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.
   1. Enkapsulacja danych, znana również jako ukrywanie danych, to mechanizm, dzięki któremu szczegóły implementacji klasy są ukrywane przed użytkownikiem. Użytkownik może wykonać tylko ograniczony zestaw operacji na ukrytych członkach klasy, wykonując specjalne funkcje powszechnie nazywane metodami, aby zapobiec łatwemu przeglądaniu i dostępowi do atrybutów obiektów.
2. What are the access modifiers: public, private, protected and the default value. How these access modifiers are used.
   1. **Public** - Poziom dostępu modyfikatora private znajduje się tylko w klasie. Nie można uzyskać do niego dostępu spoza klasy.
   2. **Domyślny**: Poziom dostępu modyfikatora domyślnego znajduje się tylko w pakiecie. Nie można uzyskać do niego dostępu spoza pakietu. Jeśli nie określisz żadnego poziomu dostępu, będzie to ustawienie domyślne.
   3. **Protected**: Poziom dostępu chronionego modyfikatora znajduje się w pakiecie i poza pakietem za pośrednictwem klasy podrzędnej. Jeśli nie utworzysz klasy potomnej, nie będzie można uzyskać do niej dostępu spoza pakietu.
   4. **Publiczny**: Poziom dostępu modyfikatora publicznego jest wszędzie. Można uzyskać do niego dostęp z poziomu klasy, spoza klasy, w pakiecie i poza pakietem.
3. Watch the video “Java encapsulation”:

<https://youtu.be/eboNNUADeIc>

1. What are getters and setters methods and how they support data protection.
   1. Getters – za pomocą metody, pozwala odczytać wartość przechowywaną w zmiennej prywatnej
   2. Setters – za pomocą metody, pozwala zapisać wartość do zmiennej prywatnej

# 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.