# More on Abstraction in Java

## Deadly Diamond of Death

---

Produced by:
Mairead Meagher
Dr. Siobhán Drohan
Siobhán Roche

Ollscoil Teicneolaíochta an Oirdheiscirt
South East Technological University

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

# RECAP ON 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.

# 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.’

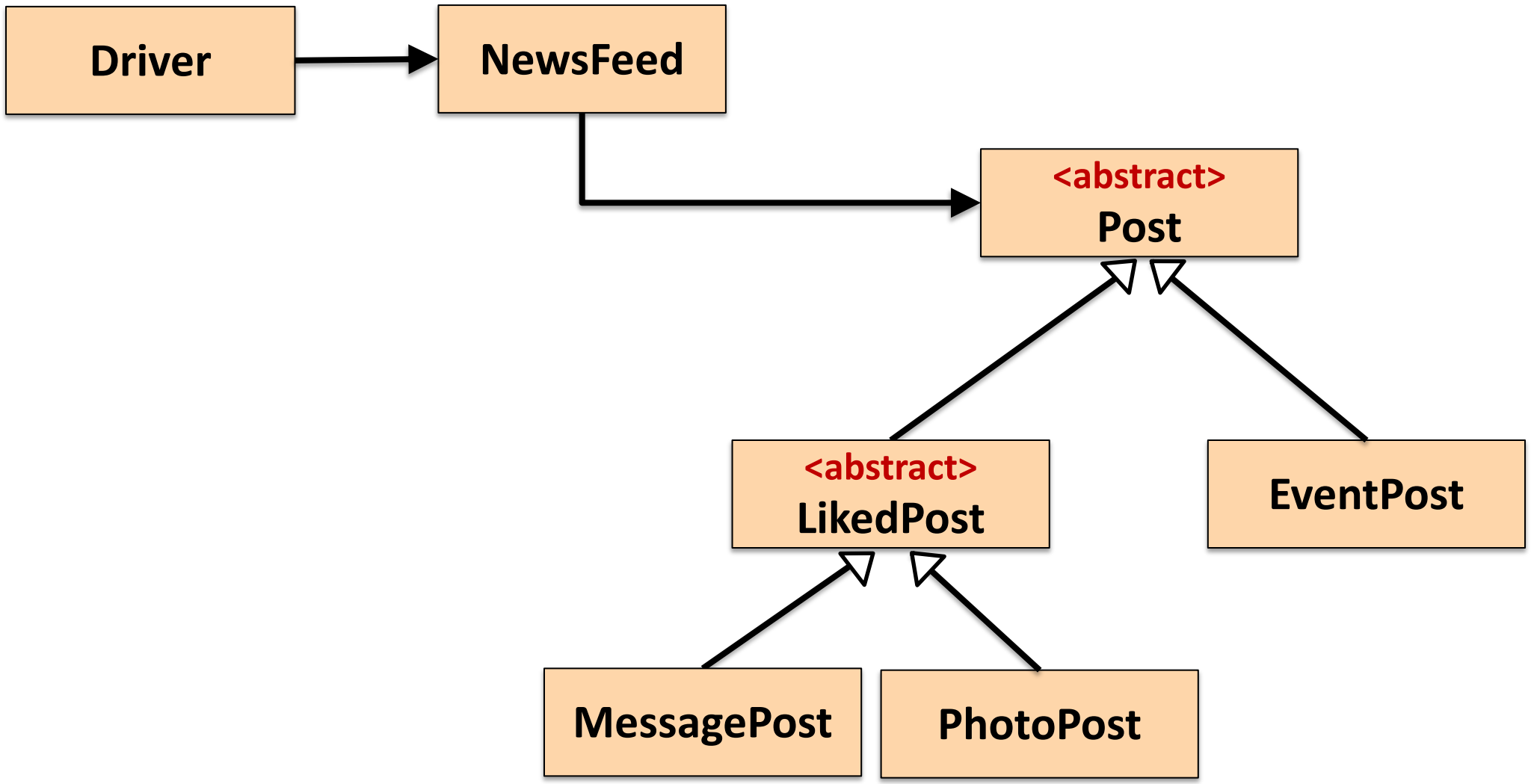
- Abstract methods make the class abstract.
  - Think about why this is?

# Abstract Classes

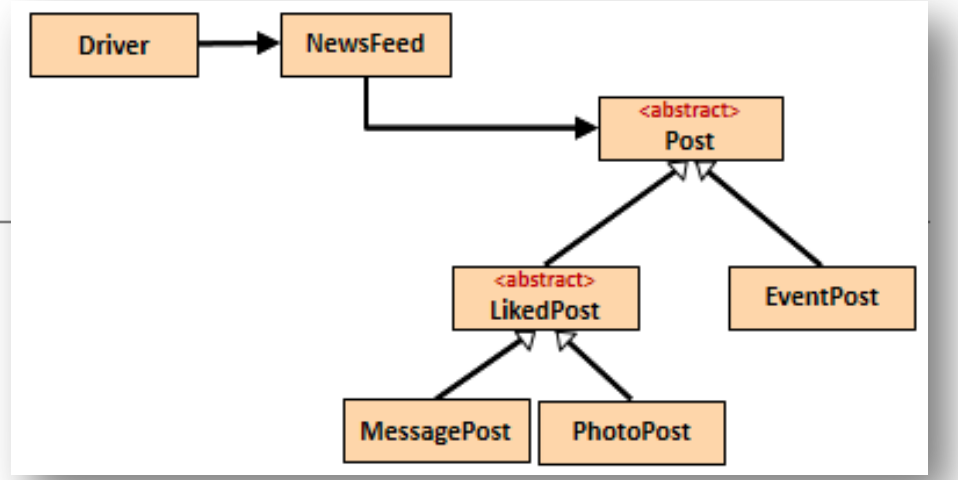
- An abstract class is a class that contains zero or more abstract methods.
- An class that has an abstract method must be declared abstract.
- Abstract classes cannot be instantiated.
- Abstract classes function as a “base” for subclasses.

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

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



# RECAP - Syntax for abstract classes



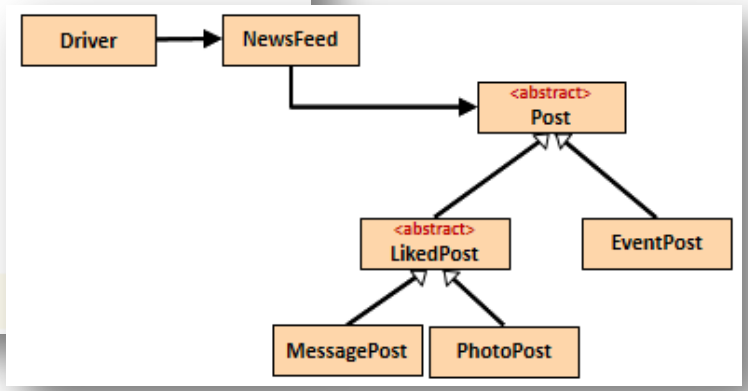

```
//code omitted

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

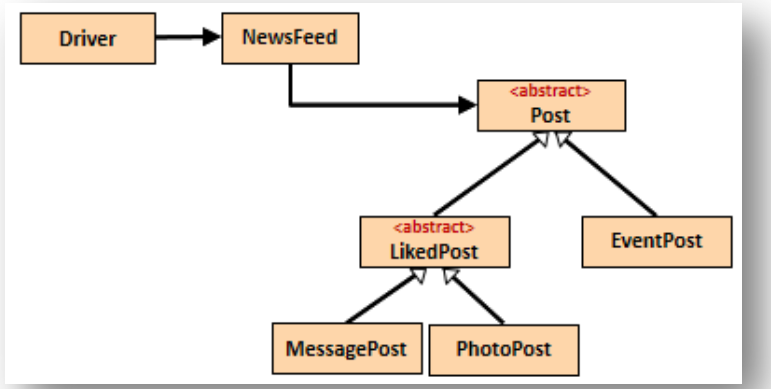
Post.java

```
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

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

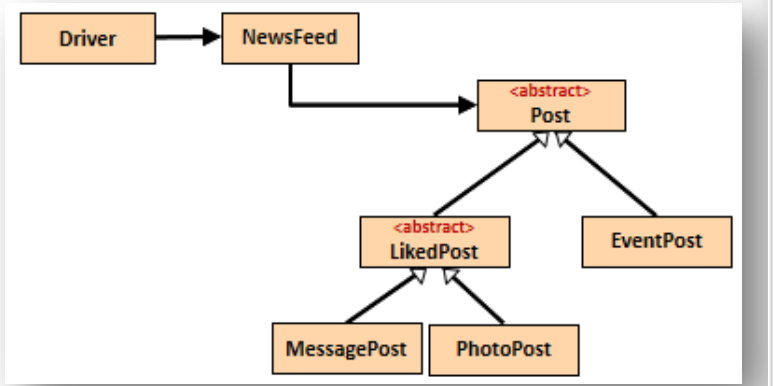



Post.java | EventPost.java

```
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



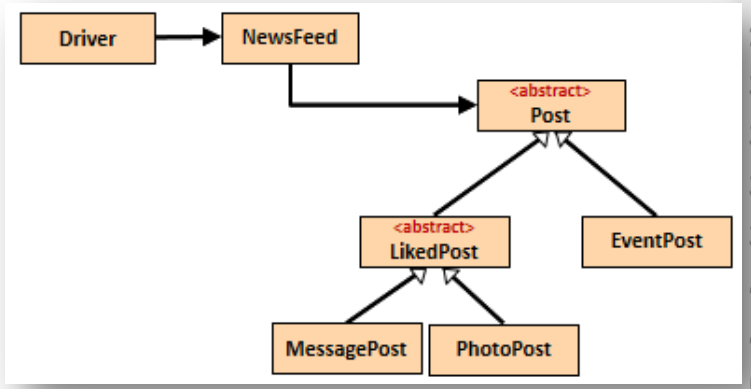

Post.java | EventPost.java

```
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

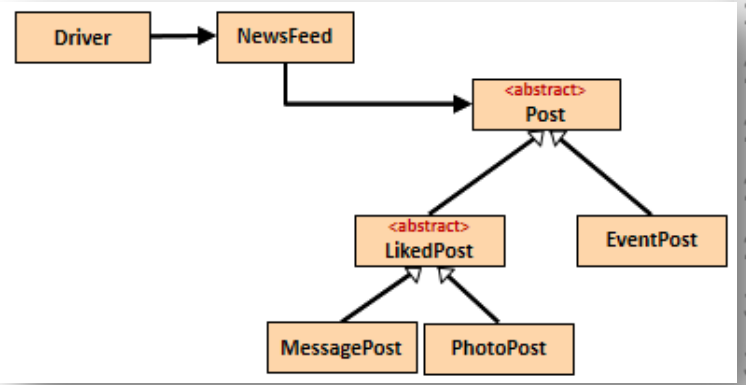
```
1   package models;
2
3   import utils.Utilities;
4
5   public class PhotoPost extends LikedPost{
6
7       private String caption = "";
8       private String filename = "";
9
10      public PhotoPost(String author, String caption, String filename) {...}
15
16      @Override
17      public String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

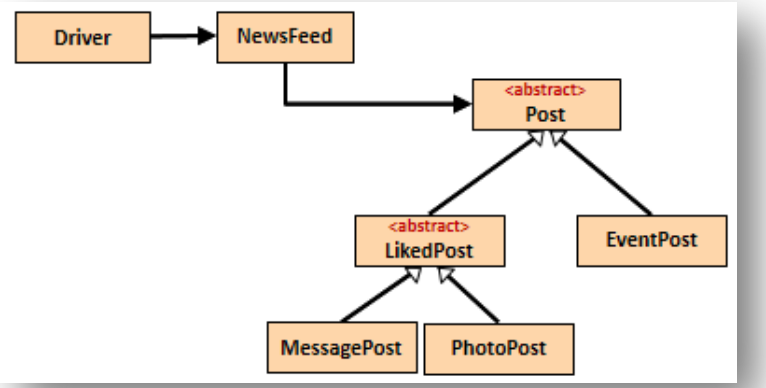
MessagePost.java

```
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;

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      }
```

# DEADLY DIAMOND OF DEATH

# Deadly Diamond of Death!

- Recall that multiple inheritance is not allowed in Java.

- Any idea why the Java designers decided to not allow multiple inheritance?

# Deadly Diamond of Death!

- Recall that multiple inheritance is not allowed in Java.
- Any idea why the Java designers decided to not allow multiple inheritance?
- It is because of the Deadly Diamond of Death problem!

# Deadly Diamond of Death!

- This is easiest explained by example.

- Let’s pretend that Java allows multiple inheritance and we will see really quickly what the Deadly Diamond of Death is!

http://javacodeonline.blogspot.ie/2009/08/deadly-diamond-of-death.html

# Deadly Diamond of Death - Example

- Suppose that we have an abstract super class, with an abstract method in it.

```
public abstract class AbstractSuperClass{
        abstract void do();
}
```

http://javacodeonline.blogspot.ie/2009/08/deadly-diamond-of-death.html

# Deadly Diamond of Death - Example

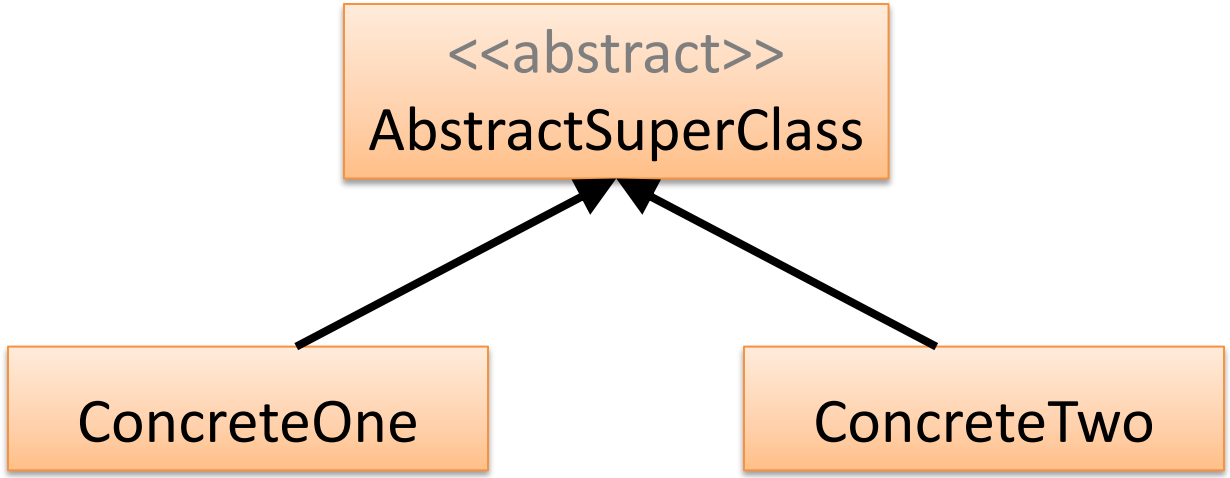
- Now two concrete classes extend this abstract super class.
- Each classes provides their own implementation of the abstract method defined in the super class.

```
public class ConcreteOne extends AbstractSuperClass{
    void do(){
        System.out.println("I am testing multiple Inheritance");
    }
}
```

```
public class ConcreteTwo extends AbstractSuperClass{
     void do(){
         System.out.println("I will cause the Deadly Diamond of Death");
     }
}
```

# Deadly Diamond of Death - Example

- So far, our class diagram looks like this:




http://javacodeonline.blogspot.ie/2009/08/deadly-diamond-of-death.html

# Deadly Diamond of Death - Example

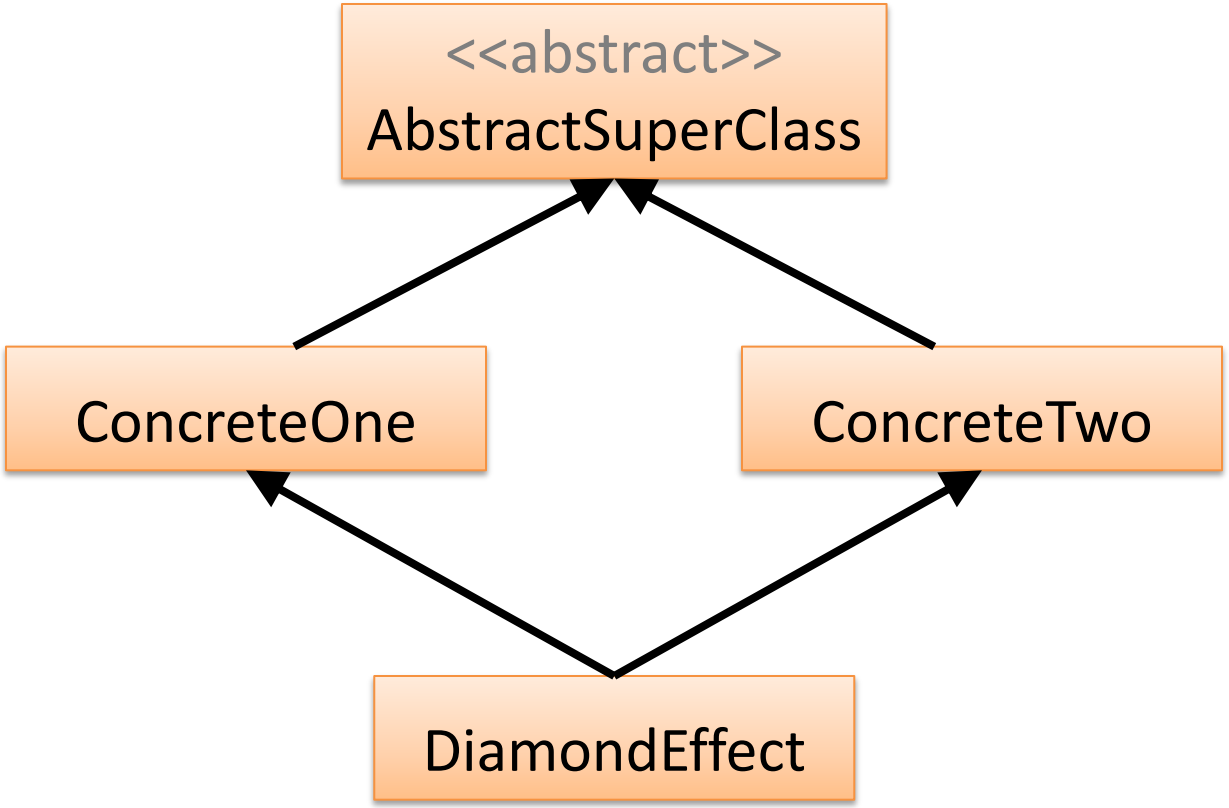
- Now a fourth class comes into picture which **extends** the above two concrete classes.

```
public class DiamondEffect extends ConcreteOne, ConcreteTwo{
    //Some methods of this class
}
```

http://javacodeonline.blogspot.ie/2009/08/deadly-diamond-of-death.html

# Deadly Diamond of Death - Example

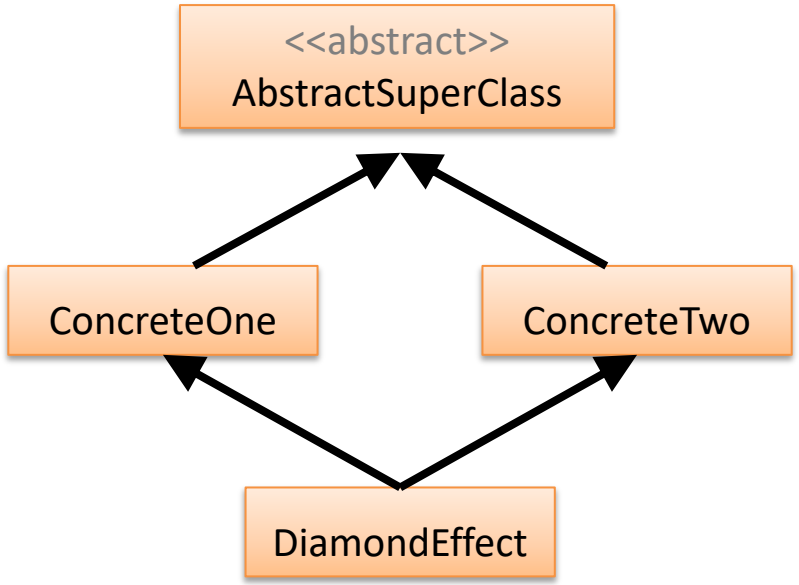
- Note that our class diagram is a diamond shape.




http://javacodeonline.blogspot.ie/2009/08/deadly-diamond-of-death.html

# Deadly Diamond of Death - Example

- The DiamondEffect class inherits all the methods of the parent classes.
- BUT we have a common method (void do()) in the two concrete classes, each with a different implementation.
- So which **void do()** implementation will be used for the DiamondEffect class as it inherits both these classes?




http://javacodeonline.blogspot.ie/2009/08/deadly-diamond-of-death.html

# Deadly Diamond of Death - Example

- Actually no one has got the answer to the above question, and so to avoid this sort of critical issue, Java banned multiple inheritance.

- The class diagram which is formed above is like that of a diamond, but with no solution or outcome, and so it is called Deadly Diamond of Death.

http://javacodeonline.blogspot.ie/2009/08/deadly-diamond-of-death.html

# Any Questions?