Data Encapsulation

# Before Class

1. What is the mechanism whereby the implementation details of a class are kept hidden from the user. Familiarise yourself with the data encapsulation concept.

**encapsulation** refers to the bundling of data with the methods that operate on that data, or the restricting of direct access to some of an object's components.[[1]](https://en.wikipedia.org/wiki/Encapsulation_(computer_programming)#cite_note-Rogers01-1) Encapsulation is used to hide the values or state of a structured data object inside a [class](https://en.wikipedia.org/wiki/Class_(computer_programming)), preventing direct access to them by clients in a way that could expose hidden implementation details or violate state invariance maintained by the methods.

Inaczej hermetyzacja danych

1. What are the access modifiers: public, private, protected and the default value. How these access modifiers are used.  
   PUBLIC – accessible from any other class  
   PRIVATE – accessible only within the declared class itself  
   PROTECTED – provides the same access as the default access modifier, with the addition that subclasses can access protected methods and variables of the superclass  
   DEFAULT – a variable or method declared with no access control modifier to any other class in the same package

FOR CLASSES  
PUBLIC – the class is accessible by any other class  
DEFAULT – the class is accessible only by classes in the same package

1. Watch the video “Java encapsulation”:

<https://youtu.be/eboNNUADeIc>

1. What are getters and setters methods and how they support data protection.

Getters and setters are **used to protect your data, particularly when creating classes**. For each instance variable, a getter method returns its value while a setter method sets or updates its value. Given this, getters and setters are also known as accessors and mutators, respectively.

# During Class

## Access modifiers

1. Find an image of a driving license on the Internet. Take a look at what information the driving license contains. Then define the DrivingLicense class, containing the following attributes: driver's name and surname, address, postal code, city, driving license number, year of issue and driving license category. Use private access modifiers when declaring attributes. Then create a driving license and try to assign values to the attributes. What message are you getting? After that, change the access modifiers from private to public and try to assign values once again.
2. Complete the DrivingLicense class. Define a method that displays the driving license (all data contained on the driving license). Try to display the data in an attractive form. What access modifier will you apply to the defined method?

## Getters and Setters

1. Apply the private access modifier to all attributes of the DriverLicense class. Then create the get and set methods for each of the attributes. After that, modify the program and the method displaying the driving license. Replace attributes names with get and set methods.
2. Add a toString() method in the DrivingLicense class to return driving license information. Use getter methods to get the driving license data. Then check the method in action.
3. In the DrivingLicense class, modify the setter method for the year of issue attribute. The valid value of the attribute should be greater than or equal to 1980 and less than or equal to the current year. If the given value is different, do not change the attribute.
4. Modify the setName() method in the DrivingLicense. Regardless of the given name value, the value of the attribute should begin with a capital letter. Replace the remaining letters with lowercase. Then check the method in action. Tip. Use the methods available in the String class.

# After Class

1. The Product class describes food products using two attributes: the product name and whether the product is vegetarian. Define the class, its attributes, and getter and setter methods for all attributes. Then create a product, set attributes' values and display product information.
2. Choose any object. Then define a class that describes such objects. Hide data about an object using data encapsulation.