Insurance Policy system

Assignment brief:

Time 2hr 30 min (+15 min to upload) (open book – no external websites).

Create a **project solution** (named **P2<your name and student id> e.g. P2McGowanAidan6048201). 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 Insurance Policy application.

The system is expected to support many specific Insurance Policy types such Motor, Home, Travel, Medical etc. You are the first developer of the system and are tasked with designing and implementing the Motor policy, although you are encouraged to design the system to support the other specific future policy types.

Data required for the system.

DATA	DESCRIPTION	APPLICATION	BUSINESS RULES
SURNAME	The surname of the policyholder	All policies	String: min length 3 and max 20
AGE	The age (whole number) of the policyholder	All policies	Whole number: range 18 and 50 (inclusive)
MOTOR TYPE	The motor type category.	Motor Policy only	Allowable values CAR, TAXI, BUS
MOTOR POLICY ID	A reference ID for each motor policy. A string that consists of three parts with the format: **<SUR><TIME STAMP><PARITY>** Note the <> brackets are shown here to separate the parts and are not part of the string (see business rule example).	Motor Policy only	<SUR> the first three characters of the policy holder's surname. <TIME STAMP> a numeric value based on a time stamp – see the code example (next page) * consists of the sum of the current year and month. <PARITY> An additional 0 or 1 is added to the end of the string depending on the Time Stamp value, whereby if the Time Stamp is an EVEN number then 0 (Zero) is added to the end and if ODD then a 1 is added to the end eg MCG20231 <SUR><TIME STAMP><PARITY> where the surname was MCGOWAN the Timestamp was 2023 the parity is 1 as the timestamp is an ODD number.

*Suggested code to create the **timestamp** :

```
import java.util.Calendar;

…

// time stamp number
int year = Calendar.getInstance().get(Calendar.YEAR);
int month = Calendar.getInstance().get(Calendar.MONTH) +1;

// sum of the year and month
int timeStamp = year+month;
```

Additional method: you have been asked to create a **displayAll** method which will output to screen (console) all data for each Policy instance. Note you do not need to Unit Test this functionality.

For example, the output for the Motor Policy would be :

```
Surname      :MGGOWAN
Age          :40
Policy Ref   :MGG20231
Motor Type   :CAR
```

Part 2 - Policy Search class - 20%

Create a ***PolicySearch*** class to support the system for search policies. 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 **searchByAge** method i.e. search for all objects in the parameter argument ArrayList that are owned by Policy holders *within a specified age range* e.g. between 20 and 30 (range inclusive of both values). You should return an ArrayList containing any that satisfy the search criteria.

2. Create a **searchForAllByMotorPolicyType** method i.e. search for all objects in the parameter argument ArrayList that match a specified Motor type (eg Car, Taxi, Bus). You should return an ArrayList containing any that satisfy the search criteria.

Part 3 – Testing – 40%

1. Unit Test the application – (all code from parts 1 and 2).

When complete compress (zip) the entire ***Eclipse solution*** and upload it to **Assignments** (P2 assessment) on CANVAS. Remember to record and then upload a short commentary walk-through of the code of your solution – also run the test classes. Keep the separate screen recording safe (no need to upload at this point).

Now : check the uploads to ensure you have submitted the correct files.

[END]