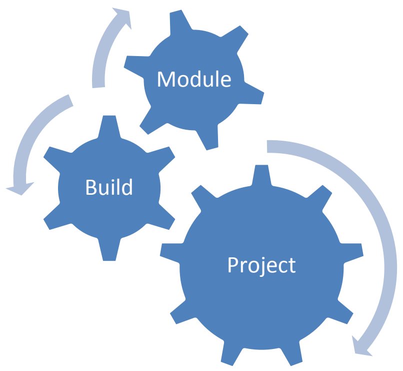

Please:

- Use Java 8
- Use Maven to support the build process, which can be compiled and tested by using `mvn verify`.
- Deliver an archived (ZIP) project, containing a folder with your project. Name the directory and ZIP after your first and last name.
- Do not deliver any compiled classes or project files of your IDE.

Object Orientation

Task 1. Objects

Define a class `Student` with the reference variables `id` and `name`, a constructor and default methods. It is given:

ID	Name
0054	Albert Einstein
1234	Gottfried Wilhelm Leibniz
5421	Carl Friedrich Gauss

Write code for a console output (Name and ID) that is sorted by the student names.

Task 2. Inheritance

Look at the following class:

```java
public class Product {

    String name;
    String description;
    double price;

    public Product (String name, String desc, double price) {
        this.name = name;
        this.description = desc;
        this.price = price;
    }

    public final double getPriceWithTax() {
        return price * 1.19;
    }

    public String toString() {
        return name + " _ " + description + " _ " + price + " EUR";
    }
}
```

Write a subclass `Clothing` that extends the class `Product`. The subclass must have additional attributes for the size (`int`) and the material (`String`). The new attributes shall be initialized in the constructor like the other attributes `name`, `description` and `price`, as well as implemented in the method `toString()` to override the output.

Java API

Task 3. Converting data

Write two methods `stringToBase64(String)` which encodes a string as base64 and `base64ToString(String)` which decodes a base64 string into a human readable string.

Don't use any external dependencies (like Apache Commons).

Write a unit test.

Task 4. XML Processing

Data needs to be exchanged between systems. The data format is XML, but you want to work with objects in your program. To achieve this serialize an object into a XML document and deserialize a XML document into an object. Please use a widely known Java technology.

Maybe you can use Maven to generate Java beans from a XML schema description.

Write a unit test.

Design Pattern

Task 5. Singleton

What is a singleton pattern and when do you use it?

Please code an example.

Task 6. Dependency Injection

There are two tightly coupled classes `ClassA` and `ClassB`. In a method of `ClassA` an instance of `ClassB` is created and a method `callMe()` is called on that instance.

Decouple this two classes by introducing an interface. The first class should not need to create the instance by its own, use a dependency injection framework.

Code Katas

Task 7. ROT-N

Write a method to encrypt a text with ROT-*N*. The algorithm translates a text character by character, lower case will be converted to upper case. N is the distance in the alphabet between the character to be translated and the result of the translation, for example with N=13 the character “a” is translated into character “n”. When the translation reaches the end of the alphabet continue at its beginning, e.g. “w” will be translated into “j”. N should be flexible and given via the constructor.

Example: `Hello, World -> URYYB, JBEYQ`

Write a unit test.

Task 8. Ordered Jobs

Executing dependent jobs in the right order is important for any business. Develop a class which plans the execution of jobs which may depend on each other.

Every job is represented by a single character. Maybe a job “a” must be completed before job “b”, then job “b” depends on “a”. Any job may depend on any number of other jobs. After registering jobs with their dependencies the class should compute the order of jobs for execution.

The interface is defined as:

```
interface OrderedJobs {
    void register(char job);
    void register(char job, char dependentJob);
    String[] sort();
}
```

For example, registering the following jobs:

```
register('c');
register('b', 'a');
register('c', 'b');
```

`sort()` should return this result: "abc".

The same job appeared in multiple registrations just occurs once in the result of `sort()`. Jobs with no dependency can occur in any order as long as they are executed before jobs which depend on them. Circular dependencies between jobs should be reported through an exception (at least in the method `sort()`).

Write a unit test.

Task 9. Bounded Blocking Queue

Develop a class that represents a queue having a certain length. It should be used for communication between threads.

Reading threads remove elements from the head of the queue, blocking while the queue is empty and waiting for a new element. Writing threads add elements at the end of the queue, blocking when the queue is full, waiting for elements being removed from the queue.

The interface is defined as:

```
interface BoundedBlockingQueue<E> {
    void add(E);
    E remove();
}
```

Example with `queue length = 2`:

Writing thread	Queue	Reading thread
	`[]`	
`add(1)`	`[1]`	
		`remove() -> 1`
`add(2)`	`[2]`	
`add(3)`	`[2,3]`	
`add(4) -> blocks`		
	`[3]`	`remove() -> 2`
`add(4) -> unblocks`	`[3,4]`	
	`[4]`	`remove() -> 3`
	`[]`	`remove() -> 4`
	`[]`	`remove() -> blocks`
`add(5)`	`[5]`	
		`remove() unblocks -> 5`

Do not use any existing implementations, plan and code your own.

Write a unit test.