## CS-115 Lab Class 2 03/02/2020

## ☐ Task 2.1

Swansea Museum has recruited you to be part of a team producing an exhibit which simulates a historic Television Set (TV). You are responsible for coding an object which models such a TV.

The following is an informal natural language specification of the historic TV:

> A TV has a power state which can be toggled *on* and *off* using a single power button. That is, we are modelling a single toggle button as opposed to two separate on and off buttons.
>
> If a TV is currently in the *on* state then the following operations are possible:
>
> - The current channel number can be set or read.
> - The current channel number can be incremented or decremented by 1 (i.e., next and previous channels).
> - The volume level can be incremented and decremented in increments of 2%. The volume level cannot be set directly, but it can be read directly. The volume level should be capped and always remain within the range of 0 to 100.
>
> If a TV is currently in the *off* state then the **only** operation that can be performed is to toggle the power to *on*.

Your task is to model a TV using a Java class. You will also need to provide a main function (preferably in a separate class) to test out your TV class implementation. The code in the main class is only to test the implementation of the TV class, it does not need to do any real computation. It can be as simple as a few hard coded lines. Make sure you follow the coding conventions.

You need to pay careful consideration to what parameters are used in your constructor and methods, along with exactly what setter, getter and other methods you make available. For example, as you cannot set the *on* and *off* power status of the TV directly, there should be no setter method for the power state; there should however be a toggle method (which changes the power state to *on* when the TV is *off*, and to *off* when the TV is *on*) that takes no parameters.

If a method is called which is not available (e.g., an attempt to change the channel when the TV is off, or to read the volume level when the TV is off) then you should print out an error message to the screen. This will abort the current operation (i.e., method). The state of the TV will stay the same as before the operation was executed.

Note: You do not need to imposed limits on the channel number for this task. The channel numbers are allowed to go negative or get very large.

## ☐ Task 2.2

Extend your TV class so that each TV that is created has a **unique automatically generated serial number** (e.g., an integer). The serial number needs to be guaranteed unique. You cannot use a random number. The first available serial number is 1000. Negative serial numbers are not allowed. A serial number of 0 is not allowed.

It should be possible to inspect (get) each individual TV's serial number. However, It should

be impossible to set the serial number of a TV.

Hint: **static** is very useful to implement this efficiently.

## ☐ Task 2.3

In your main method create an array that can hold 5 TV objects. Create 4 TVs (do not create 5 TVs). These TVs will have the serial numbers 1000, 1001, 1002, and 1003. Place these four TVs in increasing order of their serial numbers into entries 0-3 of the array (1000 in 0, 1001 in 1 etc). Finally, for the last position in the array (index 4), store a reference to 1002 (index 2).

Write code that does the following. Set the **all** the TVs to channel number 1. Increment the channel of the TV index 2 Print the TV's channel number at index 4.

Explain why the channel at index 4 of the array is also incremented.

## ☐ Task 2.4

Set the value of the array index 4 to `null` and toggle the on/off state of the TV at this index. What happens and why?

## ☐ Task 2.5

Upon creation of a TV, a channel limit should be specified which should cap the maximum channel number, e.g., expensive TV sets might cap the maximum channel number to 999, while cheaper models might cap it to 10. This limit cannot be changed once a TV is created.

The user should not be able to set the channel number so it exceeds this limit. The lowest channel number is always 1. If a user tries to set the channel outside of this range then an exception should be used to abort the operation. You should use an `IllegalArgumentException` – this is a standard exception which indicates that a method has been passed an illegal or inappropriate argument.

*If you don't know exceptions, you should look it up in the Java API. Do a search for "java throw exceptions" and use the official Oracle page. Also, look up the IllegalArgumentException.*

You should provide two constructors:

- A constructor that takes no parameters and sets the channel limit to 10 channels automatically.
- A constructor that takes a single `int` parameter. This value is used as the channel limit. If the value of the parameter is below 1 (i.e., negative or zero) then an exception should be used to abort the construction of the instance. You should again use an `IllegalArgumentException`.

## ☐ Challenge Task 2.6

The channel number should cycle round. That is, assuming the channel limit is 100, then:

- decrementing the channel number when it is 1 will result in the channel number being set to 100; and
- incrementing the channel number when it is 100 will result in the channel number being set to 1.

Note: The volume level does not wrap around in this fashion.