## Problem 1

Create three classes:

- **CalculatingDevice** with a sole constructor taking a name (**String**). The class defines also a method
  `public String calculate(double x, double y)`
  which calculates the sum of its arguments and returns *a string* (not a number) with the name of the machine and the result of the addition.
  Add a static method

  ```
  public static void printRes(CalculatingDevice[] a,
                              double x, double y)
  ```

  taking an array of calculataing machines and arguments x and y — the functions prints the results of calculations with these arguments for each machine from the array.
- **Calculator** extending **CalculatingDevice** and overriding the **calculate** method. The method gets the result of addition by invoking `super.calculate` and adds the result of subtraction (as a string).
- **Computer** extending **Calculator** and overriding the **calculate**. The method gets the result of addition and subtraction by invoking `super.calculate` and adds, as a string, the results of multiplication and division.

The following program

download *Computers.java*

```
public class Computers {
    public static void main (String[] args) {
        CalculatingDevice[] arr = {
                new Computer("Cray"),
                new CalculatingDevice("Abacus"),
                new Calculator("HP")
        };
        CalculatingDevice.printRes(arr, 21, 7);
    }
}
```

should print something like

```
Cray: 21.0+7.0=28.0; 21.0-7.0=14.0; 21.0*7.0=147.0; 21.0/7.0=3.0
Abacus: 21.0+7.0=28.0
HP: 21.0+7.0=28.0; 21.0-7.0=14.0
```

## Problem 2

Write a class **Vehicle** objects of which represent vehicles. The class should have the following members:

```java
public class Vehicle {
    private String name;
    private int price;
    private double mpg;
    public Vehicle(String n, int p, double f) { /* ... */ }
    @Override
    public String toString() { /* ... */ }
    public String getName() { /* ... */ }
    public int getPrice() { /* ... */ }
    public double getMpg() { /* ... */ }
    public int compareTo(Vehicle v) { /* ... */ }
    public static void sort(Vehicle[] vs) { /* ... */ }
}
```

Define also two derived classes:

```java
class Car extends Vehicle {
    public Car(String n, int p, double f) { /* ... */ }
    @Override
    public String toString() { /* ... */ }
}
```

and

```java
class Bicycle extends Vehicle {
    public Bicycle(String n, int p) { /* ... */ }
    @Override
    public String toString() { /* ... */ }
}
```

The field mpg corresponds to the fuel consumption in miles per gallon: the greater this number, the smaller is the fuel consumption. Bicycles do not consume fuel.

The getters throw exception

```java
throw new UnsupportedOperationException();
```

if the object on which they are invoked is of type **Vehicle**, and not of one of its derived types (this simulates, in a way, declaring the class as abstract).

The **compareTo** method returns a negative number if *this* object is considered "smaller", a positive number if the object passed as the argument is "smaller", and zero if we consider them equal. All bicycles are "smaller" than cars. For two bicycles, the less expensive is "smaller". For two cars, the one with smaller fuel consumption is "smaller".

The **toString** methods in the derived classes use the same method from the base class.

The static **sort** method sorts vehicles using **compareTo** method to comapare them.

**Reminder:**
Each class loaded by the JVM is represented by exactly one object of class Class. You can get the reference to this object either by name of the class (`NameOfClass.class`) or by an object (`object.getClass()`).

For example, the following program

download *Vehicles.java*

```
public static void main(String[] args) {
    Vehicle[] vs = {
        new Car("Toyota", 25000, 50.5),
        new Bicycle("Pinarello", 1800),
        new Car("Ford", 21000, 55.1),
        new Bicycle("Canyon", 1600)
    };
    Vehicle.sort(vs);
    System.out.println(Arrays.toString(vs)
              .replaceAll(", ", "\n "));
}
```

should print

```
[Bicycle Canyon($1600)
 Bicycle Pinarello($1800)
 Car Ford($21000; fc=55.1)
 Car Toyota($25000; fc=50.5)]
```