ASE Exercise 5 (Fall 2021)

Task 1 (static)

Create a new class `Bicycle`.

- It has four private instance variables of type `int`: `cadence`, `gear`, `speed` and `id`.
- It has one private class variable `numberOfBicycles` of type `int`.
- It has one constructor to initialize the instance variables `cadence`, `gear`, `speed`, and `id`, whereby `id` is set to the value of `numberOfBicycles` and `cadence`, `gear` and `speed` are set by corresponding parameters. The class variable `numberOfBicycles` has to be increased by one, whenever a new object of type `Bicycle` is created. The increment has to be calculated directly at the beginning of the constructor, so that the first object gets the id 1.
- The class variable `numberOfBicycles` should be initialized with 0.
- Implement public getters for all instance variables.

Create a second class `Application`.

- Implement the main-method and create three instances of class `Bicycle`. You can define the initial parameter assignment yourself. Validate if the id of the bicycles is set correctly by adding a console printout.

Task 2 (Use Case Diagram)

Draw a use case diagram using the following description. For the naming of the use cases, you can orientate yourself on the underlined terms. Bold letters depict the actors and the name of the system.

The Enterprise Resource Planning of a factory is implemented by the software system **ERPOS**.

An **engineer** and a **manager** use the system.

- The engineer can start a self-diagnosis of the system.
- The manager can add resources.
- The manager can plan resources. In case a verification, if the resource is available, is desired (condition “Verification desired” is satisfied), the system can check the availability.
- Both actors can trigger a restart of the software. In this case, a restart protocol is printed by the system.

Task 3 (Component Diagram)

Draw a component diagram for the following three-tier architecture. For the naming of the components use the underlined terms, for the naming of the interfaces the bold terms.

An accounting software for a factory enables its users to analyze the production performance.

The GUI uses the interface **get results** from the performance analysis to display results to the user. The performance analysis can **get all entries** from the production database and **request the most recent entry** from it.

Task 4

Implement the method **public boolean noTriples(int[] nums)**.

Given an array of ints, we say that a triple is a value appearing 3 times in a row in the array. Return true if the array does not contain any triples.

Test set

```
noTriples({1, 1, 2, 2, 1}) → true
noTriples({1, 1, 2, 2, 2, 1}) → false
noTriples({1, 1, 1, 2, 2, 2, 1}) → false
```

Task 5

Implement the method **public boolean makeBricks(int small, int big, int goal)**.

We want to make a row of bricks that is goal inches long. We have a number of small bricks (1 inch each) and big bricks (5 inches each). Return true if it is possible to make the goal by choosing from the given bricks.

Additional challenge: You are not allowed to use loops.

Test set

```
makeBricks(3, 1, 8) → true
makeBricks(3, 1, 9) → false
makeBricks(3, 2, 10) → true
```

Task 6

Implement the method **public boolean groupNoAdj(int start, int[] nums, int target)**.

Given an array of ints, is it possible to choose a group of some of the ints, such that the group sums to the given target with this additional constraint: If a value in the array is chosen to be in the group, the value immediately following it in the array must not be chosen.

Additional challenge: You are not allowed to use loops.

Test set

```
groupNoAdj(0, {2, 5, 10, 4}, 12) → true
groupNoAdj(0, {2, 5, 10, 4}, 14) → false
groupNoAdj(0, {2, 5, 10, 4}, 7) → false
```