

**OBJECT-ORIENTED UML, CLASS DEFINITIONS AND METHODS**

**Object-Oriented Paradigm: UML, class definitions and methods**

**Worksheet 2.1**

**Read carefully the list of vocabulary and tell your teacher how many of them you know and what they mean.**

**VOCABULARY**

**Blueprint:**plano

**Cutter:**objeto para cortar

**Template:**plantilla

**Instantiation:** creacion de objetos

**Draw**:dibujar o retirar una carta

**Carried Out:**realizar una tarea

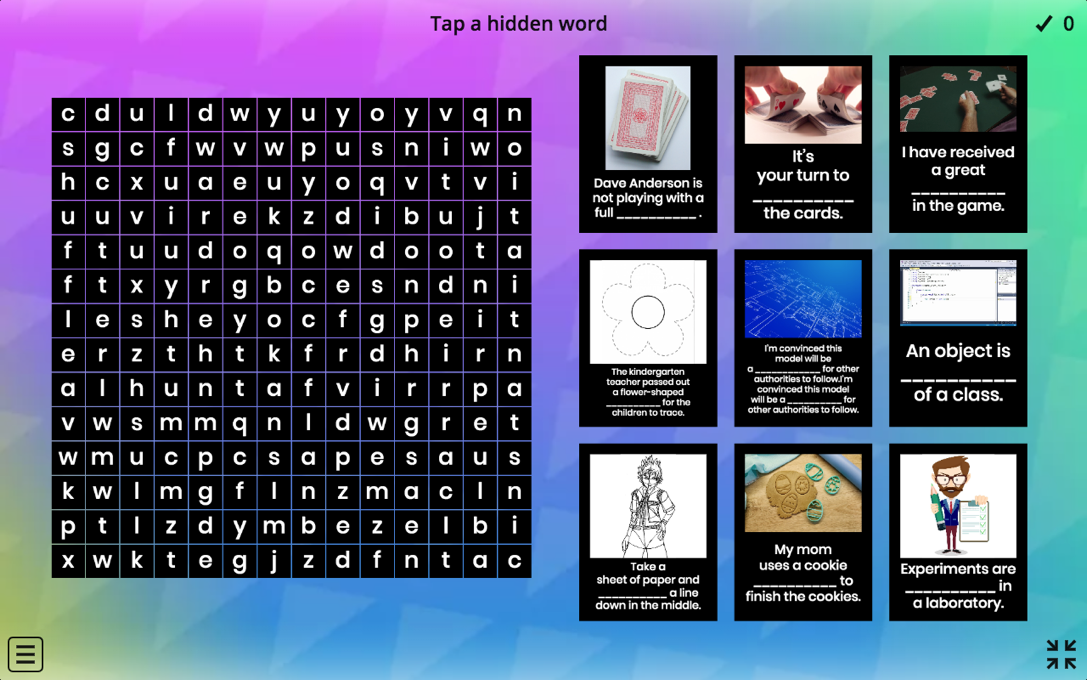
**Shuffle**:mezclar las cartas

**Deck**:baraja

**Deal**: repartir o acuerdo

**Worksheet 2.2**

**Now search for the vocabulary in the word search puzzle, then complete the sentences with the words you found:** [**https://n9.cl/hs4kb**](https://n9.cl/hs4kb)

****

1. I’m convinced this model will be a \_\_\_blueprint\_\_\_\_\_\_\_\_\_ for other authorities to follow.
2. My mom uses a cookie \_\_\_\_cutter\_\_\_\_\_\_\_ to finish the cookies.
3. The kindergarten teacher passed out a flower-shaped \_\_\_\_\_\_template\_\_\_\_\_\_ for the children to trace.
4. An object is an \_\_\_instantiation\_\_\_\_\_\_\_\_\_ of a class.
5. Take a sheet of paper and \_\_\_draw\_\_\_\_\_\_\_\_\_ a line down in the middle.
6. Experiments are \_\_\_\_\_carried out\_\_\_\_\_\_\_ in a laboratory.
7. It’s your turn to \_\_\_\_shuffle\_\_\_\_\_\_\_ the cards.
8. Dave Anderson is not playing with a full \_\_\_\_\_\_deck\_\_\_\_\_\_.
9. I have received a great \_deal\_\_\_\_\_\_\_\_\_\_\_ in the game.

**Worksheet 2.3**

Some active reading techniques involve: highlighting, writing in the margins, and explaining what you read. Make sure that you use these techniques while reading the text.

* You can highlight keywords, topic sentences, concluding sentences or main ideas.
* You can add some comments to explain what you understand from a particular part of the reading.

**OOP design: class definitions**

In the last class, you learned how to identify objects and their attributes. Once you have finished that, yo*u* are ready to start on a formal class definition.

A class definition is a **blueprint** that defines the properties of all objects of the same type. Each object is one of a group of similar objects, and you define the group as a class. The class definition will describe every object of that type.

A classic example to describe a class is a 'cookie **cutter'**. The video clip shows how the **cutter** is used to make a cookie. The **cutter** is a **template** that can be used to make as many cookies as needed. Similar to this, a class definition is a **template** for creating as many objects as necessary. The process of making an object from the class definition is called **instantiation**.

You can **draw** a high-level class diagram for each of the objects. Class diagrams are the main component of Unified Modeling Language (UML), a method that is widely used for modelling OOP systems.

In UML, classes are represented by boxes that contain three sections:

* The top section contains the name of the class.
* The middle section contains the attributes of the class.
* The bottom section contains the methods of the class (the operations that the class can execute).

**OOP design: methods**

When you have finished defining the classes, you should think about methods. Methods are procedures and functions that access or modify the values of the attributes of an object. In fact, the methods must define every operation that is allowed to be **carried out** on an attribute.

In the example of the card **deck**, the collection of cards will need to be **shuffled**. In OOP, you ask the card **deck** to **“*shuffle*”** itself. The cards belong to the card **deck** object and the only way to access them is through a defined method. This method will also belong to the card **deck** object.

During the game, the program will need to **deal** a card. Dealing a card means removing a card from the **deck**. Therefore, you would add a ***“deal”*** method to the class definition. As you may want the program to **deal** more than one card, you would specify that this method will take a parameter to indicate the number of cards to be **dealt**.

At the moment, you are using abstraction, and you are not writing the code. First, you should concentrate on the high-level design before you start to think about the implementation details.

**Worksheet 2.5**

**Wrap-up**

After reading the text about Class definitions and Methods in OOP, in groups create a class diagram, you can create your diagram based on the situation “at school”. It is up to you to decide what class and object you will describe – for example, you can choose to describe people => students/teachers/etc, or you can choose something else.

|  |
| --- |
| **Student** |
| **Name**  **Age**  **Grade** |
| **Study**  **Take Tests**  **Ask Questions**  **Socialize** |

Example:

|  |
| --- |
| **Courses** |
| **id** |
| **name** |
| **showItems** |

|  |
| --- |
| **Teacher** |
| **id** |
| **subjects** |
| **Teaching()** |

|  |
| --- |
| **Director** |
| **Id director** |
| **name** |
| **Planningmeating()** |

|  |
| --- |
| **Classes** |
| **courses** |
| **teacher** |
| **Starclass()** |

**Worksheet 2.4**

**Self-evaluation**

1. Entiendo qué son *class definitions* en OOP.

Yes **😃** Maybe **😐** No **😟**

1. Entiendo qué son *methods* en OOP.

Yes **😃** Maybe **😐** No **😟**

1. Comprendo y pongo en prácticas las técnicas de *active reading*.

Yes **😃** Maybe **😐** No **😟**

1. Las técnicas de *active reading* me ayudan a asimilar mejor la información el texto.

Yes **😃** Maybe **😐** No **😟**