Java exercises

January 2018

-

François de Coligny, Nicolas Beudez

0. Preliminary

Create a directory called **java/** on your machine to host all exercises.

1. Create a *Training* application

```java
package training;

/**
 * The Training application
 * @author F. de Coligny - January 2018
 */
public class Training {

    /**
     * Main method
     */
    public static void main(String[] args) {
        // print to screen
        System.out.println("The Java training ...");
    }

}
```

Check list

- in a directory called ***training/***
- class name: ***Training.java***
- **compile** in a terminal:

```
coligny@marvin-13:~/workspace/java$ javac training/Training.java
coligny@marvin-13:~/workspace/java$
```

- **execute** in a terminal:

```
coligny@marvin-13:~/workspace/java$ java training.Training
The Java training ...
coligny@marvin-13:~/workspace/java$
```

2. Create a *Tree* class

```java
package training;

/**
 * A simple tree
 * @author F. de Coligny - January 2018
 */
public class Tree {

    // diameter at breast height (cm)
    private double dbh;

    /**
     * Constructor
     */
    public Tree (double dbh) {
        this.dbh = dbh;
    }

    public double getDbh () {
        return dbh;
    }

    public String toString () {
        return "Tree dbh: " + dbh;
    }

}
```

Check list :

- in same directory than the *Training* class
- the **instance variable** is ***private***
- the **methods** are ***public***
- the **constructor** takes a *dbh*
- the ***toString ()*** method returns a *String* representation of the tree
- test it from the *Training* class

```
    Tree t1 = new Tree (5.6);
    System.out.println("t1: " + t1);
```

```
coligny@marvin-13:~/workspace/java$ java training.Training
The Java training ...
t1: Tree dbh: 5.6
```

3. Create a *SpatializedTree* class

```
package training;

/**
 * A tree with coordinates
 * @author F. de Coligny - January 2018
 */
public class SpatializedTree extends Tree {

    // x, y coordinates of the base of the trunk (m)
    private double x;
    private double y;

    /**
     * Constructor
     */
    public SpatializedTree (double dbh, double x, double y) {
        super (dbh);
        setXY (x, y);
    }

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

    public double getX () {return x;}
    public double getY () {return y;}

    public String toString () {
        return super.toString () + " x: "+ x + " y: "+ y;
    }
}
```

Check list :

- this class ***extends*** *Tree*
- its constructor **calls** the constructor of *Tree*
- the ***toString ()*** method relies on *Tree.toString ()*
- test it from the *Training* class

```
    Tree t2 = new SpatializedTree (5.6, 2, 1);
    System.out.println("t2: " + t2);
```

```
coligny@marvin-13:~/workspace/java$ javac training/*.java
coligny@marvin-13:~/workspace/java$ java training.Training
The Java training ...
t1: Tree dbh: 5.6
t2: Tree dbh: 5.6 x: 2.0 y: 1.0
coligny@marvin-13:~/workspace/java$
```

4. Add instance variables in the tree

```
private int id; // unique
private int age; // years
private double height; // m

public int getId() {
    return id;
}
...
```

Check list :

- must these properties be added in *Tree* or in *SpatializedTree* ?
- add the **accessors** for all variables : *getHeight ()*...
- adapt the constructors and *toString ()* methods
- do we adapt only one class ?
- test it from the *Training* class by adapting your code
- expected result is of this kind...

```
coligny@marvin-13:~/workspace/java$ java training.Training
The Java training ...
t1: Tree id: 1 age: 5 height: 3.5 dbh: 5.6
t2: Tree id: 2 age: 6 height: 6.7 dbh: 6.1 x: 4.0 y: 3.0
```

5. Add methods in the tree

```
public double getCrownRadius () { // m
    return dbh / 5d; // e.g. 5cm -> 1m
}

public double getCrownBaseHeight () { // m
    return height * 2d / 3d; // e.g. 6m -> 4m
}
```

Check list :

- must these methods be added in *Tree* or in *SpatializedTree* ?
- they do not rely on specific instance variables
- adapt what must be adapted to get this kind of result…

```
coligny@marvin-13:~/workspace/java$ javac training/*.java
coligny@marvin-13:~/workspace/java$ java training.Training
The Java training ...
t1: Tree id: 1 age: 5 height: 3.5 dbh: 5.6 cr: 1.1199999999999999 cbh: 2.3333333333333335
t2: Tree id: 2 age: 6 height: 6.7 dbh: 6.1 cr: 1.22 cbh: 4.466666666666667 x: 4.0 y: 3.0
```

- use a *NumberFormat* to get a better result...

```
import java.text.NumberFormat;
import java.util.Locale;
...
// code example, to be adapted
NumberFormat nf = NumberFormat.getInstance (Locale.ENGLISH); // decimal separator: '.'
return ... + " cr: " + nf.format(getCrownRadius()) + ...
```

```
The Java training ...
t1: Tree id: 1 age: 5 height: 3.5 dbh: 5.6 cr: 1.12 cbh: 2.333
t2: Tree id: 2 age: 6 height: 6.7 dbh: 6.1 cr: 1.22 cbh: 4.467 x: 4.0 y: 3.0
```

6. A method to create a list of trees

```
import java.util.Random;
import java.util.List;
import java.util.ArrayList;
...

private static List createTrees(int numberOfTrees, double xSize, double ySize) {

    List trees = new ArrayList(); // a list
    Random random = new Random(); // a random generator

    // create trees, add them in the list
    for (int i = 0; i < numberOfTrees; i++) {

        int id = i + 1;
        int age = random.nextInt(25) + 1; // [0,24[ + 1
        double height = age / 2d;
        double dbh = age;
        double x = random.nextDouble() * xSize; // [0,1[ * xSize
        double y = random.nextDouble() * ySize;

        Tree t = new SpatializedTree(id, age, height, dbh, x, y);
        trees.add(t);
    }

    // return the list
    return trees;
}
```

Check list :

- create a **list** and add trees in it
- in which class do you add this method ?
- use a **random generator** to draw random *int* and *double* numbers
- test your method from the *Training* class

```
List l = Training.createTrees(20, 50, 30);
for (Object o : l) {
    System.out.println(o.toString ());
}
```

- expected result is of this kind...

```
coligny@marvin-13:~/workspace/java$ java training.Training
The Java training ...
t1: Tree id: 1 age: 5 height: 3.5 dbh: 5.6 cr: 1.12 cbh: 2.33
t2: Tree id: 2 age: 6 height: 6.7 dbh: 6.1 cr: 1.22 cbh: 4.47 x: 4 y: 3
Tree id: 1 age: 2 height: 1.0 dbh: 2.0 cr: 0.4 cbh: 0.67 x: 18.55 y: 29.94
Tree id: 2 age: 16 height: 8.0 dbh: 16.0 cr: 3.2 cbh: 5.33 x: 8.24 y: 29.84
Tree id: 3 age: 20 height: 10.0 dbh: 20.0 cr: 4 cbh: 6.67 x: 1.73 y: 12.71
Tree id: 4 age: 15 height: 7.5 dbh: 15.0 cr: 3 cbh: 5 x: 37.97 y: 28.51
Tree id: 5 age: 8 height: 4.0 dbh: 8.0 cr: 1.6 cbh: 2.67 x: 19.26 y: 17.71
Tree id: 6 age: 5 height: 2.5 dbh: 5.0 cr: 1 cbh: 1.67 x: 35.52 y: 15.04
Tree id: 7 age: 19 height: 9.5 dbh: 19.0 cr: 3.8 cbh: 6.33 x: 31.22 y: 14.47
Tree id: 8 age: 5 height: 2.5 dbh: 5.0 cr: 1 cbh: 1.67 x: 23.35 y: 26.14
Tree id: 9 age: 25 height: 12.5 dbh: 25.0 cr: 5 cbh: 8.33 x: 49.58 y: 19.16
Tree id: 10 age: 10 height: 5.0 dbh: 10.0 cr: 2 cbh: 3.33 x: 23.73 y: 17.6
Tree id: 11 age: 14 height: 7.0 dbh: 14.0 cr: 2.8 cbh: 4.67 x: 8.89 y: 23.68
Tree id: 12 age: 3 height: 1.5 dbh: 3.0 cr: 0.6 cbh: 1 x: 29.4 y: 26.91
Tree id: 13 age: 25 height: 12.5 dbh: 25.0 cr: 5 cbh: 8.33 x: 15.61 y: 2.51
Tree id: 14 age: 23 height: 11.5 dbh: 23.0 cr: 4.6 cbh: 7.67 x: 37.34 y: 0.86
Tree id: 15 age: 17 height: 8.5 dbh: 17.0 cr: 3.4 cbh: 5.67 x: 47.63 y: 9.84
Tree id: 16 age: 16 height: 8.0 dbh: 16.0 cr: 3.2 cbh: 5.33 x: 35.93 y: 8.16
Tree id: 17 age: 24 height: 12.0 dbh: 24.0 cr: 4.8 cbh: 8 x: 29.88 y: 25.06
Tree id: 18 age: 5 height: 2.5 dbh: 5.0 cr: 1 cbh: 1.67 x: 34.32 y: 20.84
Tree id: 19 age: 5 height: 2.5 dbh: 5.0 cr: 1 cbh: 1.67 x: 15.45 y: 7.95
Tree id: 20 age: 15 height: 7.5 dbh: 15.0 cr: 3 cbh: 5 x: 26.58 y: 5.95
coligny@marvin-13:~/workspace/java$
```

7. Write the trees in a file

```
import java.io.*;
...

private static void write(List trees, String fileName) throws Exception {
    try {
        // open the file
        BufferedWriter out = new BufferedWriter(new FileWriter(fileName));

        char tab = '\t';
        double z = 0;

        String header = "#id" + tab + "age" + tab + "x" + tab + "y" + tab + "z"
                + tab + "height(m)" + tab + "dbh(cm)"
                + tab + "crownBaseHeight(m)" + tab + "crownRadius(m)";

        // write a line in the file
        out.write(header);
        out.newLine();

        for (Object o : trees) {
            SpatializedTree t = (SpatializedTree) o;

            String line = "" + t.getId() + tab + t.getAge() + tab + t.getX()
                    + tab + t.getY() + tab + z + tab + t.getHeight()
                    + tab + t.getDbh() + tab + t.getCrownBaseHeight()
                    + tab + t.getCrownRadius();

            out.write(line);
            out.newLine();
        }

        // close the file
        out.close();

    } catch (Exception e) { // in case of error
        System.out.println("Could not write in file: " + fileName);
        throw e; // send the exception to the caller
    }
}
```

Check list :

- if **error**, the method writes a message AND **throws an exception**
- the caller method : *main ()* now **throws an exception**
- after each line, a *newLine* must be written
- the columns are separated by tabs
- the **header line** starts with a « # »
- where is located the *trees.txt* file ?
- test this method after the tree list creation in the *Training* class

```
String fileName = "trees.txt";
write (l, fileName);
System.out.println ("Wrote the tree list in " + fileName);
```

```
Wrote the tree list in trees.txt
coligny@marvin-13:~/workspace/java$
```

- resulting file (extract) is...

```
#id age x    y    z    height(m)    dbh(cm) crownBaseHeight(m)  crownRadius(m)
1   6   35.74267332133339   8.949606218235797   0.0 3.0 6.0 2.0 1.2
2   23  28.13057680560921   29.14153605187614   0.0 11.5    23.0    7.666666666666667   4.6
3   17  4.288716952792826   29.188933560399043  0.0 8.5 17.0    5.666666666666667   3.4
4   14  1.384694917745799   6.1493276895649185  0.0 7.0 14.0    4.666666666666667   2.8
5   20  41.12412346883229   29.415986518822276  0.0 10.0    20.0    6.666666666666667   4.0
6   12  4.408600958868614   4.145146285159146   0.0 6.0 12.0    4.0 2.4
7   22  48.50601122266476   8.95947047956726    0.0 11.0    22.0    7.333333333333333   4.4
8   25  22.819975882509347  13.960418996533248  0.0 12.5    25.0    8.333333333333334   5.0
```

8. Pass parameters on the command line

```
/**
 * Main method expects 4 arguments: numberOfTrees, xSize, ySize, fileName
 */
public static void main(String[] args) throws Exception {

    for (int i = 0; i < args.length; i++) {
        System.out.println("args[" + i + "]: " + args[i]);
    }

    int numberOfTrees = 20;
    double xSize = 50;
    double ySize = 30;
    String fileName = "trees.txt";

    try {
        if (args.length != 4) throw new Exception();

        numberOfTrees = new Integer (args[0]).intValue ();
        xSize = new Double (args[1]).doubleValue ();
        ySize = new Double (args[2]).doubleValue ();
        fileName = args[3];

    } catch (Exception e) {
        System.out.println("Training expects 4 arguments: numberOfTrees, xSize, ySize,
fileName");
        return; // stops the program
    }

    // Prints to screen
    System.out.println("The Java training ...");

//  Tree t1 = new Tree(1, 5, 3.5, 5.6);
//  System.out.println("t1: " + t1);

//  Tree t2 = new SpatializedTree(2, 6, 6.7, 6.1, 4, 3);
//  System.out.println("t2: " + t2);

    List l = Training.createTrees(numberOfTrees, xSize, ySize);
    for (Object o : l) {
        System.out.println(o.toString());
    }

    write(l, fileName);
    System.out.println("Wrote the tree list in " + fileName);

}
```

Check list :

- **check** that all expected arguments are there
- transform the *Strings* into numbers
- in case of **trouble**, **explain** and **stop the program**
- change the literal values by the given variables
- expected result is of this kind...

```
coligny@marvin-13:~/workspace/java$ java training.Training 20
args[0]: 20
Training expects 4 arguments: numberOfTrees, xSize, ySize, fileName
coligny@marvin-13:~/workspace/java$ java training.Training 20 50 30 trees-2.txt
args[0]: 20
args[1]: 50
args[2]: 30
args[3]: trees-2.txt
The Java training ...
Tree id: 1 age: 7 height: 3.5 dbh: 7.0 cr: 1.4 cbh: 2.33 x: 46.83 y: 2.69
Tree id: 2 age: 4 height: 2.0 dbh: 4.0 cr: 0.8 cbh: 1.33 x: 36.25 y: 17.85
Tree id: 3 age: 4 height: 2.0 dbh: 4.0 cr: 0.8 cbh: 1.33 x: 33.96 y: 23.82
Tree id: 4 age: 18 height: 9.0 dbh: 18.0 cr: 3.6 cbh: 6 x: 37.78 y: 26.35
Tree id: 5 age: 8 height: 4.0 dbh: 8.0 cr: 1.6 cbh: 2.67 x: 25.22 y: 16.32
Tree id: 6 age: 19 height: 9.5 dbh: 19.0 cr: 3.8 cbh: 6.33 x: 8.84 y: 11.26
Tree id: 7 age: 1 height: 0.5 dbh: 1.0 cr: 0.2 cbh: 0.33 x: 33.08 y: 11.39
Tree id: 8 age: 1 height: 0.5 dbh: 1.0 cr: 0.2 cbh: 0.33 x: 47.85 y: 19.1
Tree id: 9 age: 6 height: 3.0 dbh: 6.0 cr: 1.2 cbh: 2 x: 30.14 y: 23.55
Tree id: 10 age: 19 height: 9.5 dbh: 19.0 cr: 3.8 cbh: 6.33 x: 23.24 y: 23.34
Tree id: 11 age: 18 height: 9.0 dbh: 18.0 cr: 3.6 cbh: 6 x: 37.71 y: 16.08
Tree id: 12 age: 18 height: 9.0 dbh: 18.0 cr: 3.6 cbh: 6 x: 15.26 y: 5.97
Tree id: 13 age: 12 height: 6.0 dbh: 12.0 cr: 2.4 cbh: 4 x: 23.1 y: 16.6
Tree id: 14 age: 20 height: 10.0 dbh: 20.0 cr: 4 cbh: 6.67 x: 0.21 y: 12.07
Tree id: 15 age: 2 height: 1.0 dbh: 2.0 cr: 0.4 cbh: 0.67 x: 5.06 y: 16.35
Tree id: 16 age: 9 height: 4.5 dbh: 9.0 cr: 1.8 cbh: 3 x: 34.19 y: 27.61
Tree id: 17 age: 12 height: 6.0 dbh: 12.0 cr: 2.4 cbh: 4 x: 13.9 y: 28.41
Tree id: 18 age: 16 height: 8.0 dbh: 16.0 cr: 3.2 cbh: 5.33 x: 42.18 y: 20.08
Tree id: 19 age: 8 height: 4.0 dbh: 8.0 cr: 1.6 cbh: 2.67 x: 31.6 y: 11.55
Tree id: 20 age: 15 height: 7.5 dbh: 15.0 cr: 3 cbh: 5 x: 46.3 y: 3.3
Wrote the tree list in trees-2.txt
```