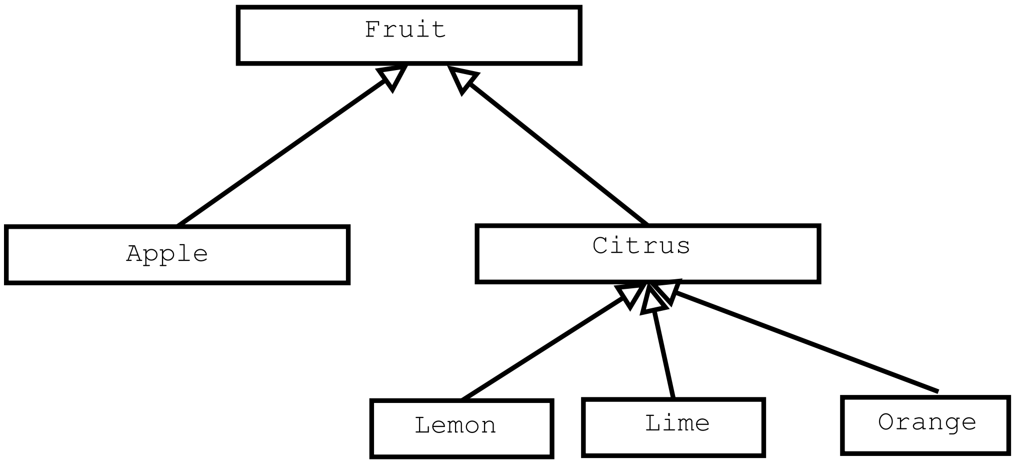
Joanna Klukowska
joannakl@cs.nyu.edu

ProblemSet 1 Solutions
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();`

   Answers: a) yes, b) yes, c) no, because a reference of a subclass cannot point to an object of a superclass (not every fruit is a citrus), d) yes, e) no, because Citrus is not a subclass of Apple, f) yes

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();`

   Answers:
   a)
   Fruit class constructor called
   Citrus class constructor called
   Lemon class constructor called
   b)
   Fruit class constructor called
   Apple class constructor called.

Joanna Klukowska
joannakl@cs.nyu.edu

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
```

Answer: This is my implementation. There might be others, equally correct, implementations.

```java
boolean findDuplicates( ArrayList<String> names ) {
  boolean modified = false;
  int i = 1;
  while (i < names.size()) {
    if (names.get(i).equals(names.get(i - 1))) {
      names.remove(i);
      modified = true;
    }
    else
      i++;
  }
  return modified;
}
```

Problem 3

Consider the following class definition

```java
public class Foo implements Comparable<Foo>{

    double x;
    double y;

    public Foo ( double x, double y ) {
        this.x = x;
        this.y = y;
    }

    public int compareTo ( Foo other ) {
        double d1 = x*x + y*y;
        double d2 = other.x * other.x + other.y * other.y;
        if ( d1 < d2 ) return -1 ;
        if ( d1 == d2 ) return 0;
        return 1;
    }

    public String toString ( ) {
        return "( " + x + ", " + y + " )"; //returns ( x, y )
    }
}
```

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
$x = 1.0$ $y = 1.0$	$x = -2.0$ $y = 2.0$	$x = 1.0$ $y = 2.0$	$x = 1.0$ $y = -1.0$	$x = 2.5$ $y = 0.0$	$x = -1.0$ $y = 0.0$	$x = 0.0$ $y = 3.0$	$x = -1.0$ $y = -4.0$	$x = 0.0$ $y = 0.0$	$x = 0.0$ $y = 1.5$

Answer: The compareTo() method of the Foo class is used by the sort method of the Collections class to determine the relative ordering of any two objects. Since the compareTo method uses the distances from origin to compare items, the resulting ordering is from smallest to largest distance of any Foo object from the origin.

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

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.

Answer: Acceptable answers should have code that looks like the following code fragments or something that is equivalent.

```
ArrayList <Foo> fooList = new ArrayList <Foo> ();
Foo tmp = null;

for (int i = 0; i < 10; i++ ) {
  tmp = new Foo(Math.random(), Math.random() );
  fooList.add(tmp);
}
```

```
ArrayList <Foo> fooList = new ArrayList <Foo> ();
Foo tmp = null;
Random rand = new Random();

for (int i = 0; i < 10; i++ ) {
  tmp = new Foo(rand.nextDouble(), rand.nextDouble() );
  fooList.add(tmp);
}
```

```
ArrayList <Foo> fooList = new ArrayList <Foo> ();

for (int i = 0; i < 10; i++ ) {
  fooList.add( new Foo(Math.random(), Math.random() ) );
}
```

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

Answer: `Collections.sort(fooList);`

Problem 5

A subclass inherits ________________ from its superclass.

- private methods
- protected methods $\Leftarrow$
- public methods $\Leftarrow$
- constructors

Joanna Klukowska
joannakl@cs.nyu.edu

Extra Challenge

What does the following Java code print:

```java

public class PolymorphismQ3 {

  public static void f(A x) {
    A y = x;
    y.key = x.key + 1;
  }

  public static void f(B x) {
    B y = new B();
    y.key = x.key + 2;
    x = y;
  }

  public static void main(String[] args) {
    A p = new A( );
    p.key = 3;
    B q = new B( );
    q.key = 10;
    f(p);
    System.out.println(p.key);
    f(q);
    System.out.println(q.key);
    p = q;
    f(p);
    System.out.println(p.key);
  }
}


class A {
  public int key;
}

class B extends A {
}
```

Answer:
4
10
11

HINT: step through this program using a debugger to make sure you understand how it works.