

```
public class E extends F {
    public void method2() {
        System.out.print("E 2 ");
        method1();
    }
}

public class F extends G {
    public String toString() {
        return "F";
    }

    public void method2() {
        System.out.print("F 2 ");
        super.method2();
    }
}
```

```
public class G {
    public String toString() {
        return "G";
    }

    public void method1() {
        System.out.print("G 1 ");
    }

    public void method2() {
        System.out.print("G 2 ");
    }
}

public class H extends E {
    public void method1() {
        System.out.print("H 1 ");
    }
}
```

```
public class EFGHclient {  
    public static void main(String[] args) {  
        G[] elements = {new F(), new E(), new G(), new H()};  
        for ( int i = 0 ; i < elements.length ; i++ ) {  
            System.out.println(elements[i]);  
            elements[i].method1();  
            System.out.println();  
            elements[i].method2();  
            System.out.println();  
        }  
    }  
}
```

## Step 1 – Write/draw the class hierarchy

G  
↓  
F  
↓  
E  
↓  
H

Class headers in the order shown:


```
public class E extends F {  
public class F extends G {  
public class G {  
public class H extends E {
```

## Step 2 – Write a method output table

G  
↓  
F  
↓  
E  
↓  
H

List methods in the order called

- List classes in order of inheritance starting with base class
- Draw arrows showing which class is the super/parent class



	G	F	E	H
toString()				
method1()				
method2()				


**In each cell we will put the expected output:**

- If it is just a string, write it in double quotes to show it is a string
- If the class does not have this method, copy what its super class does
- If it has a method call:
  - If it is super. then it is specific and copy what the super class method does
  - If it is not super. then write the method call itself (not what it does) and circle it

### Step 3 – Start with the base class and fill out the table for it.

```
public class G {  
    public String toString() {  
        return "G";  
    }  
  
    public void method1() {  
        System.out.print("G 1 ");  
    }  
  
    public void method2() {  
        System.out.print("G 2 ");  
    }  
}
```

This is the base class, it has all the methods, just write each in.




	G	F	E	H
toString()	"G"			
method1()	"G 1 "			
method2()	"G 2 "			

#### In each cell we will put the expected output:

- If it is just a string, write it in double quotes to show it is a string
- If the class does not have this method, copy what its super class does
- If it has a method call:
  - If it is super. then it is specific and copy what the super class method does
  - If it is not super. then write the method call itself (not what it does) and circle it

## Step 3 – Continue with each of the child classes in order...

```
public class F extends G {  
    public String toString() {  
        return "F";  
    }  
  
    public void method2() {  
        System.out.print("F 2 ");  
        super.method2();  
    }  
}
```



	G	F	E	H
toString()	"G"	"F"		
method1()	"G 1 "	"G 1 "		
method2()	"G 2 "	"F 2 G 2"		


- 1) Has toString(), write it in
- 2) Does not have method1(), copy from parent
- 3) Has method2(), with a print and a super. method call, write in the print output and add/copy in G's method2()

### In each cell we will put the expected output:

- If it is just a string, write it in double quotes to show it is a string
- If the class does not have this method, copy what its super class does
- If it has a method call:
  - If it is super. then it is specific and copy what the super class method does
  - If it is not super. then write the method call itself (not what it does) and circle it

## Step 3 – Continue with each of the child classes in order...

```
public class E extends F {  
    public void method2() {  
        System.out.print("E 2 ");  
        method1();  
    }  
}
```



	G	F	E	H
toString()	"G"	"F"	"F"	
method1()	"G 1 "	"G 1 "	"G 1 "	
method2()	"G 2 "	"F 2 G 2"	"E 2 " + method1()	


- 1) Does not have toString(), copy from parent
- 2) Does not have method1(), copy from parent
- 3) Has method2(), with a print and a NON super. method call, write in the print output and write in and circle "method1()"

### In each cell we will put the expected output:

- If it is just a string, write it in double quotes to show it is a string
- If the class does not have this method, copy what its super class does
- If it has a method call:
  - If it is super. then it is specific and copy what the super class method does
  - If it is not super. then write the method call itself (not what it does) and circle it

## Step 3 – Continue with each of the child classes in order...

```
public class H extends E {  
    public void method1() {  
        System.out.print("H 1 ");  
    }  
}
```



	G	F	E	H
toString()	"G"	"F"	"F"	"F"
method1()	"G 1 "	"G 1 "	"G 1 "	"H 1 "
method2()	"G 2 "	"F 2 G 2"	"E 2 " + method1()	"E 2 " + method1()

- 1) Does not have toString(), copy from parent
- 2) Has method1(), but no method calls, just write in the output of the print
- 3) Has no method2(), copy from parent ... it is important that we do not (at this point) fill in what method1() does...

### In each cell we will put the expected output:


- If it is just a string, write it in double quotes to show it is a string
- If the class does not have this method, copy what its super class does
- If it has a method call:
  - If it is super. then it is specific and copy what the super class method does
  - If it is not super. then write the method call itself (not what it does) and circle it



## Step 4 – Answer the question using the table

```
public class EFGHclient {
    public static void main(String[] args) {
        G[] elements = {new F(), new E(), new G(), new H()};
        for ( int i = 0 ; i < elements.length ; i++ ) {
            System.out.println(elements[i]);
            elements[i].method1();
            System.out.println();
            elements[i].method2();
            System.out.println();
        }
    }
}
```

F  
G 1  
F 2 G 2  
F  
G 1  
E 2 G 1  
G  
G 1  
G 2  
F  
H 1  
E 2 H 1



	G	F	E	H
toString()	"G"	"F"	"F"	"F"
method1()	"G 1 "	"G 1 "	"G 1 "	"H 1 "
method2()	"G 2 "	"F 2 G 2"	"E 2 " + method1()	"E 2 " + method1()
	3rd	1st	2nd	4th

### Process:

1. Go in the order listed in the array
2. If it has a method call [E.method2() and H.method2()], pick the method for the current class!!!!
3. Write in what the method does.