Smart Home Device Monitoring System

Assignment brief:

Time 2hr 30 min (open book – no external websites).

Create a **project solution** (named **P2<your name and student id> e.g. P2CollinsMatthew12345678). Ensure your name and student number are placed in the Javadoc comments of all the classes / Test classes etc you create.**

Part 1 – 40%

You have been tasked to design, implement, and unit test part of an Smart Home Device Monitoring and Management application.

The system is expected to support and interact with several specific **Device types** such as **Smart Plug**, **Smart Bulb**, **Smart Fan**, **Smart Radiator**. You are the first developer of the system and are tasked with designing and implementing the **Smart Radiator** type, although you are encouraged to design the system to support the future development of other specific future Smart Home Device types.

Data required for the system.

DATA	DESCRIPTION	APPLICATION	BUSINESS RULES
NAME	The Name Given to the Device	All Devices	Text: min length 3 and max 20 E.g. Warm White Bulb, Power Plug 2000 etc.
MANUFACTURER	The Company/Brand who makes the device	All Devices	Text: min length 3 and max 20 E.g. Philips Hue, Signify, Tado, Google Nest etc.
ROOM	The kind of Room where the device has been installed.	All Devices	Allowable values (at this stage of development) House, Kitchen, Bedroom, Bathroom, Lounge
POWERSTATE	The current power state of the device	All Devices	Allowable values ON,OFF
TEMPNOW	The current temperature in Celsius as read from temperature sensors	Radiators Only	Numerical data which should be capable of recording temperatures Allowable values: -10 to +30 Degrees Celsius (inclusive) If setter for current temperature is called with a value lower than target temp, then the Powerstate should be switched to ON Similarly, if current temperature is set to a value higher than or matching target, Powerstate should be switched to OFF
TEMPTARGET	Target temperature for smart radiator	Radiators Only	Numerical data Allowable values 5 to 26 (inclusive)

Any attempt to set a field to a value outside allowable rules should raise an appropriate exception

Additional methods:

showAll()

you have been asked to create a **showAll** method that will output to screen (console) all data for each Device instance. Note you do not need to Unit Test this functionality.

For example, the expected output for a Smart Radiator should be:

```
NAME         : Radiator Stat
MANUFACTURER : Tado
ROOM         : BEDROOM
POWERSTATE   : ON
TEMPNOW      : 18.7 Degrees
TEMPTARGET   : 21.5 Degrees
```

(note numerical values presented to 1 decimal place)

status()

It is a requirement of all Device types i.e. Smart Plug, Smart Bulb, Smart Fan and Smart Radiator to have a method to be named **status**. However, each of the Device types will implement the method differently. The method takes no parameter arguments but generates and returns a string.

The implementation of the status method for the Smart Radiator will return a string in the format SR-NAME-ROOM-NOW:X.X-TARGET:X.X-POWERSTATE

e.g. For a Radiator with values as shown in the showAll example above, the expected result would be

SR-RADIATORSTAT-BEDROOM-NOW:18.7-TARGET:21.5-ON

(Note the use of uppercase, the lack of spaces, and the numerical values presented to 1 decimal place).

Part 2 - Device Search class - 20%

Create an ***DeviceSearch*** class to support the system for searching. Each search method should be ***static*** and accept and return an ***ArrayList*** of the **appropriate type**. Using your knowledge of OOP you should create the following functionality based on the following:

1. Create a **searchByRoom** method i.e. search for all objects in the parameter argument ArrayList that match a specified Room (e.g. House, Kitchen, Bedroom, Bathroom, Lounge). You should return an ArrayList containing any that satisfy the search criteria.

2. Create a **searchByTemp** method i.e. search for all objects in the parameter argument ArrayList that have a current temperature *within a specified lower and upper range* e.g. between 15 and 22 (range inclusive of both values). You should return an ArrayList containing any that satisfy the search criteria.

Part 3 – Testing – 40%

1. Unit Test the application.

When complete, upload a zipped archive of the entire **Eclipse Project Structure**, and submit it to **Assignments (P2 assessment)** on CANVAS.
Check you have uploaded the correct file. It is your responsibility to ensure you have submitted the correct file.

(Files can be redownloaded to check if necessary after submission)

Remember you must also record and upload a short narrated walk through video of your submission (separate assignment). Any Submission without a corresponding narrated video will not be accepted.

[END]