Abstraction

Classes and Methods

Produced by: Dr. Siobhán Drohan
Mairead Meagher
Siobhan Roche

Ollscoil Teicneolaíochta an Oirdheiscirt
South East Technological University

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

Topic List

- Abstract vs Concrete
- Abstract Methods and Classes
- Social Network V7.0 (RECAP)
- Social Network V8.0 (with abstraction)

Abstract vs Concrete

- Abstract
  - Implementation delayed

    → abstract method has no code

    → cannot instantiate an abstract class (it has, by definition "unfinished" methods)

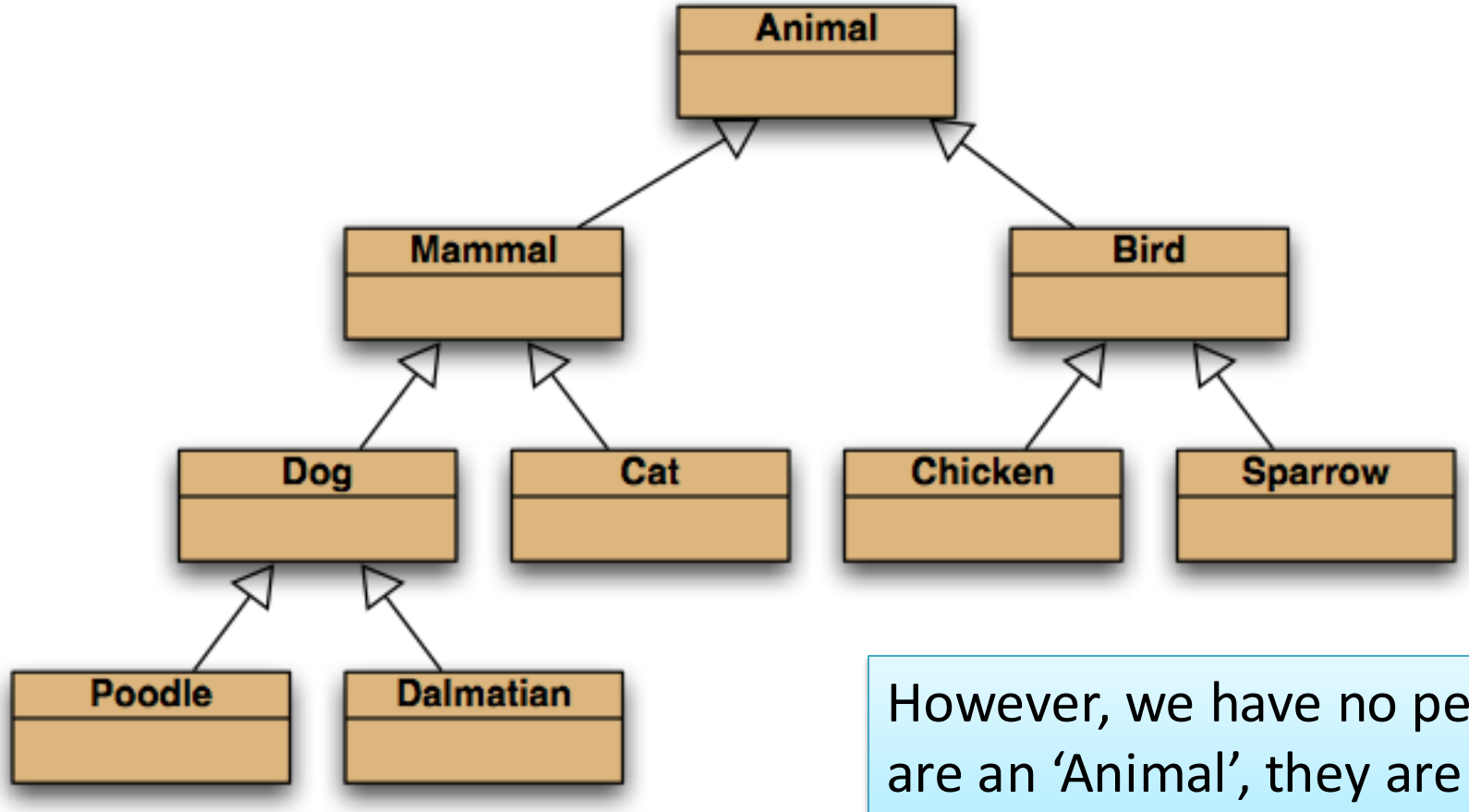
- Concrete
  - Ready to go.
  - Everything up to now has been concrete.

Topic List

- Abstract vs Concrete
- Abstract Methods and Classes
- Social Network V7.0 (RECAP)
- Social Network V8.0 (with abstraction)

Recap: Inheritance hierarchies

"is a" relationship!



However, we have no pets who are an 'Animal', they are either a Dog, a Cat, a Chicken, etc.

Abstract Methods

- Abstract methods have `abstract` in the signature.

- Abstract methods have no body.
  - ‘We promise to write this later. Every (concrete) subclass of this class will have this implemented in the subclass.’

- Abstract methods make the class abstract.
  - Think about why this is?

Abstract Classes

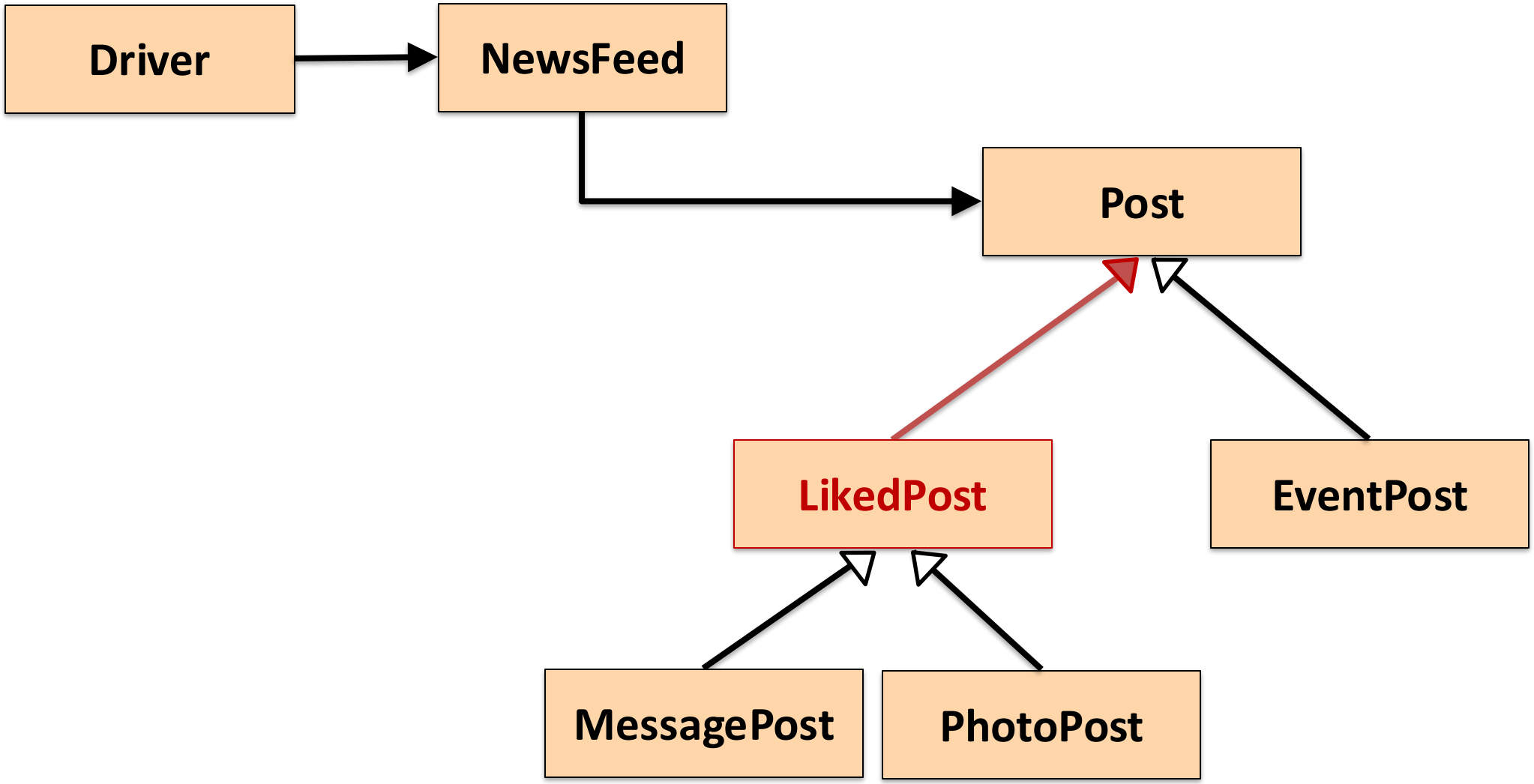
- An abstract class is a class that contains zero or more abstract methods.
- Any class that has an abstract method must be declared abstract.
- Abstract classes cannot be instantiated.
- Abstract classes function as a “base” for subclasses.

  → abstract classes can be subclassed.
- Concrete subclasses complete the implementation.

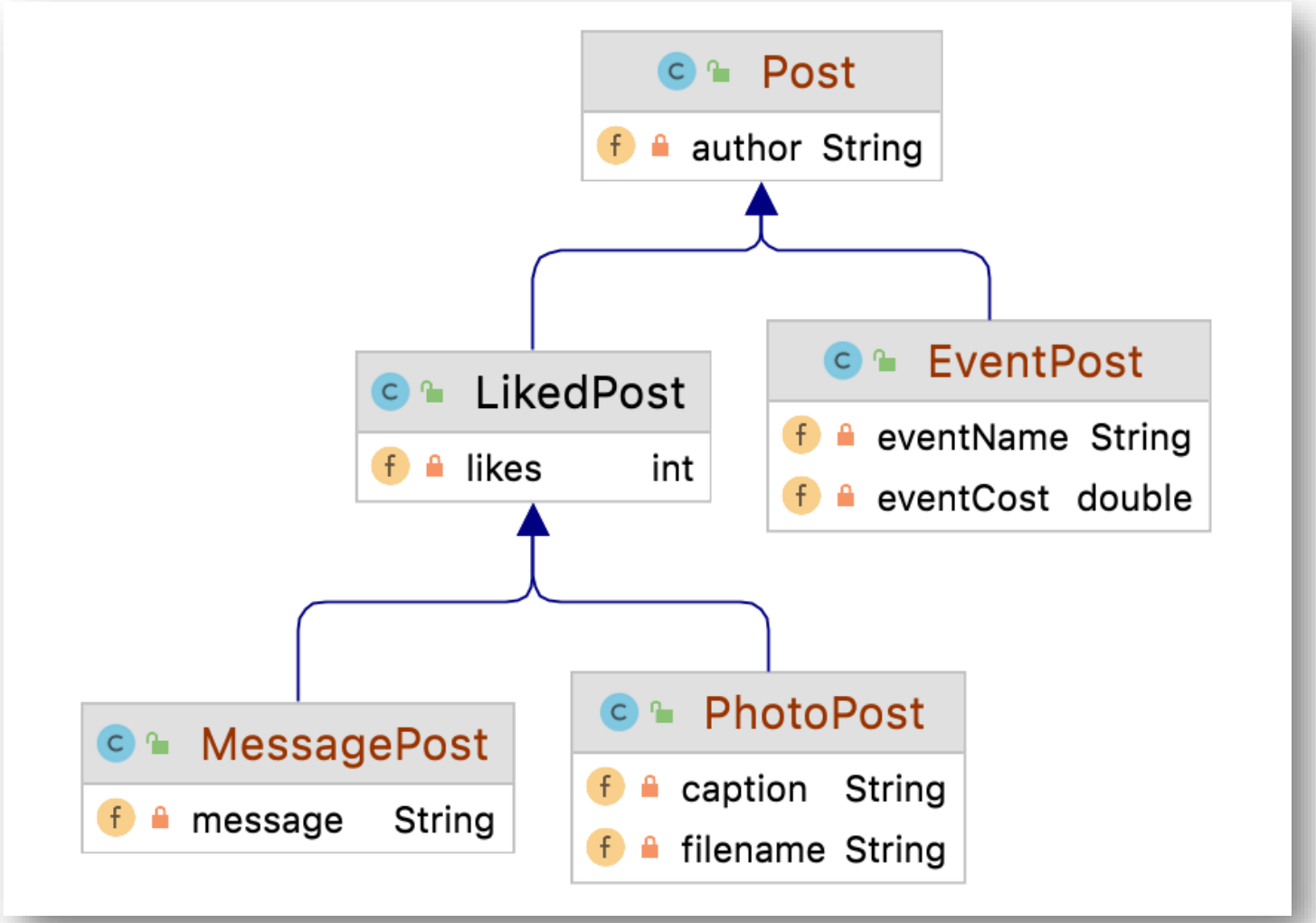
Topic List

- Abstract vs Concrete
- Abstract Methods and Classes
- Network-V7 (RECAP)
- Network-V8 (with abstraction)

RECAP: Social Network V7.0 – Deeper Hierarchy



RECAP: Social Network V7.0 – Deeper Hierarchy



RECAP: Social Network V7.0 – Driver

```
Social Network Menu
   ----------------------
   1) Add a Post
   2) Update a Post
   3) Delete a Post
   4) List Posts
   5) Like / Unlike Pos
   ----------------------
   6) Save Posts
   7) Load Posts
   ----------------------
   0) Exit
==>>
```

```
--------------------------
|   1) Add a Message Post |
|   2) Add a Photo Post   |
|   3) Add an Event Post  |
--------------------------
==>>
```

```
--------------------------
|   1) View ALL Posts     |
|   2) View Message Posts |
|   3) View Photo Posts   |
|   4) View Event Posts   |
--------------------------
==>>
```

Our news feed displays MessagePost, PhotoPost and EventPost objects.

We never create a Post object but our ArrayList is of Post.

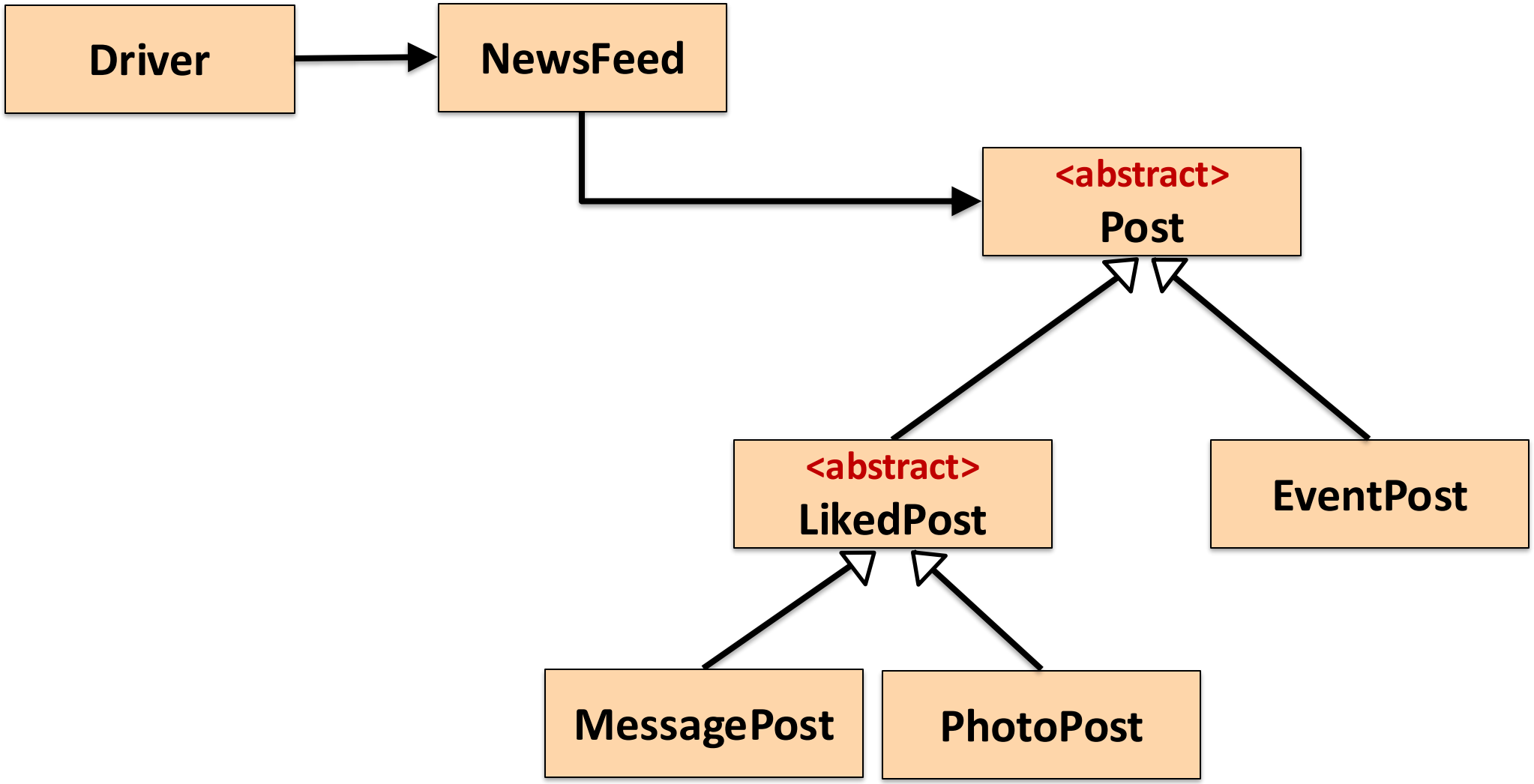
Topic List

- Abstract vs Concrete
- Abstract Methods and Classes
- Social Network V7.0 (RECAP)
- Social Network V8.0 (with abstraction)

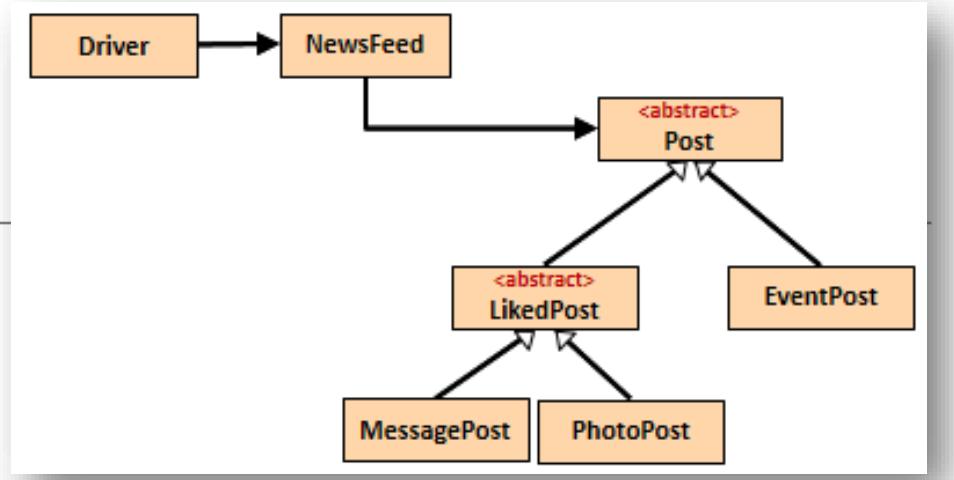
Social Network V8.0 (Post as an abstract class)

- We can never create a ‘post’ object
  - We cannot instantiate one.
- In Post, we define fields, methods that can be used later for all subclasses (using super)
  - e.g. display(), constructor..

Social Network V8.0 – Post and LikedPost Abstract



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
```

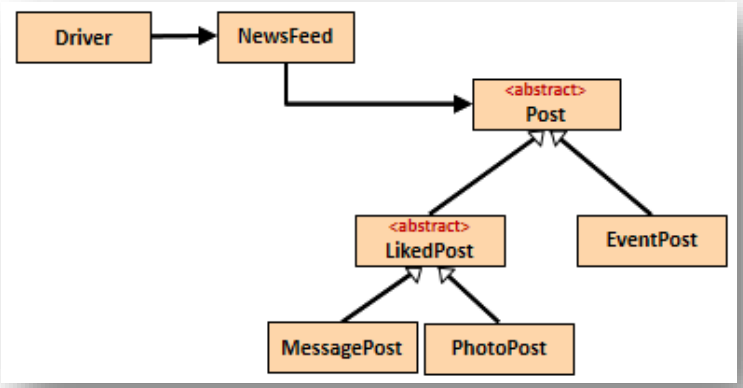
Abstract methods

- 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 superclass.

- Each concrete subclass must have this method fully coded.

displayCondensed() – new abstract method

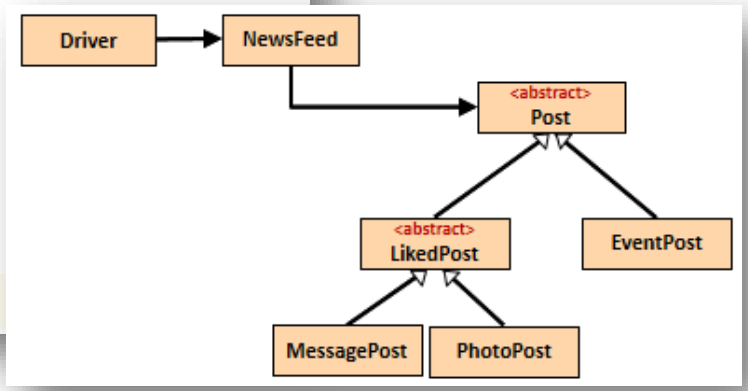
- Currently our display method, displays the contents over a few lines.
- Let's add a new method, displayCondensed(), but this time, condense the display to only ONE line.
- We will add this as an abstract method in Post; this will mean that each concrete class that inherits from Post MUST provide an implementation of it.



displayCondensed() in Post

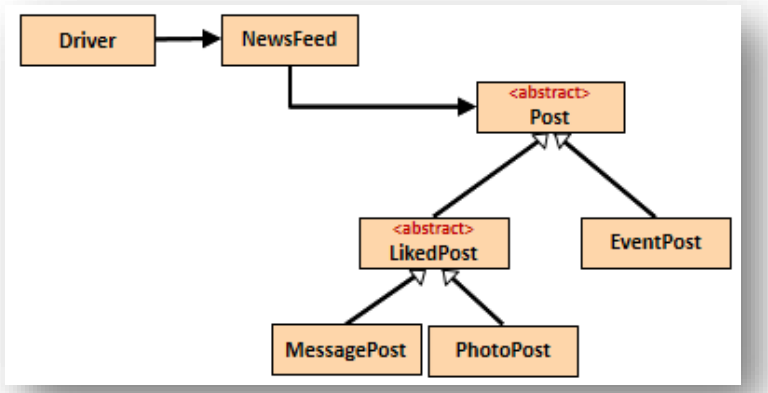
Post.java

```
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

The compiler is complaining that the concrete classes in the hierarchy have no implementation of **displayCondensed()**

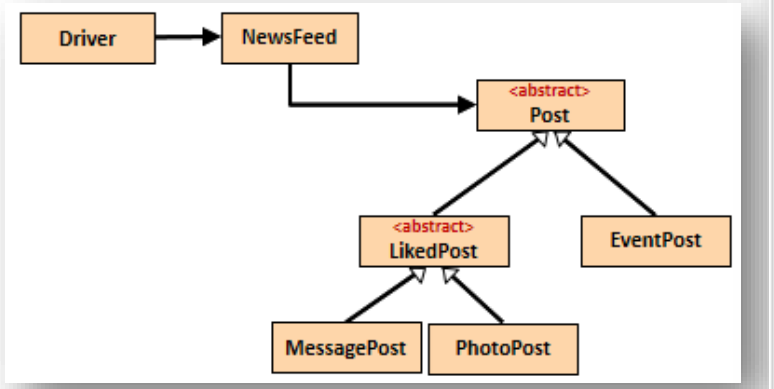


Post.java | EventPost.java

```
1  package models;
2
3  import utils.Utilities;
4
5  public class EventPost extends Post {
6
7
8
9
10
11
12
13      setEventCost(eventCost);
14  }
```

Implement methods
Make 'EventPost' abstract
Create Test
Create subclass
Make 'EventPost' package-private
Press Ctrl+Shift+I to open preview

displayCondensed() in EventPost



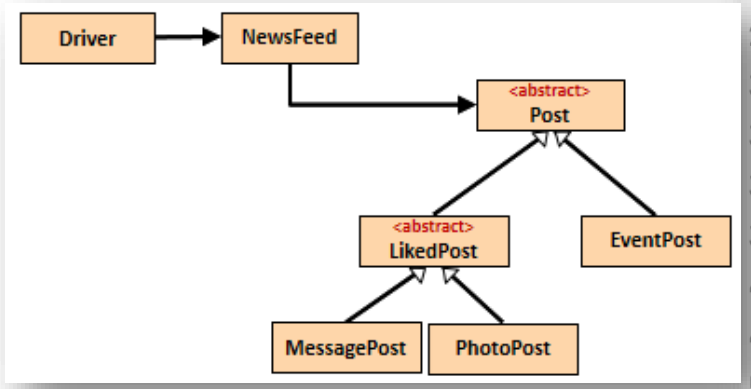
Post.java | EventPost.java

```
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

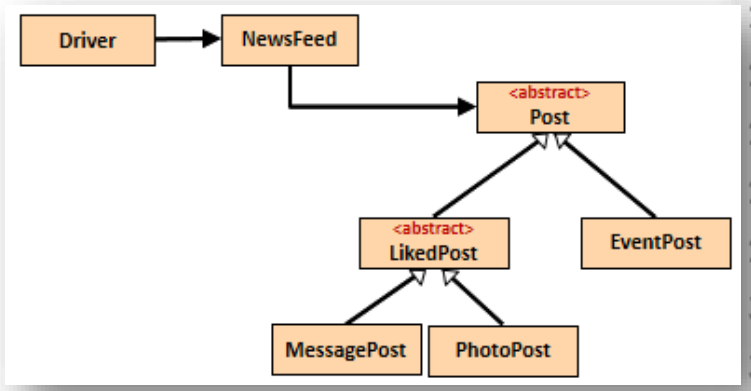
```
1   package models;
2
3   import utils.Utilities;
4
5   public class PhotoPost extends LikedPost{
6
7       private String caption = "";
8       private String filename = "";
9
10      public PhotoPost(String author, String caption, String filename) {...}
15
16      @Override
17      public String 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

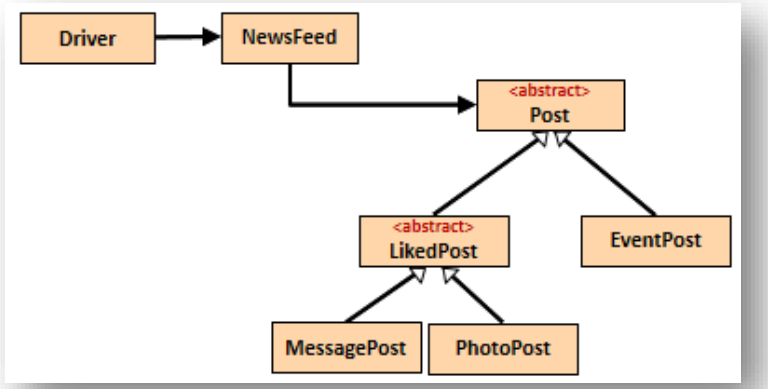
MessagePost.java

```
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;
16
17      public NewsFeed() { posts = new ArrayList<Post>(); }
18
21      public boolean addPost(Post post) { return posts.add(post); }
24
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      }
```

IntelliJ UML Decorators

	class
	abstract class
	private
	public

	method
	abstract method
	static method
	field
	static and final field

Interfaces Topic

- Next, we will look at interfaces which are used when you can see a ‘multiple inheritance’ in your class design.

- Multiple inheritance is not allowed in Java so we use interfaces instead.

Any Questions?