CMPSC 221 **Name: _____**

OO Programming with Web-Based Applications
Spring 2022
Due Date: Friday, January 28th, 2022 (11:59 p.m. ET)

Instructions: Please create a Java application to solve the following problem. Documentation requirements follow the problem specification. Upload digital copies of your program source code and **two or more** sample runs to Canvas by the deadline. For the source code, you may: 1) compress (i.e., zip) your entire project folder (NetBeans, Eclipse, etc.) and upload that to Canvas OR 2) place your Java source code file(s) in a folder, compress the folder, and upload that to Canvas. For the sample runs, copy and paste the output for two complete runs of the program into ***MS Word***, save the Word document, and upload that to Canvas. You may submit your program source code and sample runs in a single compressed archive, if you like. **Important Notes:** Please be sure to upload your Project 1 files to Canvas by the deadline. If you miss the deadline, you WILL automatically receive a grade of 0 (zero). Also, e-mail submissions are NOT allowed.

---

1. Your task is to create a Java application that implements a simple Vehicle Identification Number (VIN) decoder. A VIN is a unique code that an automobile manufacturer assigns to an individual vehicle. Although in use since 1954, VINs were not standardized until 1981. The standardized format requires that a VIN contain 17 characters consisting only of letters and digits. (Technically, the letters I, O, and Q are not allowed.) A VIN is composed of 3 main sections:
   - World Manufacturer Identifier (WMI) – Characters 1 through 3 indicate the country in which a vehicle was made and the manufacturer who made it.
   - Vehicle Descriptor Section (VDS) – Characters 4 through 8 identify the vehicle type and may include information about model, body style, and engine type. Character 9 is a check digit used to detect invalid VINs.
   - Vehicle Identifier Section (VIS) – Characters 10 through 17 identify the model year, manufacturing plant, and serial number.

   Specific vehicle information that can be retrieved without too much difficulty is: country of origin (character 1), manufacturer (character 2), check digit (character 9), model year (character 10), and serial number (characters 12 through 17).

   Country codes that we will use for this assignment are:

1, 4, 5 = United States	J = Japan
2 = Canada	S = United Kingdom
3 = Mexico	W = Germany

   Manufacturer codes that we will use for this assignment are:

C = Chrysler	T = Toyota
F = Ford	B = BMW
G = General Motors	A = Jaguar
H = Honda	D = Mercedes-Benz

   Model years will be limited to 2000 and later. Model year encodings can be viewed at: https://en.wikipedia.org/wiki/Vehicle_identification_number#Model_year_encoding .

   Your program begins by prompting the user to enter a string representing a VIN. If the VIN is valid, it is decoded to display the 3 groups (WMI, VDS, VIS) and 5 attributes (country, manufacturer, check digit, model year, serial number) described above. If it is invalid, an appropriate error message is displayed. The

program then asks the user if s/he wishes to enter another VIN. Depending upon the user's response, the program either prompts for another VIN or thanks the user for using the VIN decoder and ends. Your program output should *strongly* resemble the sample runs at the end of this document. I encourage you to use `System.out.printf` to format the output as shown.

Implementation Requirements:

- This project is primarily an exercise in learning to use the Java `String` class and accompanying methods. Consequently, you may **not** use data structures such as arrays or `ArrayList`s. You may use related classes such as `StringBuffer` or `StringBuilder`, however.
- The string representing the VIN **must** be case-insensitive. It may initially contain spaces, but they must be removed before it is checked for validity.
- **Required** validity checks for the VIN string include checking that it contains exactly 17 characters and that those characters consist only of letters and digits. (Checking that the letters I, O, and Q are not present is optional.)
- Y/N (i.e., yes/no) responses **must** be case-insensitive **and** validated. In other words, only responses of y, Y, n, or N are allowed.
- You may implement this program using one or more Java classes. You may use methods in addition to `main`, even though methods of other types are not introduced until Chapters 4 and 5.

References:

- Savitch, 6th Edition: Chapters 1 – 3 (Java basics)
- Vehicle identification number (https://en.wikipedia.org/wiki/Vehicle_identification_number)
- VIN Decoder & Lookup (https://driving-tests.org/vin-decoder/)

Documentation Requirements:

1) Each program source code file (i.e., Java class) must have a header at the beginning of the class containing the following:

   - Name of author, PSU e-mail address of author, name of course, assignment number and due date, name of file, purpose of class, compiler/IDE, operating system, and any external references used (e.g., Website)
   - Example:

```
/*
Author:          Wanda Kunkle
E-mail:          wmk12@psu.edu
Course:          CMPSC 221
Assignment:      Programming Assignment 1
Due date:        1/28/2022
File:            VIN_Decoder.java
Purpose:         Java application that implements a simple Vehicle
                 Identification Number (VIN) decoder
Compiler/IDE:    Java SE Development Kit 17.0.1/Apache NetBeans 12.6
Operating
system:          MS Windows 10 Home
Reference(s):    Java 17 API - Oracle Documentation
                 (https://docs.oracle.com/en/java/javase/17/docs/api/);
                 (Include ALL additional references (Web page, etc.) here.)
*/
```

2) The purpose of each method in the source code file(s) must be documented as shown in the example below. I prefer that you use the **javadoc** comment style.

```
/** This method checks a VIN string for valid length.
 *
 * @param vin    The vin to check
 * @return       True if the length is valid, false otherwise
 */
public static boolean checkVinLength(String vin)
{
   // Method definition (i.e., body)
}
```

Sample run #1:

```
cd C:\Users\Owner\Desktop\VIN_Decoder_NB_12_6; "JAVA_HOME=C:\\Program Files\\Java\\jdk-17.0.1"
cmd /c "\"C:\\Program Files\\NetBeans-12.6\\netbeans\\java\\maven\\bin\\mvn.cmd\" -Dexec.vmArgs=
\"-Dexec.args=${exec.vmArgs} -classpath %classpath ${exec.mainClass} ${exec.appArgs}\" -
Dexec.appArgs= -Dexec.mainClass=com.mycompany.vin_decoder_nb_12_6.VIN_Decoder \"-
Dexec.executable=C:\\Program Files\\Java\\jdk-17.0.1\\bin\\java.exe\" \"-
Dmaven.ext.class.path=C:\\Program Files\\NetBeans-12.6\\netbeans\\java\\maven-nblib\\netbeans-
eventspy.jar\" -Dfile.encoding=UTF-8 org.codehaus.mojo:exec-maven-plugin:3.0.0:exec"
Running NetBeans Compile On Save execution. Phase execution is skipped and output directories of
dependency projects (with Compile on Save turned on) will be used instead of their jar artifacts.
Scanning for projects...

-----------------< com.mycompany:VIN_Decoder_NB_12_6 >------------------
Building VIN_Decoder_NB_12_6 1.0-SNAPSHOT
--------------------------------[ jar ]---------------------------------

--- exec-maven-plugin:3.0.0:exec (default-cli) @ VIN_Decoder_NB_12_6 ---

********************************************************************************
                                    VIN Decoder
********************************************************************************

Please enter a string representing a Vehicle Identification Number (VIN):
WDDZH6JB1KA505518

VIN is valid. Decoding in progress ...

Values by group:
  World Manufacturer Identifier (WMI): WDD
  Vehicle Descriptor Section (VDS):    ZH6JB1
  Vehicle Identifier Section (VIS):    KA505518

Vehicle attributes:
  Country of manufacture: Germany
  Manufacturer:           Mercedes-Benz (Germany)
  Check digit:            1
  Model year:             2019
  Serial number:          505518

Enter another VIN (Y/N)?
Y

********************************************************************************

Please enter a string representing a Vehicle Identification Number (VIN):
2T1-BU4EE-CC84882

** VIN can only contain letters and digits. **

Enter another VIN (Y/N)?
Y

********************************************************************************

Please enter a string representing a Vehicle Identification Number (VIN):
sajaj4BV9ha957333
```

```
VIN is valid. Decoding in progress ...

Values by group:
  World Manufacturer Identifier (WMI): SAJ
  Vehicle Descriptor Section (VDS):    AJ4BV9
  Vehicle Identifier Section (VIS):    HA957333

Vehicle attributes:
  Country of manufacture: United Kingdom
  Manufacturer:           Jaguar (United Kingdom)
  Check digit:            9
  Model year:             2017
  Serial number:          957333

Enter another VIN (Y/N)?
n

************************************************************************************
Thanks for using my VIN decoder!
************************************************************************************
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
Total time:  08:56 min
Finished at: 2022-01-16T20:55:46-05:00
------------------------------------------------------------------------
```

Sample run #2:

```
cd C:\Users\Owner\Desktop\VIN_Decoder_NB_12_6; "JAVA_HOME=C:\\Program Files\\Java\\jdk-17.0.1"
cmd /c "\"C:\\Program Files\\NetBeans-12.6\\netbeans\\java\\maven\\bin\\mvn.cmd\" -Dexec.vmArgs=
\"-Dexec.args=${exec.vmArgs} -classpath %classpath ${exec.mainClass} ${exec.appArgs}\" -
Dexec.appArgs= -Dexec.mainClass=com.mycompany.vin_decoder_nb_12_6.VIN_Decoder \"-
Dexec.executable=C:\\Program Files\\Java\\jdk-17.0.1\\bin\\java.exe\" \"-
Dmaven.ext.class.path=C:\\Program Files\\NetBeans-12.6\\netbeans\\java\\maven-nblib\\netbeans-
eventspy.jar\" -Dfile.encoding=UTF-8 org.codehaus.mojo:exec-maven-plugin:3.0.0:exec"
Running NetBeans Compile On Save execution. Phase execution is skipped and output directories of
dependency projects (with Compile on Save turned on) will be used instead of their jar artifacts.
Scanning for projects...

-----------------< com.mycompany:VIN_Decoder_NB_12_6 >------------------
Building VIN_Decoder_NB_12_6 1.0-SNAPSHOT
--------------------------------[ jar ]---------------------------------

--- exec-maven-plugin:3.0.0:exec (default-cli) @ VIN_Decoder_NB_12_6 ---

********************************************************************************
                                    VIN Decoder
********************************************************************************

Please enter a string representing a Vehicle Identification Number (VIN):
JHMFC1F9XKX00708

** VIN must contain 17 characters. **

Enter another VIN (Y/N)?
y

********************************************************************************

Please enter a string representing a Vehicle Identification Number (VIN):
JHMFC1F9XKX007088

VIN is valid. Decoding in progress ...

Values by group:
  World Manufacturer Identifier (WMI): JHM
  Vehicle Descriptor Section (VDS):    FC1F9X
  Vehicle Identifier Section (VIS):    KX007088

Vehicle attributes:
  Country of manufacture: Japan
  Manufacturer:           Honda (Japan or North America)
  Check digit:            X
  Model year:             2019
  Serial number:          007088

Enter another VIN (Y/N)?
Y

********************************************************************************

Please enter a string representing a Vehicle Identification Number (VIN):
 1FM 5K8F8X EGB32351
```

```
VIN is valid. Decoding in progress ...

Values by group:
  World Manufacturer Identifier (WMI): 1FM
  Vehicle Descriptor Section (VDS):    5K8F8X
  Vehicle Identifier Section (VIS):    EGB32351

Vehicle attributes:
  Country of manufacture: United States
  Manufacturer:           Ford (North America)
  Check digit:            X
  Model year:             2014
  Serial number:          B32351

Enter another VIN (Y/N)?
y

********************************************************************************

Please enter a string representing a Vehicle Identification Number (VIN):
2C3CCABG0KH708126

VIN is valid. Decoding in progress ...

Values by group:
  World Manufacturer Identifier (WMI): 2C3
  Vehicle Descriptor Section (VDS):    CCABG0
  Vehicle Identifier Section (VIS):    KH708126

Vehicle attributes:
  Country of manufacture: Canada
  Manufacturer:           Chrysler (United States)
  Check digit:            0
  Model year:             2019
  Serial number:          708126

Enter another VIN (Y/N)?
y

********************************************************************************

Please enter a string representing a Vehicle Identification Number (VIN):
4T1C11AK7NU658023

VIN is valid. Decoding in progress ...

Values by group:
  World Manufacturer Identifier (WMI): 4T1
  Vehicle Descriptor Section (VDS):    C11AK7
  Vehicle Identifier Section (VIS):    NU658023

Vehicle attributes:
  Country of manufacture: United States
  Manufacturer:           Toyota (Japan or North America)
  Check digit:            7
  Model year:             2022
  Serial number:          658023

Enter another VIN (Y/N)?
x
```

```
** You may only choose Y or N. **

Enter another VIN (Y/N)?
n

********************************************************************************
Thanks for using my VIN decoder!
********************************************************************************
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
Total time:  40.112 s
Finished at: 2022-01-16T21:36:36-05:00
------------------------------------------------------------------------
```

Sample VINs for Test Runs:

```
WDDZH6JB1KA505518 (Germany, Mercedes-Benz, 2019)

2T1-BU4EE-CC84882 (invalid VIN)

SAJAJ4BV9HA957333 (United Kingdom, Jaguar, 2017)

sajaj4BV9ha957333 (same as above)

JHMFC1F9XKX007088 (Japan, Honda, 2019)

1FM5K8F8XEGB32351 (United States, Ford, 2014)

 1FM 5K8F8X EGB32351 (same as above)

2C3CCABG0KH708126 (Canada, Chrysler, 2019)

4T1C11AK7NU658023 (United States, Toyota, 2022)
```