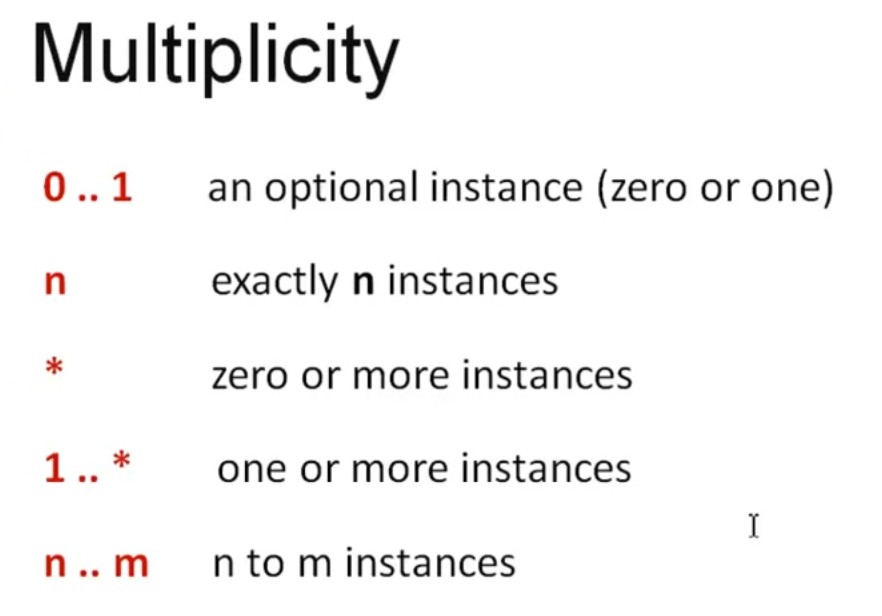
Aggregation And Composition

# Before Class

1. Explain the relationships:
   1. **Association (strzałka)** - Asocjacja to relacja między dwiema oddzielnymi klasami, która ustanawia się poprzez ich Obiekty. Powiązanie może być typu jeden-do-jednego, jeden-do-wielu, wiele-do-jednego, wiele-do-wielu. W programowaniu obiektowym obiekt komunikuje się z innym obiektem, aby korzystać z funkcji i usług dostarczanych przez ten obiekt. Kompozycja i agregacja to dwie formy asocjacji.
   2. **Aggregation (hollow diamond)–** specialna forma associacji, Reprezentuje związek Has-A. Jest to skojarzenie jednokierunkowe, czyli relacja w jedną stronę. Na przykład wydział może mieć studentów, ale odwrotnie nie jest to możliwe, a zatem ma charakter jednokierunkowy. W agregacji oba wpisy mogą przetrwać pojedynczo, co oznacza, że zakończenie jednej jednostki nie wpłynie na drugą.
   3. **Composition (diamond)-** jest ograniczoną formą agregacji, w której dwie jednostki są od siebie wysoce zależne. Reprezentuje część relacji. W składzie oba podmioty są od siebie zależne. Gdy istnieje kompozycja między dwiema jednostkami, złożony obiekt nie może istnieć bez drugiej jednostki.
2. Watch the video:

Association and Multiplicity: <https://youtu.be/BhEoV57nj0Q>

1. What symbols represent relationships of association, aggregation and composition in a UML class diagram? Arrow, hollow diamond, diamond
2. What does the concept of “multiplicity” mean and what information does it provide.



# During Class

1. Working in a group, create Computer and Processor classes in a class diagram. Define class attributes and methods. Create a relationship between the classes assuming that objects of those classes cannot exist independently. If the first object is deleted, the second one have to be also deleted. What kind of relationship should be used? Mark the multiplicity in the class diagram. Save the class diagram to a graphic file.
2. Based on the created class diagram, define the classes in the programming language. Put the classes in a separate project. Then write a program that creates objects, assigns them attribute values, and calls the available methods.
3. Working in a group, create Book and Writer classes in a class diagram. Define class attributes and methods. Consider that the book has one author. Create a relationship between the classes assuming that objects of those classes can exist independently. If one is deleted other can still exist. What kind of relationship should be used? Mark the multiplicity in the class diagram. Save the class diagram to a graphic file.
4. Based on the created class diagram, define the classes in the programming language. Put the classes in a separate project. Then write a program that creates objects, assigns them attribute values, and calls the available methods.

# After Class

1. A files system is a way of storing data on a storage device. The data is organised in computer files and folders. A folder is a container, where many files can be placed. Create a class diagram that shows classes for a folder and file, their attributes and methods, and their relationships. Assume that a folder can contain any number of files and does not contain other folders. Then, define the classes, their attributes and methods. To store any number of files in a folder, you can use a resizable array such as ArrayList. Finally, write a program that:
   1. Create a folder
   2. Add two files to the folder
   3. Display the folder contents
   4. Add another file to the folder
   5. Display the folder contents
   6. Remove the named file from the folder
   7. Display the folder contents
2. Books can be placed on a bookcase. Create a class diagram. Use the previously defined Book and Writer classes (created diagram should contain three classes). Assume that the bookcase can hold any number of books. When creating a class diagram, pay attention to the relationships between classes, taking into account that the book can exist independently of the bookcase. Mark the multiplicity in the class diagram. Then, based on the class diagram, write a program that allows books to be placed in and removed from the bookshelf. Display the bookcase contents.
3. Add a Chapter class to the diagram that you have created, describing the chapter in a book (chapter title, number of pages, starting page number). Assume that a book can contain any number of chapters. What is the relationship between the book and chapters? Can a chapter exist without a book? Then modify the program for operating the bookshelf.
4. Create a class diagram that contains at least four classes, according to your own choice. Add class components and the relationships between the classes. Mark the multiplicity in the class diagram.