Inheritance

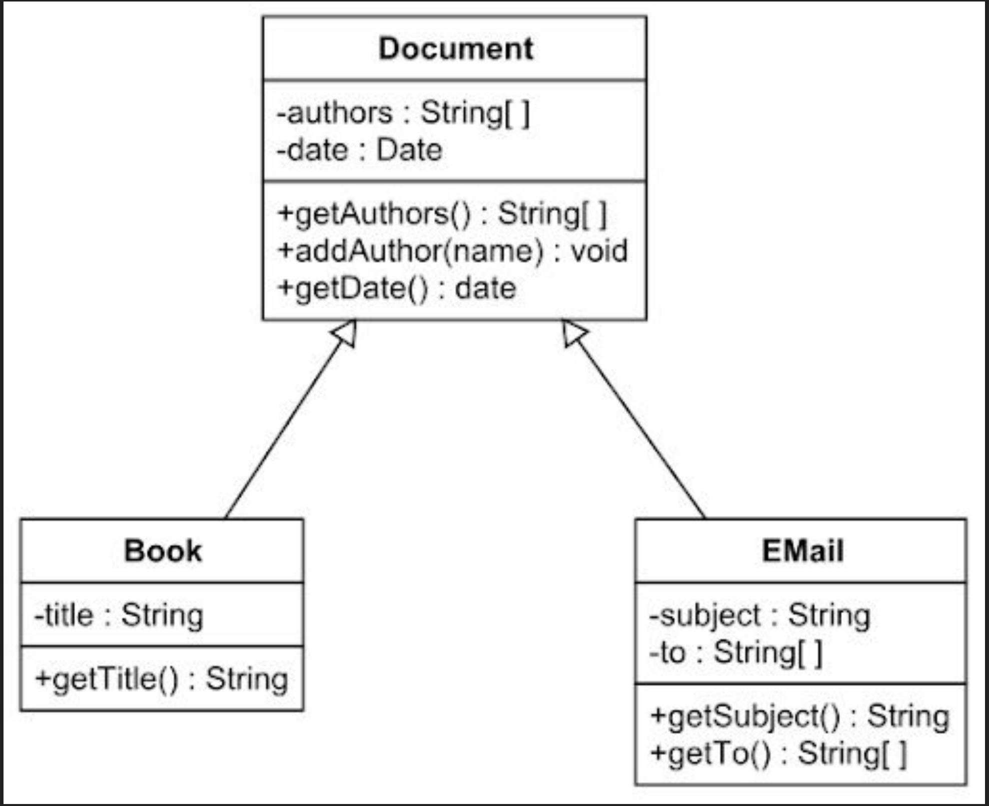
Why do we do it?

- Allows us to reuse code for common class properties and methods.
- Forces us to think about how our entities are orgnized

Reducing Repeated Code

Subclasses inherit all the properties and methods of their superclass.

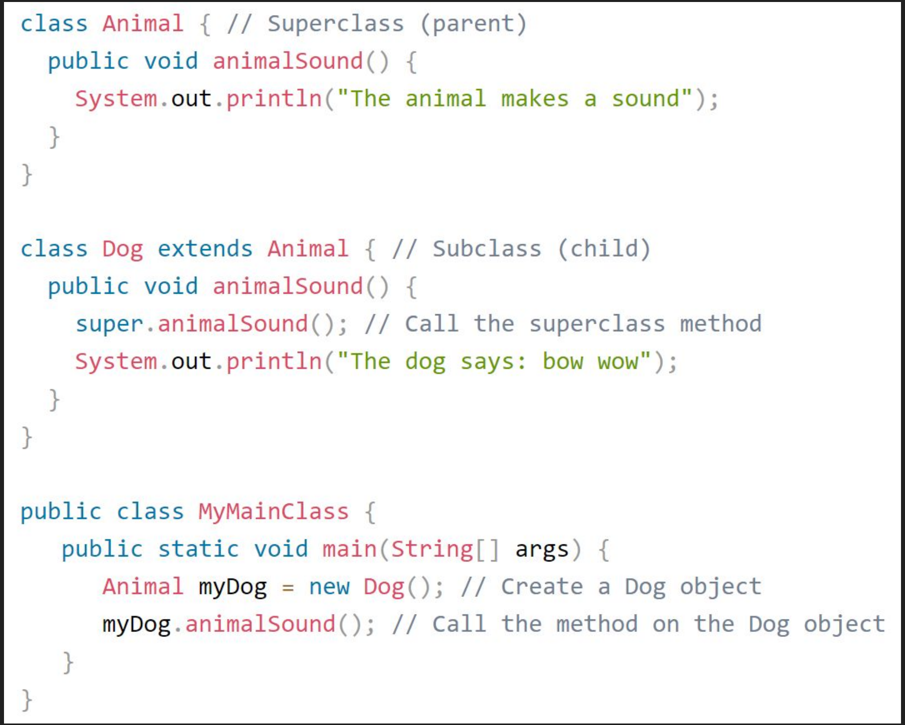
Subclasses get created through the use of the keywords “extends” or “implements” depending on the type of super class.



Super

Used to call superclass methods and the superclass constructor.

Mostly used for using the superclass constructor.



```java
class Animal { // Superclass (parent)
  public void animalSound() {
    System.out.println("The animal makes a sound");
  }
}

class Dog extends Animal { // Subclass (child)
  public void animalSound() {
    super.animalSound(); // Call the superclass method
    System.out.println("The dog says: bow wow");
  }
}

public class MyMainClass {
   public static void main(String[] args) {
      Animal myDog = new Dog(); // Create a Dog object
      myDog.animalSound(); // Call the method on the Dog object
   }
}
```

Output:

```
The animal makes a sound
The dog says: bow wow
```

Instanceof

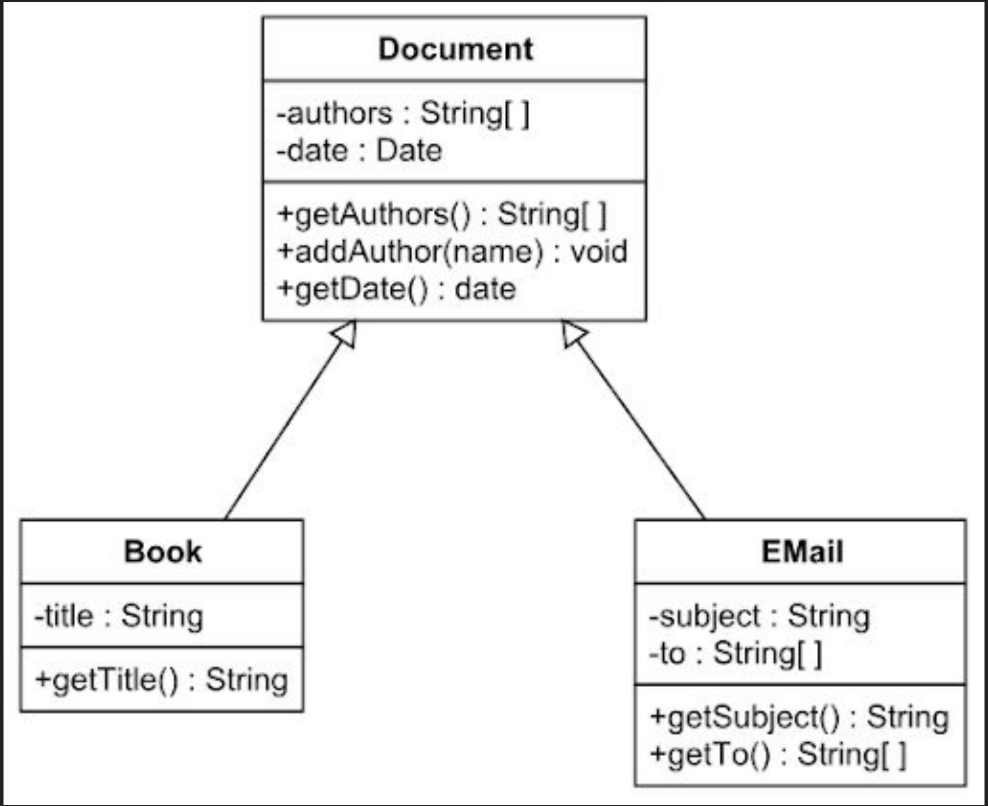
Used to test whether an object is an instance of the specified type.

- class
- subclass
- Interface

book intstanceof Document ⇒ true

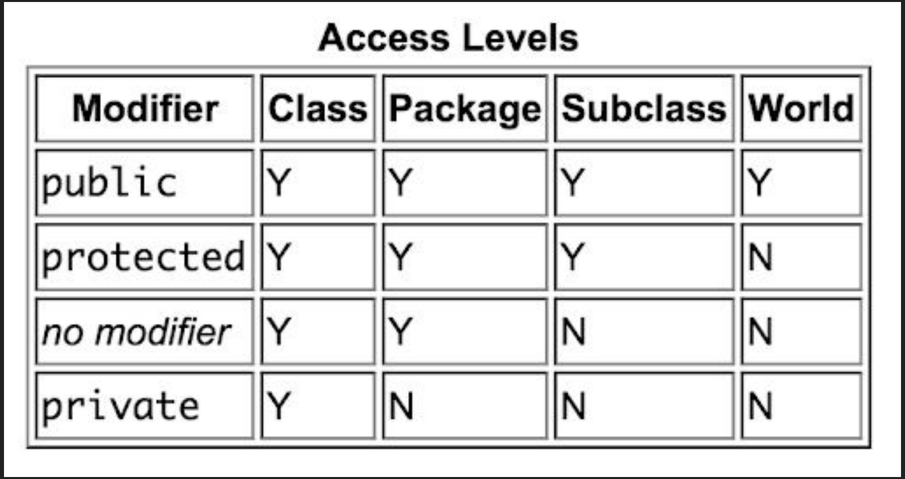
book instanceof Book ⇒ true

email instanceof Book ⇒ false



Public vs Private vs Protected

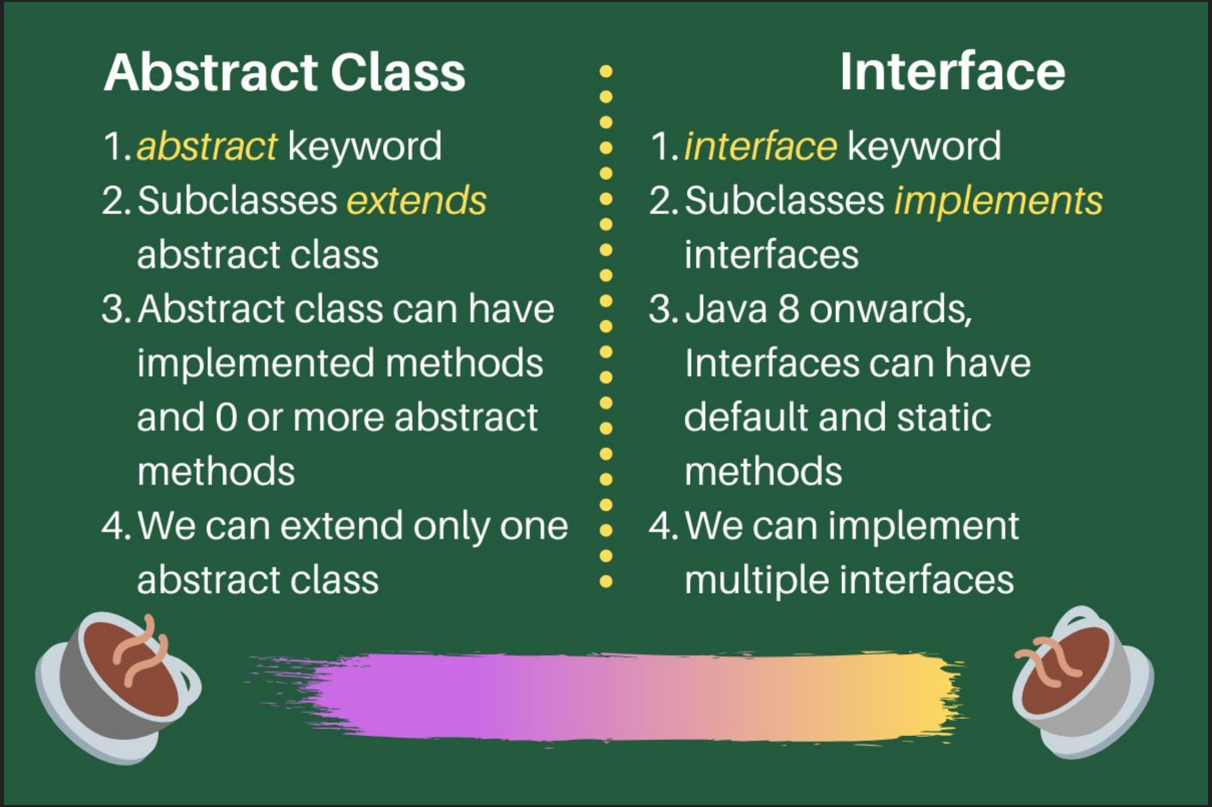
Access modifiers restrict who has access to specific class properties.



Access Levels

Modifier	Class	Package	Subclass	World
public	Y	Y	Y	Y
protected	Y	Y	Y	N
no modifier	Y	Y	N	N
private	Y	N	N	N

Differences between abstract classes and interfaces

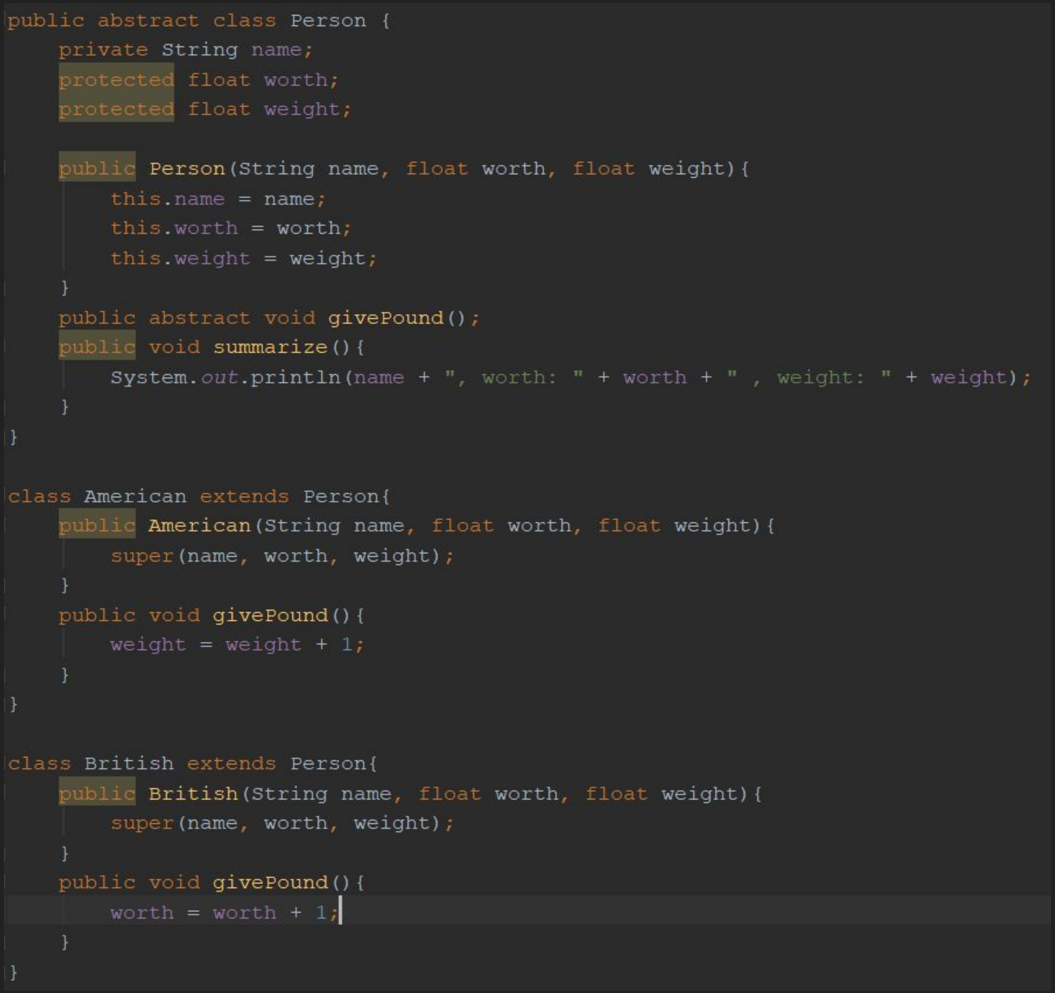


Abstract Class	Interface
1. *abstract* keyword	1. *interface* keyword
2. Subclasses *extends* abstract class	2. Subclasses *implements* interfaces
3. Abstract class can have implemented methods and 0 or more abstract methods	3. Java 8 onwards, Interfaces can have default and static methods
4. We can extend only one abstract class	4. We can implement multiple interfaces

Live Coding Example #1

Create an class *Person* that is the superclass of *American* and *British*. Every *Person* should have the properties: *name*, *weight*, and *worth*. Every *Person* should also have the methods: *summarize* and *givePound*. Giving a pound means for Americans means that you increase their *weight* by one and means for British that you increase their *worth* by one.

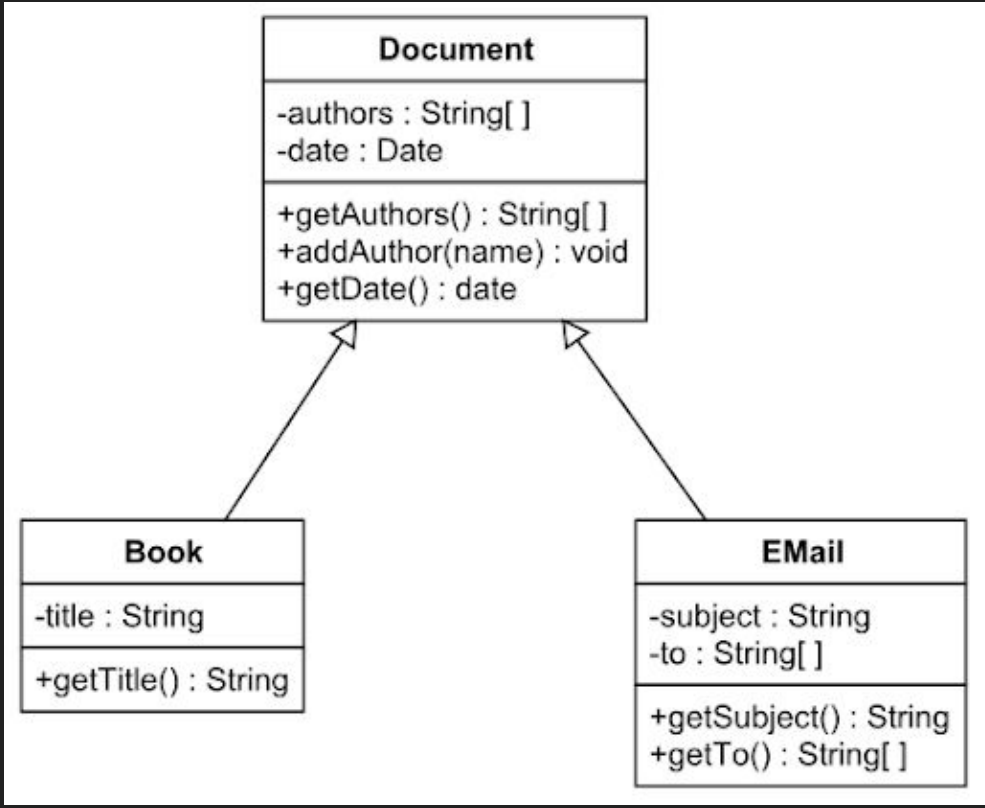
Example Solution



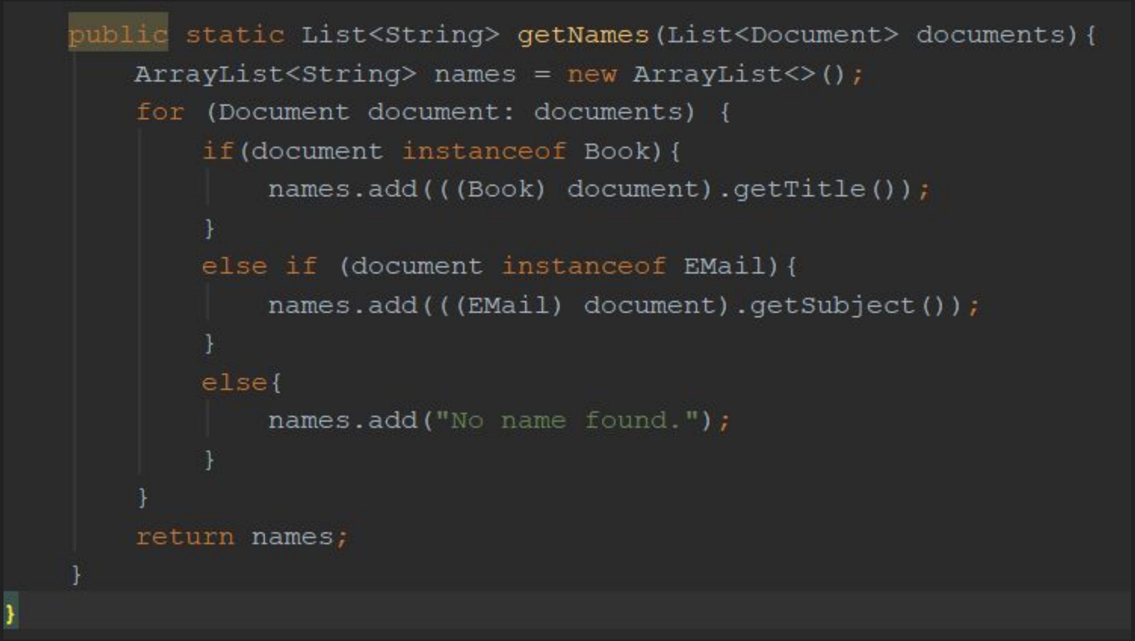
```java
public abstract class Person {
    private String name;
    protected float worth;
    protected float weight;

    public Person(String name, float worth, float weight){
        this.name = name;
        this.worth = worth;
        this.weight = weight;
    }
    public abstract void givePound();
    public void summarize(){
        System.out.println(name + ", worth: " + worth + " , weight: " + weight);
    }
}

class American extends Person{
    public American(String name, float worth, float weight){
        super(name, worth, weight);
    }
    public void givePound(){
        weight = weight + 1;
    }
}

class British extends Person{
    public British(String name, float worth, float weight){
        super(name, worth, weight);
    }
    public void givePound(){
        worth = worth + 1;
    }
}
```

Live Coding Example #2

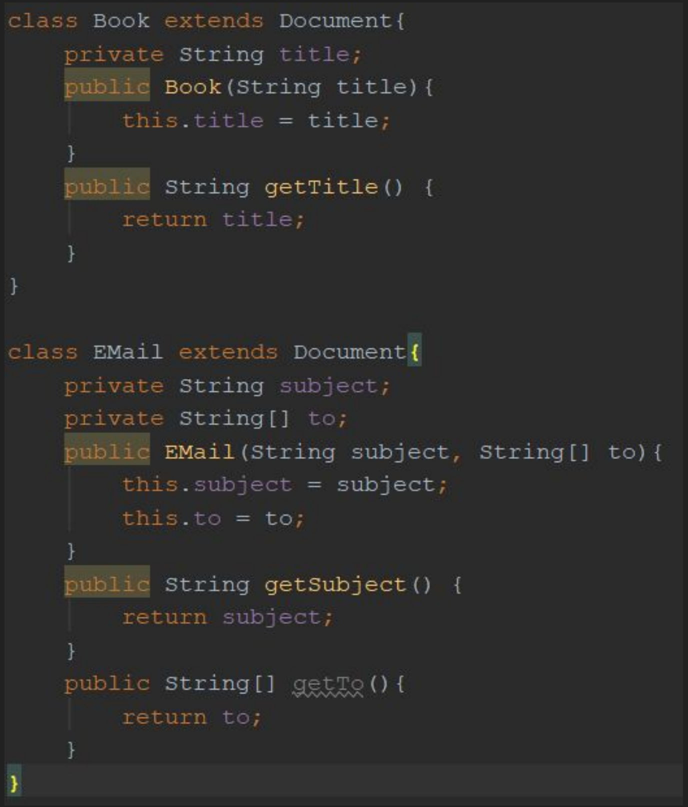
Create a method called *getNames* that takes in a list of *Documents* and returns a list of *Strings* representing the names of titles. For *Books*, use the *title*. For *EMails*, use the *subject*. For *Documents*, use “No name found.”



Example Solution



```java
    public static List<String> getNames(List<Document> documents){
        ArrayList<String> names = new ArrayList<>();
        for (Document document: documents) {
            if(document instanceof Book){
                names.add(((Book) document).getTitle());
            }
            else if (document instanceof EMail){
                names.add(((EMail) document).getSubject());
            }
            else{
                names.add("No name found.");
            }
        }
        return names;
    }
}
```



```java
class Book extends Document{
    private String title;
    public Book(String title){
        this.title = title;
    }
    public String getTitle() {
        return title;
    }
}

class EMail extends Document{
    private String subject;
    private String[] to;
    public EMail(String subject, String[] to){
        this.subject = subject;
        this.to = to;
    }
    public String getSubject() {
        return subject;
    }
    public String[] getTo(){
        return to;
    }
}
```

Assignment

Zoo Animals