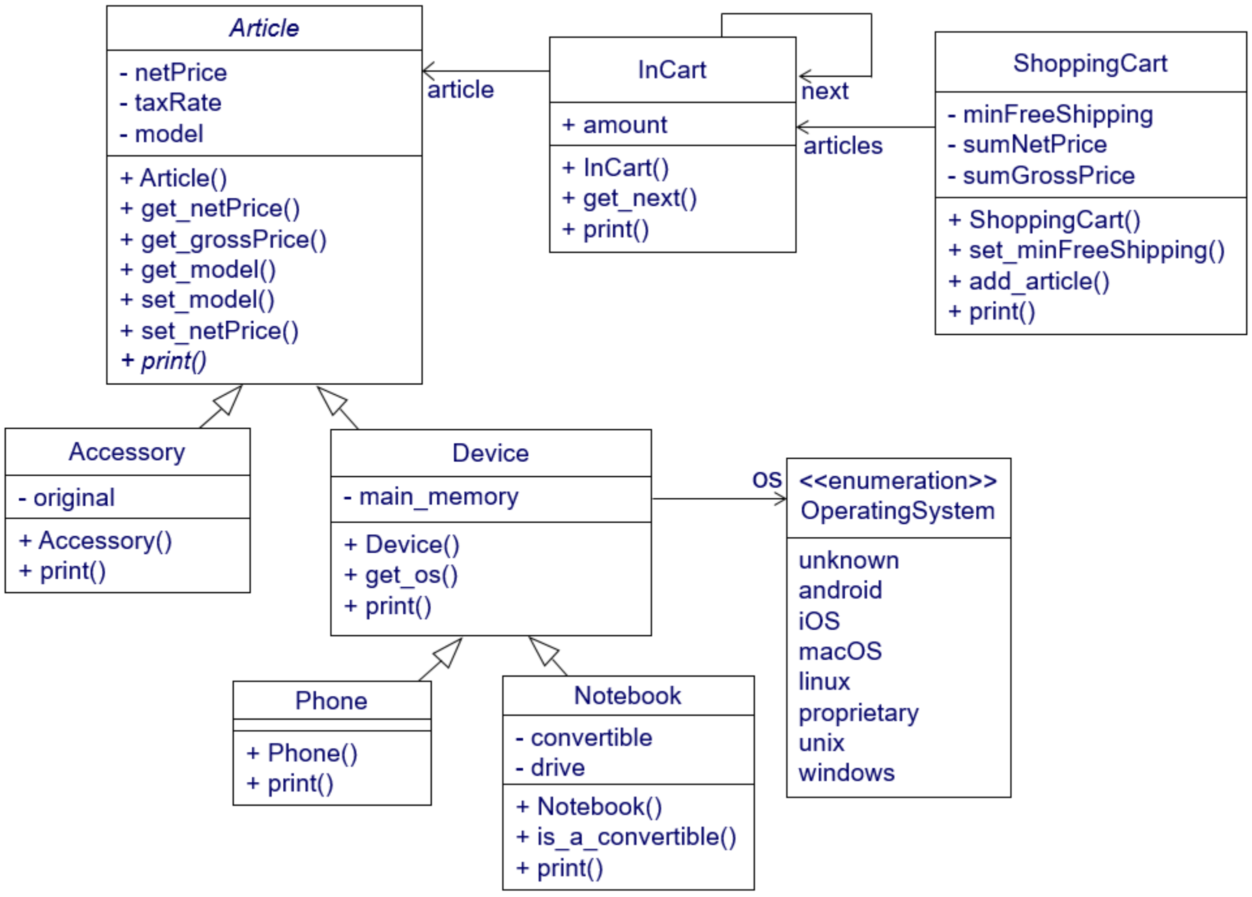
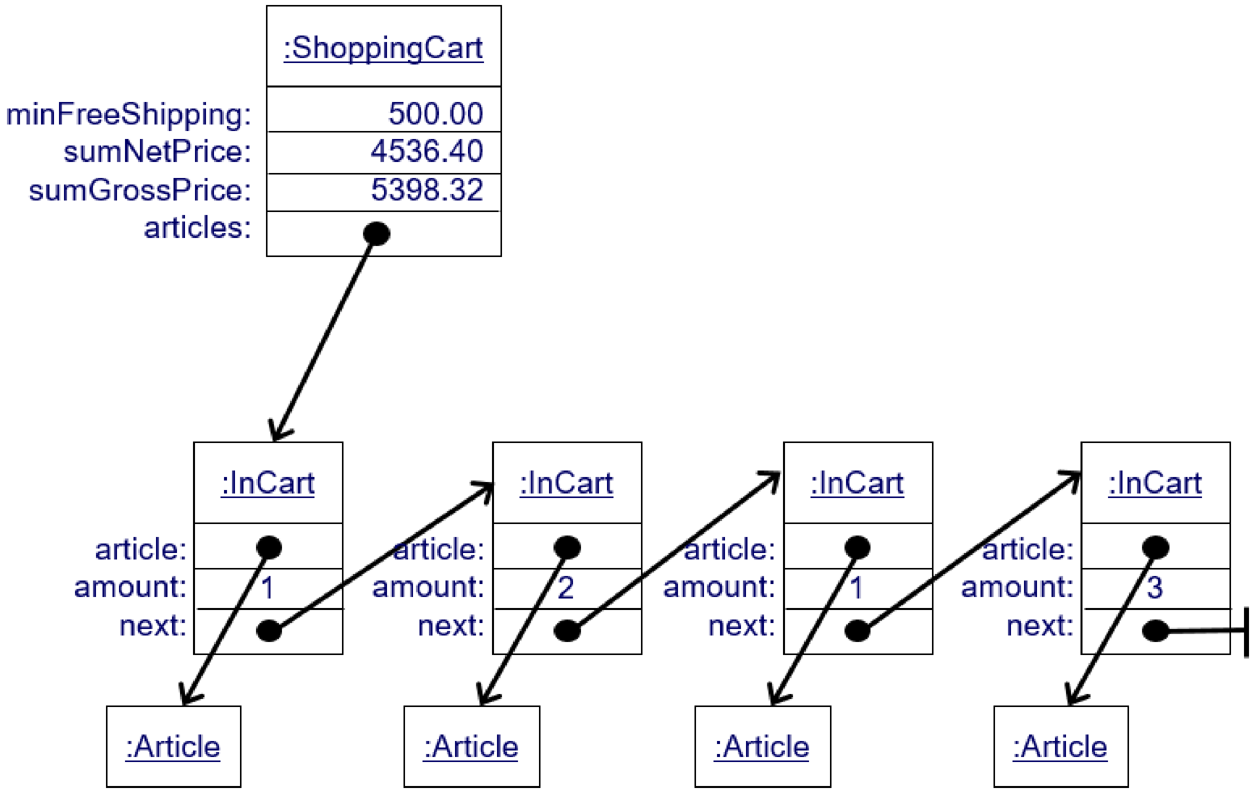
# Proto-Tec-Shop

## Overview

In this task single inheritance with static and dynamic binding shall be trained. For this a simple and extendable electronic shopping cart shall be programmed. As example articles smart phones, notebooks and accessories shall be ordered (due to clarity in the UML class diagram below data types, parameters and destructors are omitted.)



## Example Shopping Cart



**Subtask 1**
Define an abstract base class for an offered article with name ***Article***. This class shall have following members:

- a private string attribute with name **model** storing the article model name.
- two private real valued attributes with names **netprice** for the net price and **taxRate** storing the tax value in percent of the article.
- a public constructor with three parameters to initialise the attributes of an object; the tax rate shall be a default parameter with 19 %.
- a virtual public destructor producing an output **"~Article()"** at its call and the article model name (see examples below).
- a public method with name **set_model** assigning a string parameter to the model name attribute.
- a public method with name **set_netPrice** assigning the double parameter to the net price attribute.
- a public method with name **get_model** without parameters returning the model name as function value.
- a public method with name **get_netPrice** without parameters returning the net price as function value.
- a public method with name **get_grossPrice** without parameters returning the gross price (gross price = net price + tax) as function value.

- a pure virtual public method with name ***print*** without parameter or return value.

**Subtask 2**

Define a class with name **Accessory** derived from abstract class ***Article***. This class shall have following members:

- a private Boolean attribute with name **original** for an original accessory.
- a public constructor with three parameters for the article name, the net price and whether it is an original accessory with **true** as default parameter. The tax rate shall be automatically initialised by the default parameter of class constructor ***Article***.
- a virtual public destructor producing an output **"~Accessory()"** at its call (see examples at the bottom).
- a virtual public method with name **print** without parameters writing the article name and **"(original accessory)"**, if it is an original one (see examples at the bottom).

**Subtask 3**

- Define a C++11 enumeration class with name **OperatingSystem** and enumeration values **unknown**, **android**, **iOS**, **macOS**, **linux**, **proprietary**, **unix** and **windows**.
- Define an overloaded output operator << for a textual output of an operating system name (**"unknown OS"**, **"Android OS"**, **"iOS"**, **"MacOS"**, **"Linux OS"**, **"proprietary OS"**, **"Unix OS"**, **"MS Windows OS"**) onto an output stream (see example at the bottom).

**Subtask 4**

Define a class with name **Device** derived from abstract class ***Article***. This class shall have following members:

- a private integer attribute with name **main_memory** storing the size of the main memory.
- a private attribute with name **os** for the operating system of the device.
- a public constructor with four parameters for the article name, the net price, the main memory size and the operating system with default parameter **unknown**; the tax rate shall be automatically initialised by the default parameter of class constructor ***Article***.

- a virtual public destructor producing an output **"~Device()"** at its call (see examples at the bottom).
- a public method with name **get_os** without parameter returning the operating system.
- a virtual public method with name **print** without parameter or return value outputting the article name, followed by **"RAM"**, the main memory size, **"GB"** as well as the operating system (see examples at the bottom).

**Subtask 5**

Define a class with name **Notebook** derived from class **Device**. This class shall have following members:

- a private string attribute with name **drive**.
- a private Boolean attribute with name **convertible** for a convertible one.
- a public constructor with six parameters for the article name, the net price, the main memory size, the drive, the operating system with default parameter **OperatingSystem::linux** as well as whether it is a convertible with default parameter **false**; the tax rate shall be initialised automatically by the constructor of the upper class.
- a virtual public destructor producing an output **"~Notebook()"** at its call (see examples at the bottom).
- a public method with name **is_a_convertible** without parameters returning a Boolean function value whether the notebook is a convertible.
- a virtual public method with name **print** without parameters, calling method **print** of the direct upper class, in case of a convertible outputs string **"(convertible)"** and afterwards the drive.

**Subtask 6**

Define a class with name **Phone** derived from class **Device**. This class shall have following members:

- a public constructor with four parameters for the article name, the net price, the main memory size with 4GB as Defaultparameter and the operating system Android OS as default parameter; the tax rate shall be initialised automatically by the constructor of the upper class.
- a virtual public destructor producing an output **"~Phone()"** at its call (see examples at the bottom).
- a virtual public method with name **print** without parameters calling method **print** of the direct upper class and afterwards outputting **"phone"**.

**Subtask 7**
Define a class for an article in the shopping cart with name **InCart**. This class shall have following members:

- a private pointer attribute with name **article** to an object of class **Article**.
- a private integer attribute with name **amount** how many of this article are placed in the shopping cart.
- a private attribute with name **next** of type pointer to **InCart** for building up a list.
- a public constructor with three parameters to initialise the three attributes of an object; the amount shall be 0 as default parameter, the pointer to the next article in a shopping cart a null pointer also as default parameter.
- a virtual public destructor outputting **"~Incart()"** and the amount as well as calling the destructor for the article (see examples at the bottom).
- a public method with name **get_next** without parameters returning a pointer to the next article in shopping cart as function value.
- a virtual public method with name **print** without parameters, outputting the amount, then sending message **print** to the article in the shopping cart, then in a new line outputting formatted as in the examples at the bottom the single gross price and the total gross price for the chosen amount of articles onto an output stream (see example at the bottom).

**Subtask 8**

Define a class for a shopping cart with name **ShoppingCart**. This class shall have following members:

- a private pointer attribute with name **articles** to the head of a list of articles in the shopping cart of class **InCart**.
- three private real valued attributes with names **minFressShipping** for which purchase value free shipping is offered, **sumNetPrice** for the total net price sum of all articles in the shopping cart and **sumGrossPrice** for the total gross price sum.
- a public standard constructor initialising an empty shopping cart with both sum values 0 EUR and the purchase value for free shipping with1000 EUR.
- a public destructor outputting **"~ShoppingCart()"** and afterwards deleting each article in the shopping cart starting with an output **"delete article: "** such that no memory leak will exist.
- a public method with name **set_minFreeShipping** with one parameter assigning its value to the same named attribute.

- a public method with name **add_article** with an amount and a pointer to an article as parameters, adding a new object **InCart** on heap to the shopping cart and actualising the two sum variables of the class appropriately.
- a virtual public method with name **print** without parameters writing a shopping onto the standard output stream cart formatted as given in the examples at the bottom. Especially the net price sum, the tax (difference gross minus net total sum) and the gross price sum shall be outputted at the end of the shopping cart. Take care to also regard the (may be) shipping cost output of 5.90 EUR (see example at the bottom)!

**Subtask 9**

To test write a **main** function placing some articles as new objects on the heap into an electronic shopping cart:

- define a shopping cart object of type **ShoppingCart** using the standard constructor.
- Send a message to this object that the articles get free shipping starting from 500 EUR.
- Add to this shopping cart a new accessory article on heap with an amount of three powerbanks **"YOOLOX 10k Wireless Powerbank Qi"** with single net price 31.08 EUR being no original accessory.
- Print the complete shopping cart (there are shipping costs).
- Add to this shopping cart a new smart phone **"Samsung Galaxy S10+ SM-G975F/DS Dual SIM"** with net price 661.67 EUR, 8GB main memory size and Android OS operating system.
- Add to this shopping cart two new smart phones **"Apple iPhone 11 Pro 256GB"** with net price 1097.47 EUR, 4GB main memory size and iOS operating system.
- Add to this shopping cart a new notebook **"ASUS ROG Strix Scar III G731"** with net price 1586.55 EUR, 16GB main memory size, **"512GB SSD + 1TB SSHD"** as drive and MS Windows operating system.
- Print again the complete shopping cart (now there are no shipping costs).

With the end offunction **main** automatically the complete shopping cart object gets deleted.