OBJECT ORIENTED PROGRAMMING

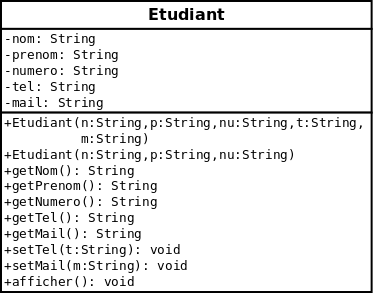
Lab 7: OOP inheritance (2)

### Exercise 1:

We now want to define a “Student” class making it possible to represent the main characteristics of a student in a computer application (such as the one used by the university's tuition service). We will only consider the following information: last name, first name, student number, phone number and email (university email address).

* Define the model of such a class using the UML representation (class diagram).
* b) Implement the “Student” class as well as a separate test class.
* c) Check the compilation and execution steps.
* d) You will take care to specify the useful accessors and mutators as well as the different constructor variants.

Answers

*****1)UML class diageram***

***2)program implementation in java***

**public class Student {**

**private String number;**

**private String firstname;**

**private String lastname;**

**public Student(String num, String first, String last) {**

**number = num;**

**firstname = first;**

**lastname = last;**

**}**

**public void display() {**

**System.out.println("Student [number=" + number + ", firstname=" + firstname**

**+ ", lastname=" + lastname + "]");**

**}**

**}**

**public class TestStudent {**

**public static void main(String[] args) {**

**Student jean = new Student("123", "Jean", "Dupont");**

**Student paul = new Student("456", "Paul", "Dubois");**

**jean.display();**

**paul.display();**

**}**

**}**

### Exercise 2:

We want to create a simple counter with an initial value of zero. It is therefore a matter of creating a Counter class to provide the requested service:

write in particular the methods incrementer () and decrementer (). We will also write a small test class which:

1. will create a counter and display its value;
2. increment it 10 times, then display its value again;
3. decrement it 20 times, then display its value a third time

If necessary, we can initially not worry about the packages but nevertheless write each

class (public) in separate source files (.java): Compteur.java and TestCompteur.java. The classes created will then be in the default package.

This program should display (something like) "0 10 0".

**(\*optional): Modify the Counter class by adding a second Counter constructor initialized to a positive starting value. Add if not already done a reset method.**

Answers

**public class Counter {**

**/\*\***

**\* The current value of the counter should always be positive.**

**\*/**

**private int value;**

**public Counter(int val) {**

**value = val;**

**}**

**@Override**

**public String toString() {**

**return "Counter [value=" + value + "]";**

**}**

**public int getValue() {**

**return value;**

**}**

**public void setValue(int val) {**

**value = val;**

**}**

**public void increment() {**

**value++;**

**}**

**public void decrement() {**

**if (value > 0) {**

**value--;**

**}**

**}**

**public void init() {**

**value = 0;**

**}**

**}**

### Exercise 3:

We now want to define a BoundedCounter class, linked to the previous Counter class by an inheritance relation.

A bounded counter is limited in high value (it cannot exceed a maximum defined during its creation).

Define the UML model of such a class (by completing the diagram of the simple counter exercise).

Write the classes BoundedCounter and TestCompteurBorne

Answers

**public class BoundedCounter extends Counter {**

**private int max;**

**public BoundedCounter(int value, int valueMax) {**

**super(value);**

**max = valueMax;**

**}**

**@Override**

**public void increment() {**

**if (getValue() < max) {**

**super.increment();**

**}**

**}**

**@Override**

**public String toString() {**

**return "BoundedCounter [value=" + getValue() +"]";**

**}**

**protected int getMax() {**

**return max;**

**}**

**}.**

### Exercise 4:

We now want to design a CyclicCounter class which will return to zero in the case of an incrementation when the value of the cyclic counter has reached its maximum, and conversely will return to its maximum during a decrementation of a cyclic counter whose value is zero. .

Define the UML model of this new class (by completing the preceding class diagram).

Write the classes CyclicCounter and TestCyclicCounter.

Answers

**public class CyclicCounter extends BoundedCounter {**

**public CyclicCounter(int value, int max) {**

**super(value, max);**

**}**

**@Override**

**public void increment() {**

**if (getValue() < getMax()) {**

**super.increment();**

**} else {**

**setValue(0);**

**}**

**}**

**@Override**

**public void decrement() {**

**if (getValue() > 0) {**

**super.decrement();**

**} else {**

**setValue(getMax());**

**}**

**}**

**@Override**

**public String toString() {**

**return "CyclicCounter [value=" + getValue() + "]";**

**}**

**}**

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*ALL test progrms\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

**public class TestCounter {**

**public static void main(String[] args) {**

**Counter counter = new Counter(0);**

**System.out.println(counter.toString());**

**for (int i = 1; i <= 10; i++) {**

**counter.increment();**

**}**

**System.out.println(counter.toString());**

**for (int i = 1; i <= 20; i++) {**

**counter.decrement();**

**}**

**System.out.println(counter.toString());**

**BoundedCounter boundedCounter = new BoundedCounter(0, 5);**

**System.out.println(boundedCounter.toString());**

**for (int i = 1; i <= 10; i++) {**

**boundedCounter.increment();**

**}**

**System.out.println(boundedCounter.toString());**

**CyclicCounter cyclicCounter = new CyclicCounter(0, 5);**

**System.out.println(cyclicCounter.toString());**

**for (int i = 1; i <= 10; i++) {**

**cyclicCounter.increment();**

**}**

**System.out.println(cyclicCounter.toString());**

**for (int i = 1; i <= 10; i++) {**

**cyclicCounter.decrement();**

**}**

**System.out.println(cyclicCounter.toString());**

**}**

**}**