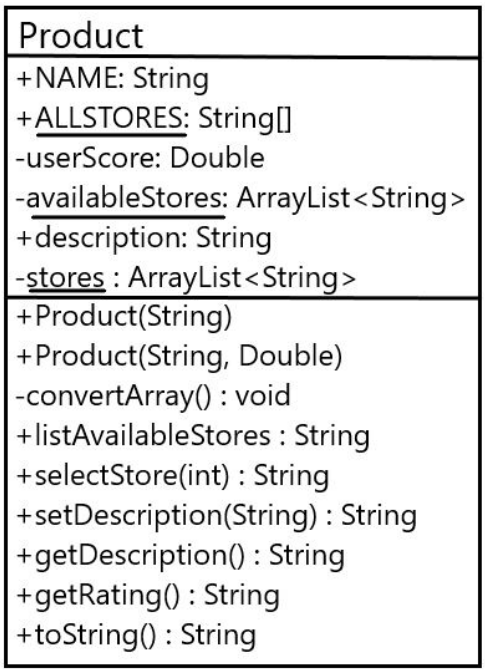
Lesson 3a

Mark Eric Dieckmann

February 2, 2023

Lesson 3a deals with single inheritance. We implement a framework that allows users to define products that are sold in several stores across Sweden. We exemplify the use for one project, TV sets. Our hierarchical scheme splits the methods and variables into two groups and places them into separate classes. The class **Product** handles the logistics of a product and is therefore not limited to TV sets, and another class (**TVSet**) addresses a specific product. **Product** is the superclass and **TVSet** the subclass. **Lesson3a** is fully implemented and tests our code.

1 Task 1: The class Product



The constant *NAME* takes the name of the product and is initialized in the constructor.

The class constant *ALLSTORES* is a list of all stores that sell products. Initialize it with Norrköping, Linköping, Stockholm, and Malmö when you declare it.

Give the variable *userScore* the value *null* when you declare it.

The class variable *availableStores* is a list of all stores that are not yet selling this product. Do not initialize it yet.

description will be filled with information about the product.

The class variable *stores* is a list of all stores that are selling this product. You initialize it when you declare it.

The constructor with one argument initializes *NAME* and calls *convertArray()*. The constructor with two arguments also initializes *userScore*.

convertArray() initializes *availableStores* and copies all elements of *ALLSTORES* into *availableStores*.

listAvailableStores() starts with the line "Number, City" followed by a line break. It adds all available stores (those that do not yet distribute the product) together with their list index and uses one line for each. It returns the concatenated string.

selectStore(arg) should pick the store with the number *arg* from the list provided by *listAvailableStores()*. It should return "Wrong number" if *arg* exceeds the maximum number of stores on the list. If there is an available store with that number, it should copy that store from *availableStores* into *stores*, and delete it in *availableStores*.

setDescription(arg) sets the value of *description* to *arg* and returns "Product description done".

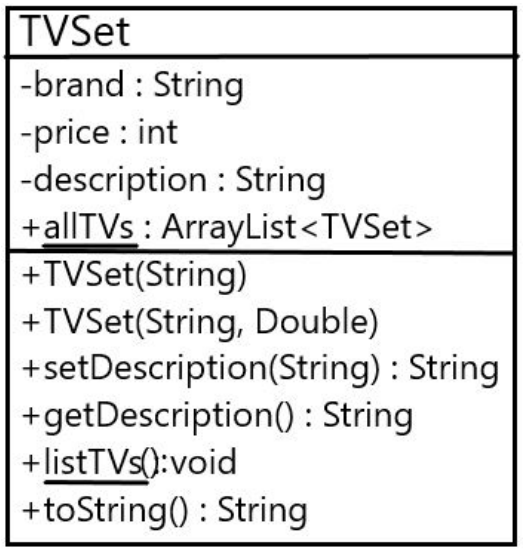
getDescription() returns the value in *description*.

getRating() returns the value of *userScore* if there is one or it returns "Not rated yet" if the value is still *null*.

toString() returns a string that starts with "This product " followed by "has the user score " followed by the value of *userScore* (if it has one) followed by " and is available in the shops in: " followed by the shops listed in *stores*. The stores should be separated by a "," and the last one should be preceeded by an "and" and followed by a ".".

2 Task 2: The class TVSet

This is a subclass of **Product** and contains information about a specific TV brand.



brand denotes the brand of the TV set and is initialized with the argument passed to the constructors.

price is the price of the TV set and is initialized in the constructors.

description describes the TV.

allTVs is a class variable that stores the list of all TV sets sold by the company.

Both constructors initialize *NAME* of **Product** with "tv set" and initialize *brand* with the string in the argument list. Both constructors initialize *price* with a random number between 10001 and 15000 and add this object to *allTVs*.

setDescription(arg) splits *arg* in two at the ",", trims both strings, attaches the second string to *description*, and initializes *description* in the superclass with the first string. It returns the return value of the *setDescription(..)* method of the superclass. *getDescription()* returns the string "TV description: " followed by the value of *description* followed by a line break followed by "Product information: " followed by the value of *description* in the superclass.

listTVs() writes the information of all TV sets in *allTVs* to the console. It should write the value of *brand* followed by " " followed by the price of the TV set followed by " " followed by the rating of the TV set. It should use one line per TV.

toString() should return the line "The " + the value of *NAME* followed by " is manufactured by " followed by *brand* followed by " and costs " followed by *price* followed by "SEK." and a line break. It should append to it the return value of **Product**'s *toString()* method and return the result.

Running **Lesson3a** should give you the console output:

```
Number of TVs: 0
Number of TVs: 8
Brand1 13212 Not rated yet
Brand2 12098 Rating: 4.0
Brand3 14601 Rating: 1.0
Brand4 10830 Rating: 2.5
Brand5 12476 Rating: 4.5
Brand6 10382 Rating: 1.5
Brand7 12262 Not rated yet
Brand8 11739 Rating: 4.5

TV description: Its brand is brand1
Product information: The product is a TV

Number, city
0 Norrköping
1 Linköping
2 Stockholm
3 Malmö

Norrköping added
Number, city
0 Linköping
1 Stockholm
2 Malmö

Malmö added

Number, city
0 Linköping
1 Stockholm

The tv set is manufactured by Brand1 and costs 13212SEK.
This product  and is available in the shops in: Norrköping,  and Malmö.
```