Abstract Classes

# Before Class

1. Familiarise yourself with the concept of abstract classes.

Abstraction is a process of hiding the implementation details and showing only functionality to the user. It shows only essential things to the user and hides the internal details, for example, sending SMS where you type the text and send the message. You don't know the internal processing about the message delivery.

Nie można za jej pomocą tworzyć obiektów.

* 1. <https://www.javatpoint.com/abstract-class-in-java>
  2. <https://docs.oracle.com/javase/tutorial/java/IandI/abstract.html>

1. How an abstract class is implemented in a programming language.
2. What an abstract class can contain.
3. How an abstract class is represented in a UML class diagram. *(in the italics)*

# During Class

1. An abstract Shape class describes abstract geometric figures. The class contains an abstract area() that returns the area of a geometric figure. Define an abstract class along with an abstract method.
2. Define the Rectangle, Triangle, and Circle classes that derive from the Shape class. Then write a program in which you create one object for each of the ageometric figures and calculate and display the area of these figures.
3. Complete the Shape class with an abstract perimeter() method that returns the perimeter of a geometric figure. Then write a program in which you create one object for each of the geometric figures, and calculate and display both the area and perimeter of these figures.
4. An abstract Message class describes an abstract message. The class contains the text attribute with the message text and accessor and mutator methods for this attribute. The parameterless constructor allows you to create a new, empty message, while the overloaded constructor with one parameter allows you to create a new message with the given content. The class also includes a charNumber() method that returns the number of characters in the message. Moreover, the abstract class includes an abstract send() method for sending a message (displays message along with its details). Define an abstract class and its attributes and methods.
5. Using inheritance, define an SMS class that derives from the Message class and describes text messages sent from a mobile phone. The class contains the phone number attribute to which the text message is sent. The class constructor allows you to initialize a phone number. The class also includes accessor and mutator methods to modify value of the attribute. Then create and send two mobile messages.
6. Using inheritance, define an Email class that derives from the Message class and describes messages sent using email. The class includes the following attributes: message subject and recipient's address. The class also includes accessor and mutator methods for class attributes. Add a constructor that allows you to create an email with the given subject and content that is sent to the given email address. Then create and send two emails.

# After Class

1. Create a User class with the following attributes: first name, last name, email, phone number. Then modify the SMS and Email classes. Use User class objects instead of the phone number and email attributes.
2. Create a UML class diagram that includes the Message, SMS, Email, and User class definitions. Pay attention to the relationships between the classes.
3. Create a project that describes the different types of vehicles. Define an abstract class Vehicle. Consider what attributes and methods (including abstract ones) an abstract class should contain. Add a constructor in the class to create a vehicle. Then, using inheritance, define Car, Bus, Truck classes representing specific types of vehicles. Consider what attributes and methods should these classes contain. Finally, create one car, bus and truck and display details on them.