

Semplificazione dell'esame di Programmazione II del 6 luglio 2012

Si consideri il seguente programma:

```
public class Main {
    public static void main(String[] args) throws DuplicatedSampleException, InconsistentSampleSizeException {
        Sample s1 = new Sample("Cars", new float[] { 187.2f, 201.4f, 88.2f, 75.8f, 156.1f } );
        Sample s2 = new Sample("Bikes", new float[] { 91.3f, 98.7f, 111.3f, 120.4f, 100.2f } );
        Sample s3 = new Sample("Motorbikes", new float[] { 122.3f, 118.7f, 144.0f, 125.4f, 88.9f } );
        Sample s4 = new Sample("Skateboards", new float[] { 21.3f, 6.3f, 44.0f, 24.3f, 18.9f } );
        Plot p1 = new SequentialPlot();
        p1.add(s1); p1.add(s2); p1.add(s3); p1.add(s4);

        Sample s5 = new Sample("Cars", new float[] { 187.2f, 201.4f, 88.2f, 75.8f, 156.1f } );
        Sample s6 = new Sample("Bikes", new float[] { 91.3f, 98.7f, 111.3f, 120.4f, 100.2f } );
        Sample s7 = new Sample("Motorbikes", new float[] { 122.3f, 118.7f, 144.0f, 125.4f, 88.9f } );
        Plot p2 = new AlternatePlot();
        p2.add(s5); p2.add(s6); p2.add(s7); p2.add(s4);

        System.out.println(p1); System.out.println(p2); System.out.println(p1.equals(p2));
    }
}
```

che alla fine del compito dovrà stampare:

```
Cars:
| ***** (187.2)
| ***** (201.4)
| ***** (88.2)
| ***** (75.8)
| ***** (156.1)

Bikes:
| @@@@@@@@@@@@@@@@@@ (91.3)
| @@@@@@@@@@@@@@@@@@ (98.7)
| @@@@@@@@@@@@@@@@@@ (111.3)
| @@@@@@@@@@@@@@@@@@ (120.4)
| @@@@@@@@@@@@@@@@@@ (100.2)

Motorbikes:
| $$$$$$$$$$$$$$$$ (122.3)
| $$$$$$$$$$$$$$$$ (118.7)
| $$$$$$$$$$$$$$$$ (144.0)
| $$$$$$$$$$$$$$$$ (125.4)
| $$$$$$$$$$$$$$$$ (88.9)

Skateboards:
| **** (21.3)
| * (6.3)
| ***** (44.0)
| **** (24.3)
| *** (18.9)

Cars| ***** (187.2)
Bikes| @@@@@@@@@@@@@@@@@@ (91.3)
Motorbikes| $$$$$$$$$$$$$$$$ (122.3)
Skateboards| **** (21.3)

Cars| ***** (201.4)
Bikes| @@@@@@@@@@@@@@@@@@ (98.7)
Motorbikes| $$$$$$$$$$$$$$$$ (118.7)
Skateboards| * (6.3)

Cars| ***** (88.2)
Bikes| @@@@@@@@@@@@@@@@@@ (111.3)
Motorbikes| $$$$$$$$$$$$$$$$ (144.0)
Skateboards| ***** (44.0)

Cars| ***** (75.8)
Bikes| @@@@@@@@@@@@@@@@@@ (120.4)
Motorbikes| $$$$$$$$$$$$$$$$ (125.4)
Skateboards| **** (24.3)

true
```

Esercizio 1 [5 punti] Una *Sample* è una sequenza di valori con un nome associato. Si completi la sua classe:

```

public class Sample {
    private final String name;
    private final float[] values;

    public Sample(String name, float[] values) {
        this.name = name;
        this.values = values;
        ....
    }

    public String getName() { return name; }
    public int getSize() { return values.length; }
    public float getMax() { .... }
    public float getValue(int pos) { return values[pos]; }
    public boolean equals(Object other) ....
}

```

in modo che `getMax()` ritorni il massimo valore nella sample e che `equals` consideri uguali due `Sample` se e solo se hanno stesso nome e stesso numero di valori nello stesso ordine. Il costruttore deve lanciare una `java.lang.IllegalArgumentException` se uno o più dei valori è negativo.

Esercizio 2 [7 punti] Un `Plot` è una sequenza di un numero arbitrario di `Sample`, tutte con nomi diversi fra di loro. È possibile aggiungere una `Sample` a un `Plot` ed è possibile aggiungere a un `Plot` tutte le `Sample` di un altro `Plot`. Si completi la sua classe:

```

public abstract class Plot {
    private Sample[] samples;

    protected Plot() { .... }
    public final void add(Sample sample) throws DuplicatedSampleException, InconsistentSampleSizeException....
    public final void add(Plot plot) throws DuplicatedSampleException, InconsistentSampleSizeException....

    public final float getMax() { // il massimo valore fra tutte le sample
        float max = 0f;
        for (Sample sample: samples)
            max = Math.max(max, sample.getMax());
        return max;
    }

    public final boolean equals(Object other) ....
    protected final Sample[] getSamples() { return samples; }
    public abstract String toString();
}

```

I metodi `add` devono lanciare una `DuplicatedSampleException` se si prova ad aggiungere una `Sample` che ha lo stesso nome di una delle `Sample` già presente nel `Plot`; devono lanciare una `InconsistentSampleSizeException` se si prova ad aggiungere una `Sample` che ha una lunghezza (`getSize`) diversa dalla lunghezza delle altre `Sample` già presenti nel `Plot`. Le due classi di eccezione devono essere definite. Due `Plot` devono essere `equals` se e solo se contengono lo stesso numero di `Sample` e queste ultime sono `equals` a due e due e nello stesso ordine (si studi l'esempio della prima pagina).

Esercizio 3 [10 punti] Si scrivano le classi `SequentialPlot` e `AlternatePlot` che estendono `Plot` ridefinendo il metodo astratto `toString`. Tali ridefinizioni devono restituire delle stringhe come nell'esempio della pagina precedente: in `SequentialPlot` viene presentata una `Sample` alla volta; in `AlternatePlot` le `Sample` sono alternate. In entrambi i casi, i caratteri alternati sono `*Q$`. La lunghezza delle barre orizzontali deve essere proporzionale al valore che ciascuna barra rappresenta. Si riservino 40 caratteri per il valore più grande (metodo `getMax`).