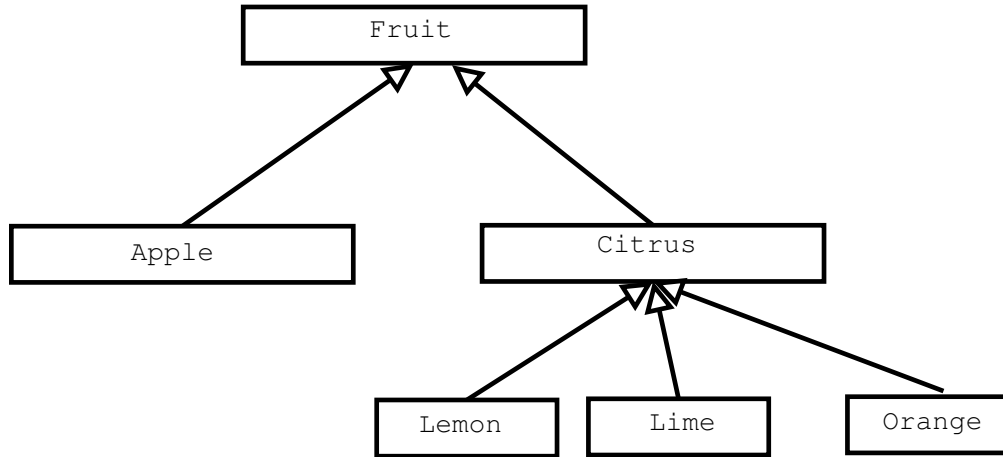


## DNHI Homework 1 Advanced Java Topics

### Problem 1

Suppose that `Fruit`, `Apple`, `Citrus`, `Lemon`, `Lime`, `Orange` are classes defined in the following inheritance hierarchy.



1. Can you create the following objects in a way specified? For each of them state "yes" if you can, or explain why not. Assume that each class provides default constructor.

- (a) `Fruit f = new Citrus();`
- (b) `Fruit f = new Lime();`
- (c) `Citrus c = new Fruit();`
- (d) `Citrus c = new Orange();`
- (e) `Apple a = new Citrus();`
- (f) `Citrus c = new Citrus();`

2. Each of the default constructors contains a print statement that states which classes constructor is called. `Fruit` class constructor prints "Fruit constructor called"; `Apple` class constructor prints "Apple constructor called"; and so on. Show the output when the following objects are created:

- (a) `Fruit f = new Lemon();`
- (b) `Apple a = new Apple();`

### Problem 2

Write a method that given a sorted `ArrayList` object of Java strings (objects of class `String`) removes all duplicates. Your method should modify the `ArrayList` object passed to it. The method should return a boolean value indicating if the list was modified or not (`true` for "has been modified", `false` for "has not been modified"). For example, if the original list passed to your method contains the following strings:

**Argentina, Chile, Chile, Czech Republic, France, Georgia, India, India, Poland, Romania, Romania**

your method should remove one occurrence of Chile, India and Romania. The resulting list should contain:

**Argentina, Chile, Czech Republic, France, Georgia, India, Poland, Romania**



### Problem 3

Consider the following class definition

```
1 public class Foo implements Comparable<Foo>{
2
3     double x;
4     double y;
5
6     public Foo ( double x, double y ) {
7         this.x = x;
8         this.y = y;
9     }
10
11    public int compareTo ( Foo other ) {
12        double d1 = x*x + y*y;
13        double d2 = other.x * other.x + other.y * other.y;
14        if ( d1 < d2 ) return -1 ;
15        if ( d1 == d2 ) return 0;
16        return 1;
17    }
18
19    public String toString ( ) {
20        return "( " + x + ", " + y + " )"; //returns ( x, y )
21    }
22 }
```

Given the array `fooList` of `Foo` objects pictured below (the values of `x` and `y` data fields are stated for each array element), show what the array will look like after the call to `Arrays.sort(fooList)`.

0	1	2	3	4	5	6	7	8	9
$x = 1.0$	$x = -2.0$	$x = 1.0$	$x = 1.0$	$x = 2.5$	$x = -1.0$	$x = 0.0$	$x = -1.0$	$x = 0.0$	$x = 0.0$
$y = 1.0$	$y = 2.0$	$y = 2.0$	$y = -1.0$	$y = 0.0$	$y = 0.0$	$y = 3.0$	$y = -4.0$	$y = 0.0$	$y = 1.5$

### Problem 4

**Part A** Given the definition of the `Foo` class in **Problem 3**, write the lines of code that are needed to create an `ArrayList` object and fill it with ten (10) `Foo` objects initialized with random values of `x` and `y` (Hint: this should be done with a loop). Do not write the entire program, just the lines that create and populate the `ArrayList` object.

**Part B** Give the `ArrayList` object that you created in Part A, write a single statement that will sort that array.

### Problem 5

A subclass inherits \_\_\_\_\_ from its superclass.

- private methods
- protected methods
- public methods
- constructors
- static methods



## Extra Challenge

What does the following Java code print:

```
1
2 public class PolymorphismQ3 {
3
4     public static void f(A x) {
5         A y = x;
6         y.key = x.key + 1;
7     }
8
9     public static void f(B x) {
10        B y = new B();
11        y.key = x.key + 2;
12        x = y;
13    }
14
15    public static void main(String[] args) {
16        A p = new A( );
17        p.key = 3;
18        B q = new B( );
19        q.key = 10;
20        f(p);
21        System.out.println(p.key);
22        f(q);
23        System.out.println(q.key);
24        p = q;
25        f(p);
26        System.out.println(p.key);
27    }
28 }
29
30
31 class A {
32     public int key;
33 }
34
35 class B extends A {
36 }
```