# Java Workbook

## 08. Java IO

~by Andrzej "Andret2344" Chmiel

# Exercise 1: Traversing through the file tree

Create a `PathBuilder` class implementing the `Builder` interface so that it stores information about the current path.

```
class MainTraversing {
    private static final Scanner SCANNER = new Scanner(System.in);

    public static void main(final String[] args) {
        final PathBuilder builder = new PathBuilder();
        String line = null;
        System.out.print(builder.get() + "$> ");
        while (!(line = SCANNER.nextLine()).equals("")) {
            /* HERE */
            System.out.print(builder.get() + "$> ");
        }
        System.out.println("Terminating...");
    }
}

interface Builder {
    void add(String s);

    Path get();
}
```

In the pinpointed place create code that handles traversing through directories.

- Using a file's name prints out `ERROR: Not a directory!` and does nothing.
- Using an invalid name prints out `ERROR: Invalid name!` and does nothing.
- Directories `.` and `..` should work correctly (`/dir` instead of `/dir/../dir/../.dir`).

because of differences between operating systems, let the code works at least on one of them, but a universal solution is more than welcome. Use only classes from the `java.nio` package (the new API). An example of solution (mix of input and output):

```
/home$> ..
/$> etc
/etc$> apache
ERROR: Invalid name!
/etc$> apache2
/etc/apache2$> .
/etc/apache2$> apache2.conf
ERROR: Not a directory!
/etc/apache2$> ..
/etc$> ..
/$> ..
/$>
```

# Exercise 2: JSON (de-)serialization

Given the following code:

```
record Animal(String name, String genre) { }

record Person(String firstName, String lastName, List<Animal> animals) { }
```

Using the `org.json:json` dependency create methods from which one allows saving `List<Person>` to a `*.json` file and the other one allows reading the content and recreating the list.

# Exercise 3: GSON instead of JSON

For classes from the previous exercise, create another method that does the same, but instead of using `org.json:json` use `com.google.code.gson:gson`.

# Exercise 4: Remote data

Given the URL `https://public.andret.eu/questions.json` containing JSON data. Analyze its structure and create a class model that corresponds with it. Then create a `QuestionsService` class which contains a method that executes an HTTPS request to gather the data and fill in the structure and returns it as an object or a list (depending on the needs).

# Exercise 5: Low-level communication

Create a TCP/UDP communication working example. Two clients should be able to send and receive data from each other.

# Exercise 6: Watcher

Given the following code:

```
interface IWatcherService {
    void watch(Path path) throws IOException;

    void setOnFileChangedListener(Runnable fileChangedListener);

    void stop();
}

class MainWatchers {
    public static void main(final String[] args)
            throws IOException, InterruptedException {
        final Path path = Paths.get("pom.xml").toAbsolutePath();
        final IWatcherService watcherService = new WatcherService();
        watcherService.setOnFileChangedListener(() ->
                System.out.println("The file changed!"));
        watcherService.watch(path);
        Thread.sleep(30000);
        watcherService.stop();
    }
}
```

Create the implementation of the `IWatcherService` interface, so the `start` method runs a separate thread and the `stop` method stops it, so running the code above watches changes of the file and if the file changes, it prints out this:

```
The file changed!
The file changed!
The file changed!
The file changed!
```

# Exercise 7: XML's

Given the following entity declaration:

```
record Person(String firstName, String lastName) {
}
```

Using the built-in XML solution, serialize a `List<Person>` to an XML file.

# Exercise 8: Codesception

Create code that writes a `Hello world` program's code into a file and then compiles and runs it.

# Exercise 9: Rainbowify

In the main method, create printing statements that print out the following colorized text:

```
Hello
world!
```

Remember to reset the color after the printout.

# Exercise 10: Magnifer

Using Java's built-in solutions, fill in the body of the `magnify` so for the path pointing to a real and correct PNG file, scales its width and height times two and saves as a copy.

```
class MainMagnifier {
    public static void main(final String[] args) {
        magnify(Paths.get("image.png"));
    }

    static void magnify(final Path path) {

    }
}
```

In case of any problem, throw an exception with a brief description.