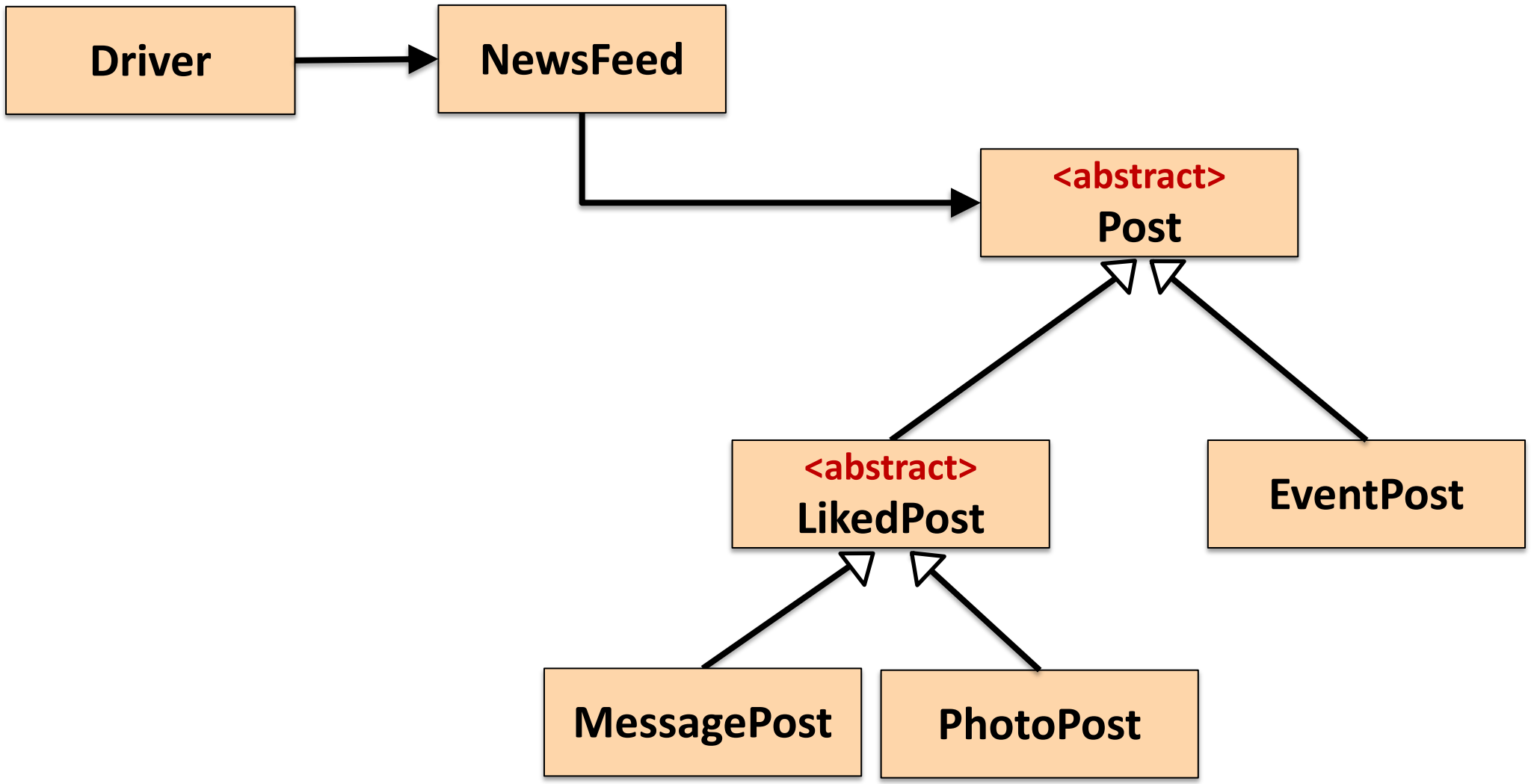
Interfaces in Collections Framework



Topic List

- Interfaces define Types.
- Recap of Social Network V8.0 – Abstract Classes and Methods
- Social Network V9.0 - An Interface Example
- Naming Conventions for Interfaces.

RECAP - Social Network V8.0 – Post and LikedPost Abstract



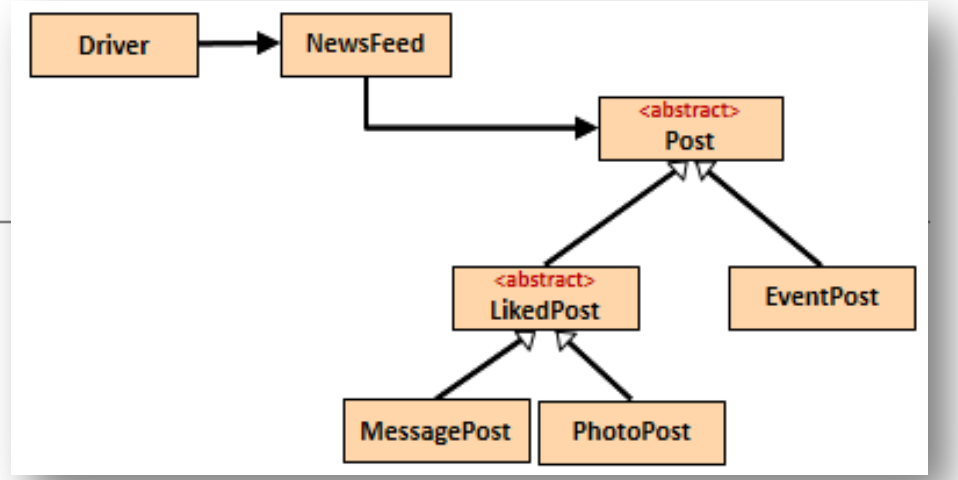
Recap - Abstract classes and methods

- If you wish all subclasses of a class to implement a particular method as part of its code, simply write an abstract method heading in the superclass.

- Each concrete subclass must have this method fully coded.

- Each class that has an abstract method must be made abstract too.

RECAP - Syntax for abstract classes



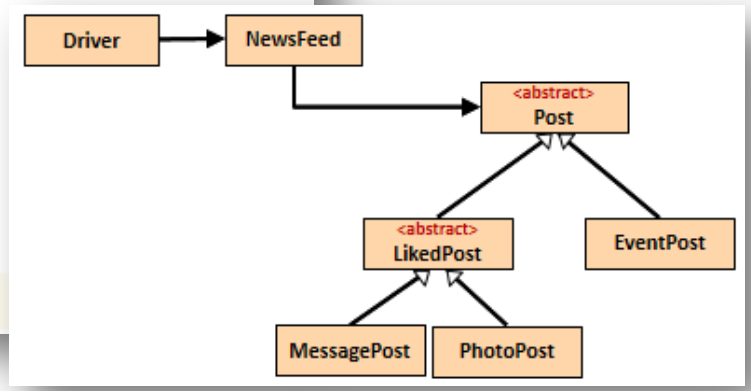

```
//code omitted

public abstract class Post {

   private String author = "";

   public Post(String author) {
      this.author = Utilities.truncateString(author, 10);
   }

  //code omitted
}
```

```
//code omitted

public abstract class LikedPost extends Post {

   private int likes = 0;

   public LikedPost(String author){
      super(author);
   }

  //code omitted
```

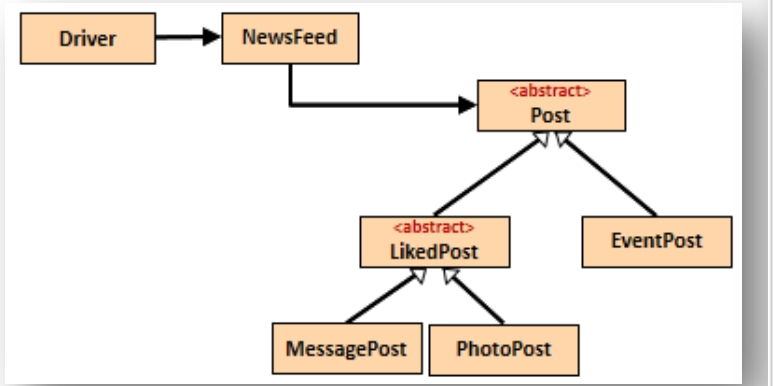
displayCondensed() in Post

Post.java

```
1   package models;
2
3   import utils.Utilities;
4
    1 related problem
5   public abstract class Post {
6
7       private String author = "";
8
9       public Post(String author) { this.author = Utilities.truncateString(author, length: 10); }
12
13      public abstract String displayCondensed();
14
15      public String getAuthor() { return author; }
18
19      public void setAuthor(String author) {...}
24
25      public String display() { return (author + "\n"); }
28
29  }
```



displayCondensed() in EventPost



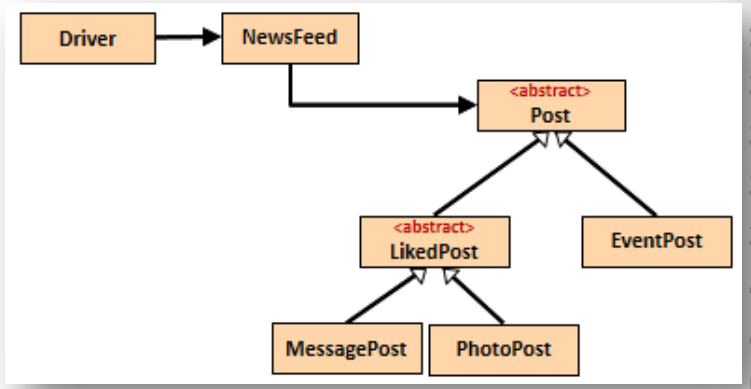
Post.java | EventPost.java

```
1   package models;
2
3   import utils.Utilities;
4
5   public class EventPost extends Post {
6
7       private String eventName = "";
8       private double eventCost = 0;
9
10      public EventPost (String author, String eventName, double eventCost){...}
15
16      @Override
17      public String displayCondensed() {
18          return super.getAuthor() + ": Event(" + eventName + ", €" + eventCost + ").";
19      }
20
21      public String getEventName() { return eventName; }
24
25      public void setEventName(String eventName) {...}
30
31      public double getEventCost() { return eventCost; }
34
35      public void setEventCost(double eventCost) {...}
40
41      public String display() {...}
51
52  }
```

displayCondensed() in PhotoPost

PhotoPost.java

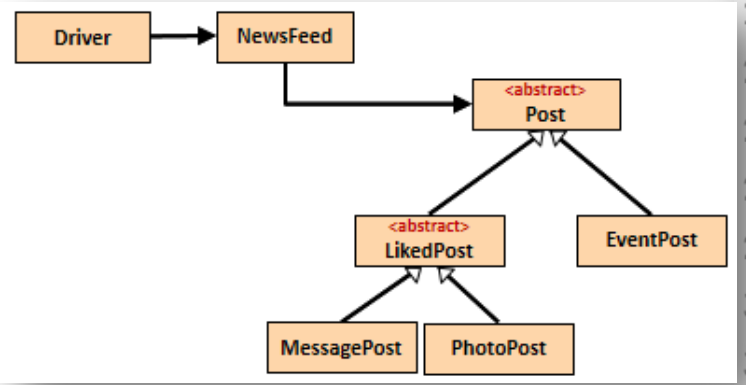
```
1   package models;
2
3   import utils.Utilities;
4
5   public class PhotoPost extends LikedPost{
6
7       private String caption = "";
8       private String filename = "";
9
10      public PhotoPost(String author, String caption, String filename) {...}
15
16      @Override
17      public String displayCondensed() {
18          return super.displayCondensed() + ": Photo(" + caption +  ", " + filename + ")";
19      }
20
21      public String getCaption() { return caption; }
24
25      public void setCaption(String caption) {...}
30
31      public void setFilename(String filename) {...}
36
37      public String getFilename() { return filename; }
40
41      public String display() {...}
54  }
```



displayCondensed() in MessagePost

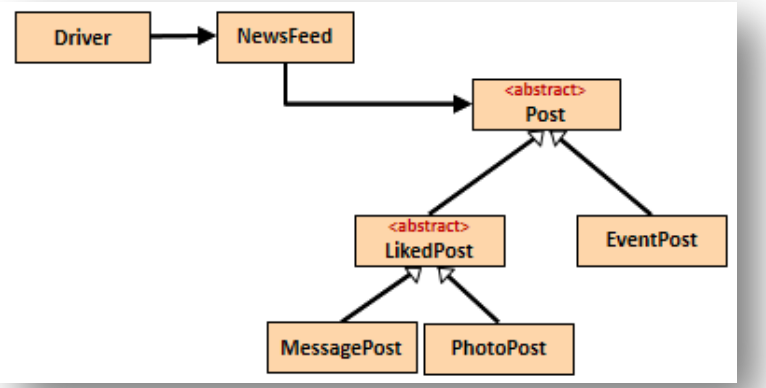
MessagePost.java

```
1   package models;
2
3   import utils.Utilities;
4
5   public class MessagePost extends LikedPost{
6
7       private String message = "";
8
9       public MessagePost(String author, String message) {...}
13
14      @Override
15      public String displayCondensed() {
16          return super.displayCondensed() + ": Message(" + message +  ")";
17      }
18
19      public String getMessage() { return message; }
22
23      public void setMessage(String message) {...}
28
29      public String display() {...}
37  }
38
```



displayCondensed() in NewsFeed

```
==>>1
List of All Posts are:
0: Siobhan (0 likes) : Message(My message is Hi There)
1: Mairead (0 likes) : Photo(Hi all, photo7hello.jpg)
2: Siobhan: Event(Coding Event, €5.0)
```



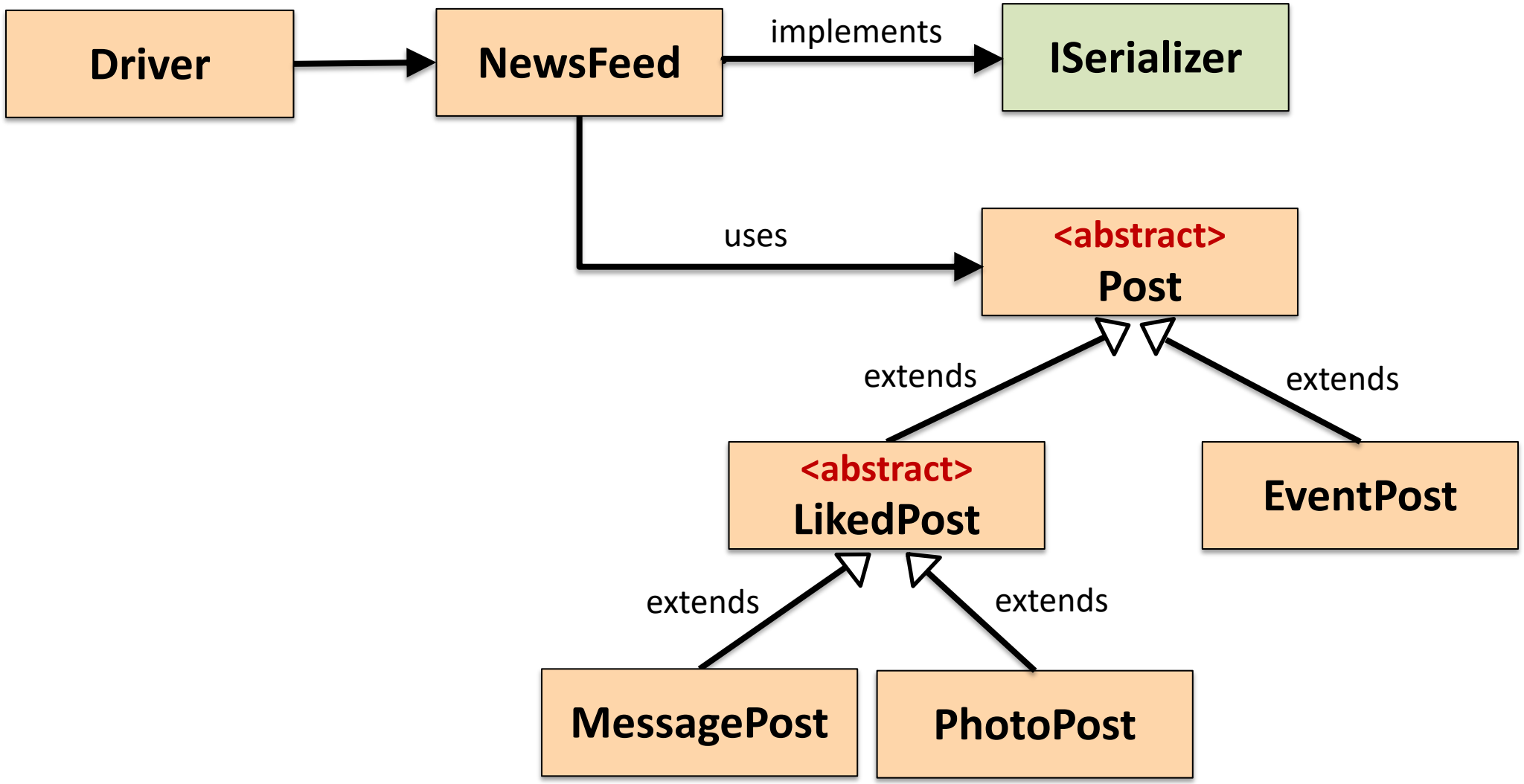
NewsFeed.java

```
1   package controllers;
2
3   import ...
12
13  public class NewsFeed {
14
15      private ArrayList<Post> posts;

7       public NewsFeed() { posts = new ArrayList<Post>(); }

1       public boolean addPost(Post post) { return posts.add(post); }

25      public String show() {
26          String str = "";
27
28          for(Post post: posts) {
29              str += posts.indexOf(post) + ": " + post.displayCondensed() + "\n";
30          }
31
32          if (str.isEmpty()){
33              return "No Posts";
34          }
35          else {
36              return str;
37          }
38      }
```

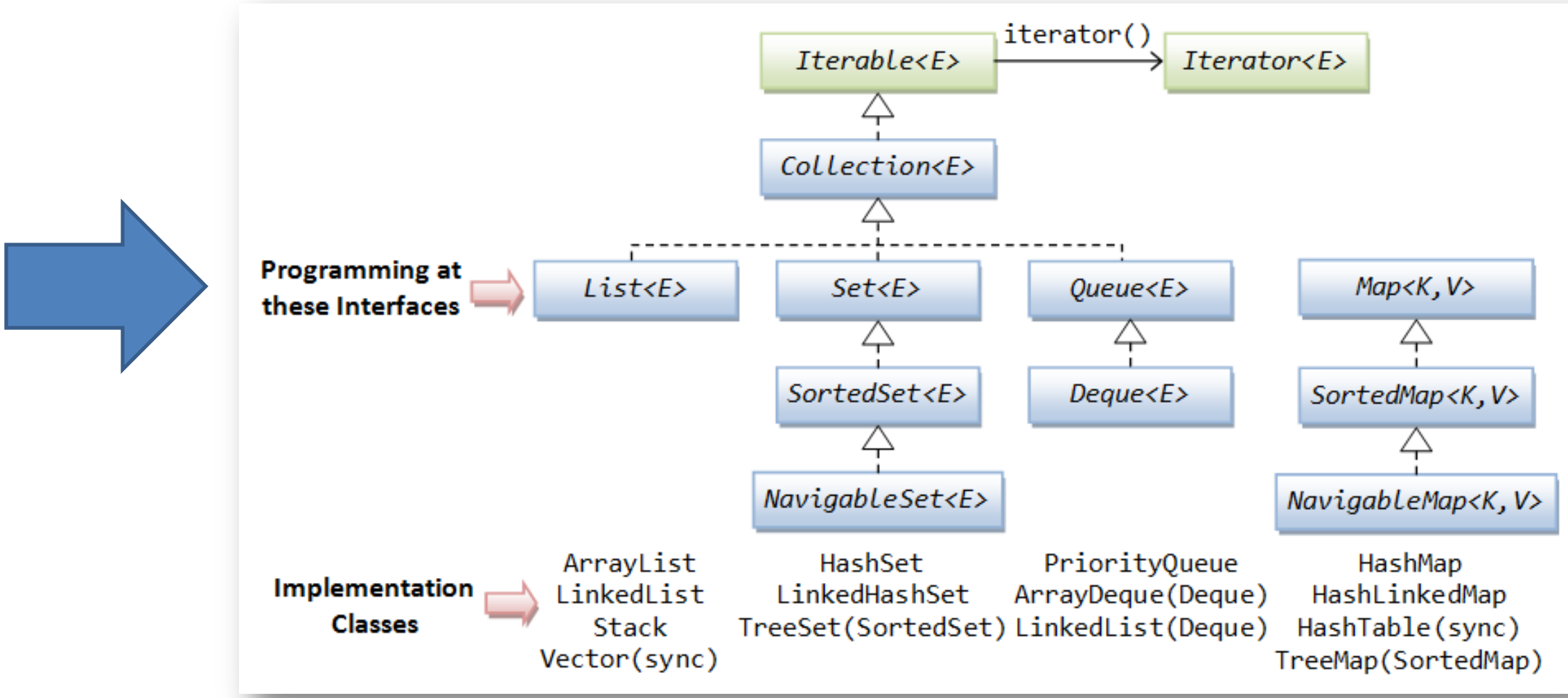
Topic List

- Interfaces define Types.
- Recap of Social Network V8.0 – Abstract Classes and Methods
- Social Network V9.0 - An Interface Example
- Naming Conventions for Interfaces.

Social Network V9.0 – An Interface Example



Interfaces in Collections Framework



```java
import java.util.ArrayList;

public class NewsFeed {

    private ArrayList<Post> posts;

    public NewsFeed() {
        posts = new ArrayList<Post>();
    }

    public ArrayList<Post> getPosts() {
        return posts;
    }

    public void setPosts(ArrayList<Post> posts) {
        this.posts = posts;
    }
```

Add a getter and a setter for the posts ArrayList in NewsFeed.

```java
import java.util.ArrayList;
import java.util.List;

public class NewsFeed {

    private List<Post> posts;

    public NewsFeed() {
        posts = new ArrayList<>();
    }

    public List<Post> getPosts() {
        return posts;
    }

    public void setPosts(List<Post> posts) {
        this.posts = posts;
    }
```

We will now refactor our app so that our **posts** ArrayList is defined using interfaces instead of concrete classes.

Interfaces in Collections Framework

Your code is more **maintainable** when you define collections like this:

- ArrayLists:
  List<Post> posts = new ArrayList<Post>();

- Maps:
  Map<String, String> addresses = new HashMap<String, String>();

- Sets:
  Set<String> words = new HashSet<String>();

Note: Include this approach in your Assignment 2.

Interfaces in Collections Framework

Why is code more maintainable when ArrayLists are defined like:

```
List<Post> posts = new ArrayList<Post>();
```

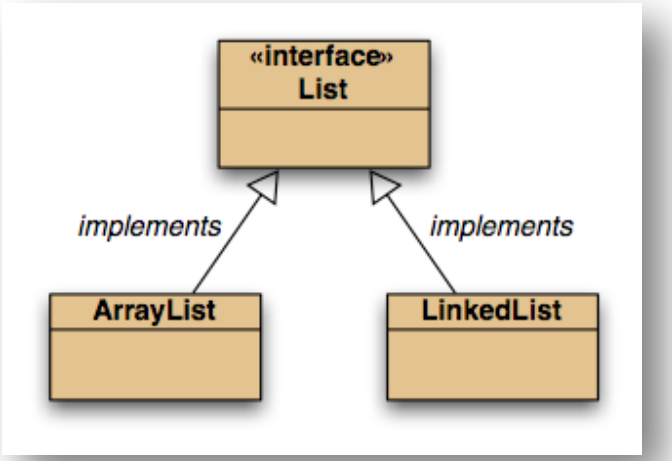
Answer: if we decided, at a later date, that we wanted to use a LinkedList instead of an ArrayList, we only have to make minor changes in the class i.e.

```
new ArrayList<Post>();
```

becomes

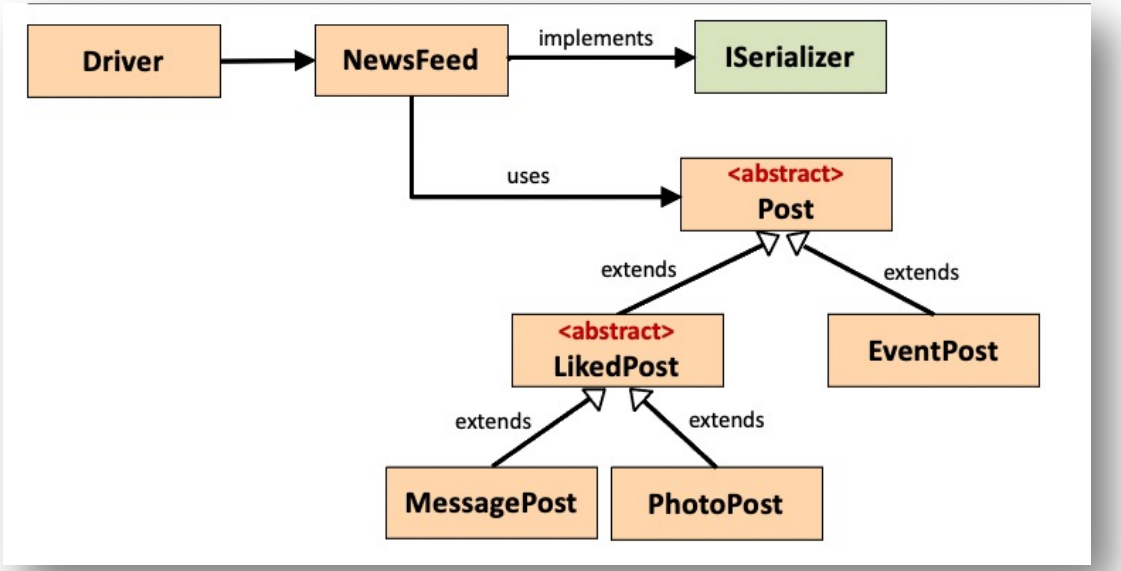
```
new LinkedList<Post>();
```

and `import java.util.LinkedList;`



Social Network V9.0 – Adding ISerializer Interface

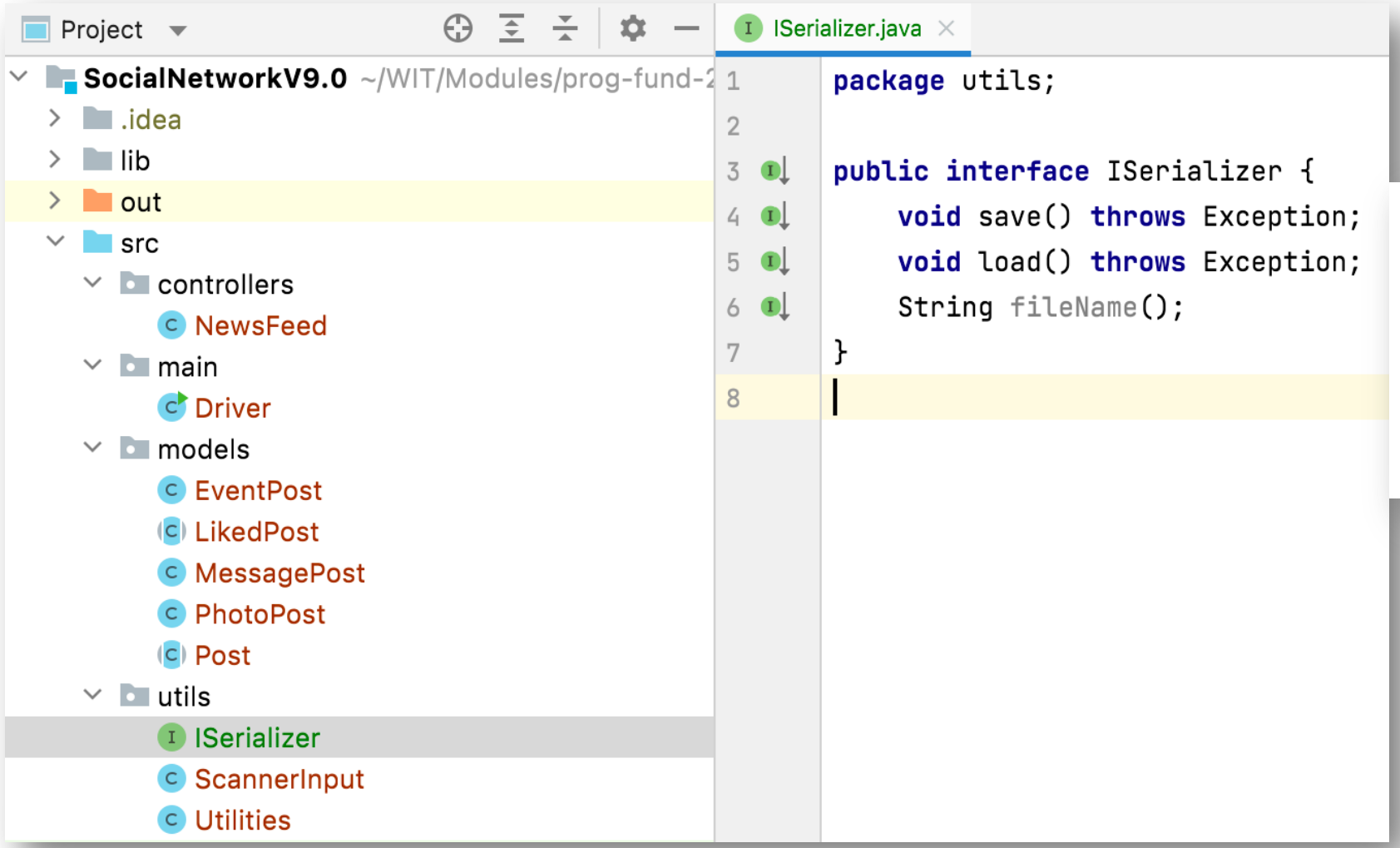
We have used Interfaces in our code (i.e. the List Collection) but we have not written an actual Interface entity ourselves.

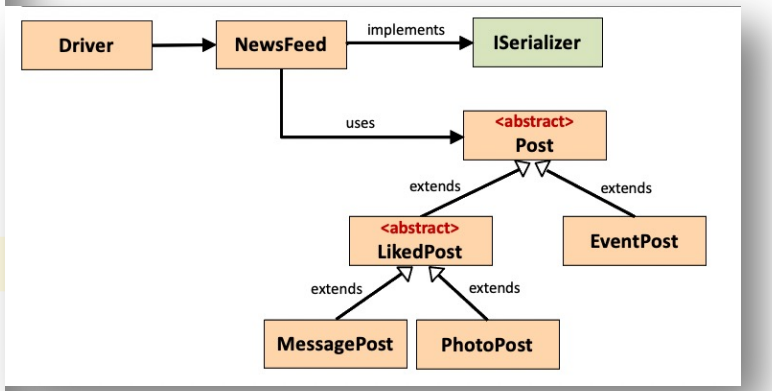


Now we will write an Interface called ISerializer that imposes a design contract on all classes implementing it.

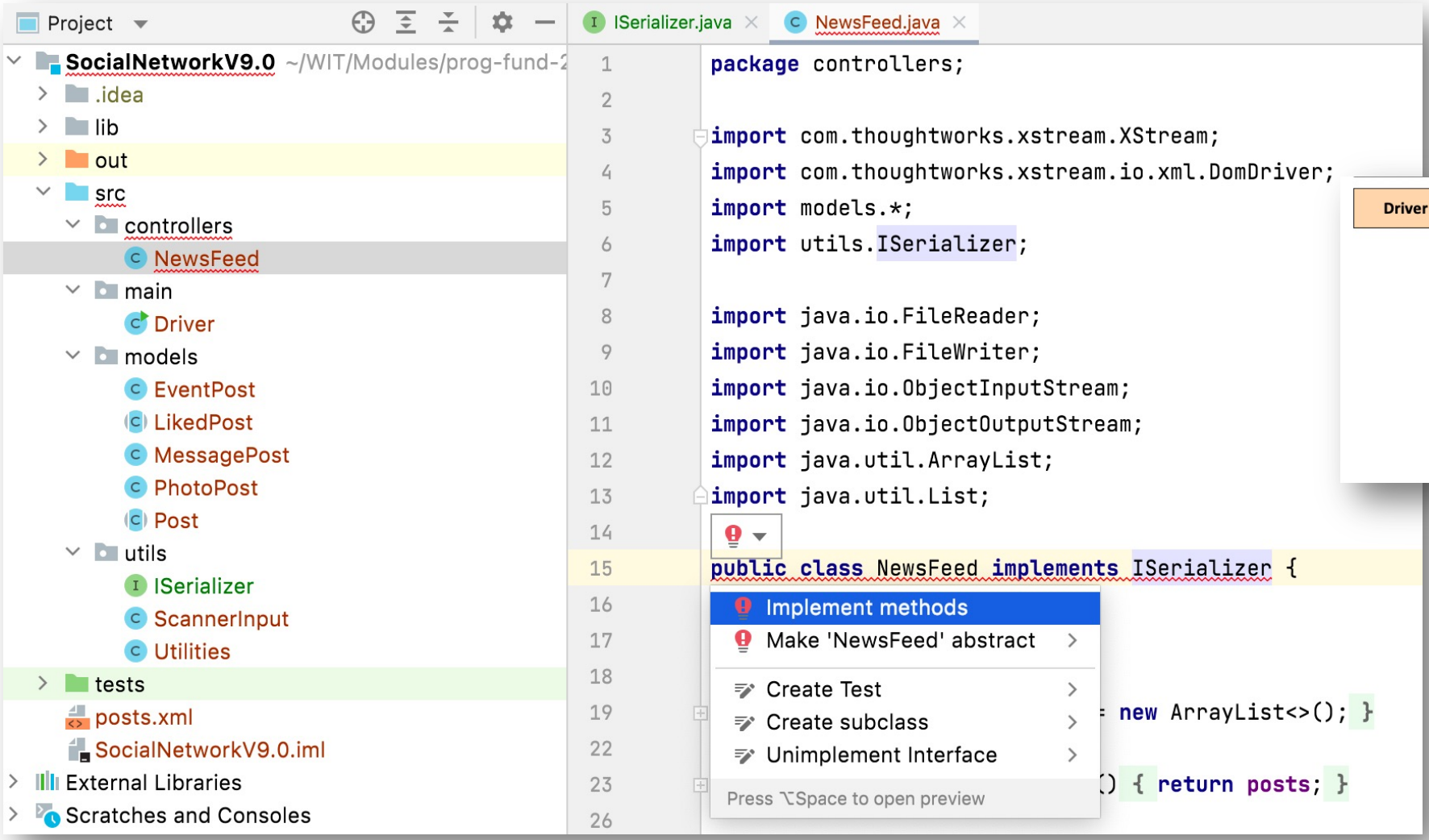
When the NewsFeed class implements this interface, it will be forced to provide an implementation for the abstract methods defined in ISerializer.

Social Network V9.0 – Create ISerializer Interface





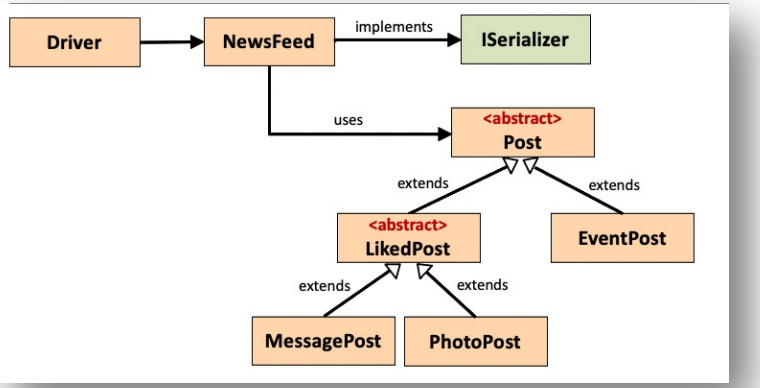
Social Network V9.0 – Implement ISerializer Interface



Social Network V9.0 – Implement ISerializer methods

Already Added

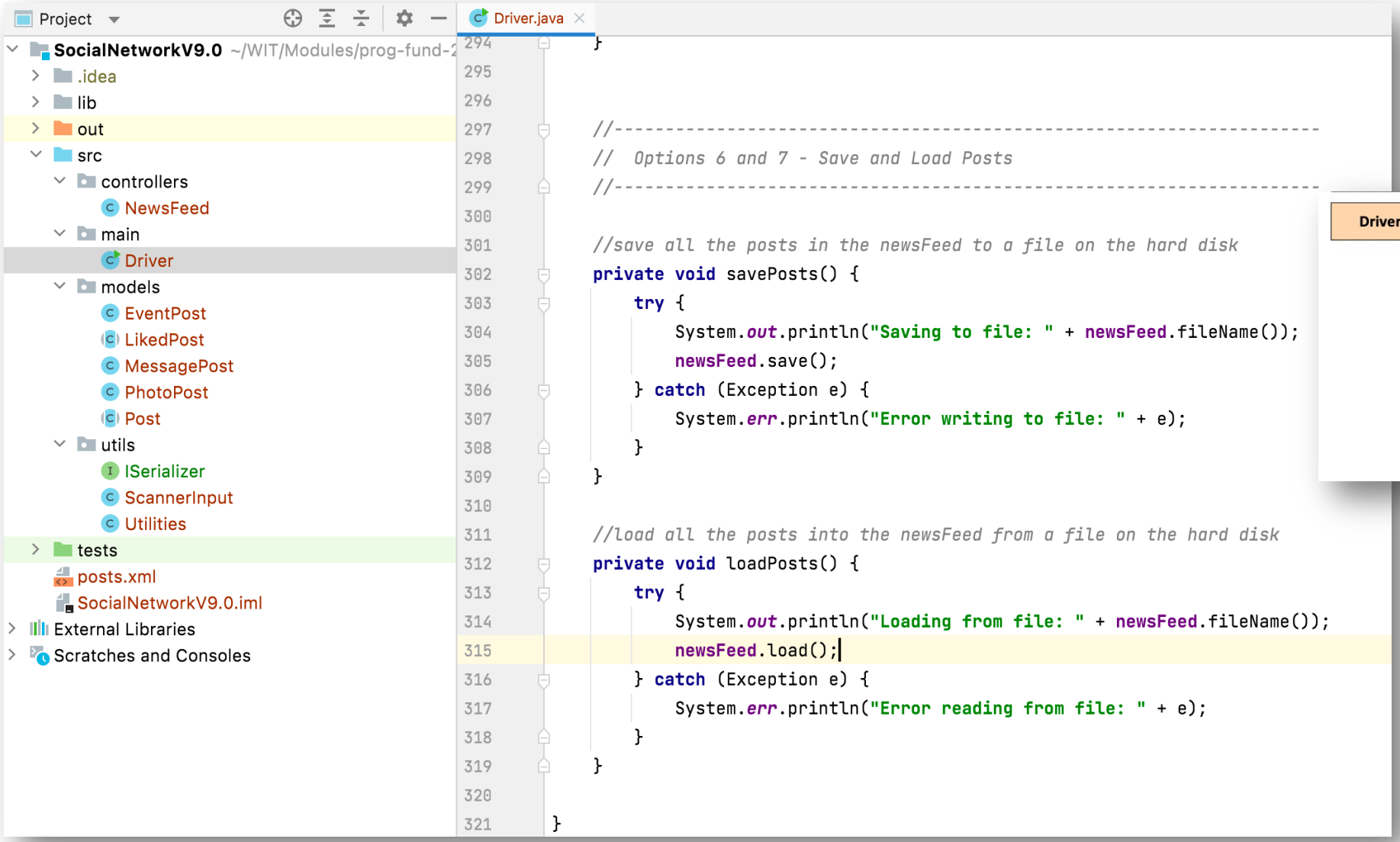
```
/** The load method uses the XStream component to read a
/unchecked/
public void load() throws Exception {...}

/** The save method uses the XStream component to write
public void save() throws Exception {...}
```

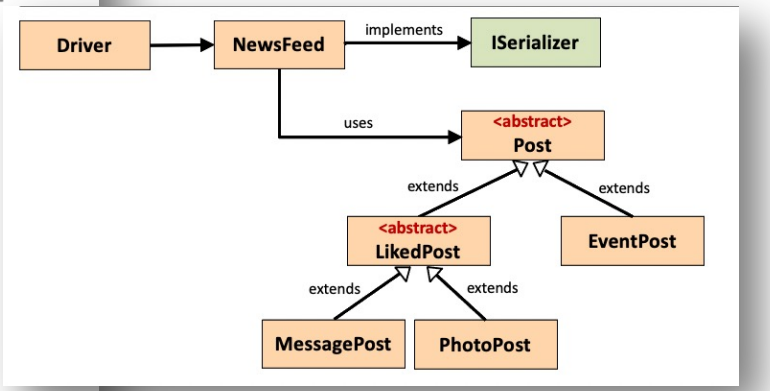


To be Added

```
public String fileName(){
    return "posts.xml";
}
```

Social Network V9.0 – Using ISerializer methods





Topic List

- Interfaces define Types.
- Recap of Social Network V8.0 – Abstract Classes and Methods
- Social Network V9.0 - An Interface Example
- Naming Conventions for Interfaces.

Common Naming Conventions for Interfaces

- Suffix ***able*** is often used for interfaces
  - Cloneable, Serializable, and Transferable
- Nouns are often used for implementing classes names, and I + noun for interfaces
  - Interfaces: IColor, ICar, and IColoredCar
  - Classes: Color, Car, and ColoredCar
- Nouns are often used for interfaces names, and noun+Impl for implementing classes
  - Interfaces: Color, Car, and ColoredCar
  - Classes: ColorImpl, CarImpl, and ColoredCarImpl

Any Questions?