Programming Technologies, Sample Test 1, 2017.

The following class is given:

```
public class Product
{
    protected int id;
    protected String name;
    protected double price;
    protected LocalDate production;
    protected bestBefore;
}
```

1. Add a constructor to the class which can set all fields of the class. (1p)

2. Add getter and setter methods for all fields. (1p)

3. Make sure that the id of a product `Product` cannot be a negative number. Use a custom exception for this in the needed places. Name this exception `InvalidIdException`) (2p)

4. Override the **`toString()`** method of the Product class in a way that it returns all fields of the product. (The format is not specified.) (1p)

5. Override the **`equals()`** method of the class in a way that two products should be equal only if ids are equal! (Do not compare the other fields.) (1p)

6. Alter the **`Product class`** so that its instances can be compared based on their price. (2p)

7. Derive the **`DiscountProduct class`** from the **`Product class`**. This class should have an additional **`int`** property called `discountAmount` besides the properties of simple Products (This attribute tells the amount of discount in percentages. 0.5 means 50%, 0.3 means 30% etc..). Add a constructor to the class that lets the user to set all three fields to an initial value. (1p)

8. Override the **`toString()`** method of the class so that all fields of discounted products should be returned in a row. The first three fields are formatted the similar way as in the super class but after that the String "SALE!!!" is added plus the amount of discount in percentage format. (1p)

10. Add a method to the **`DiscountProduct class`** that returns the new price of the product i.e. the original price multiplied by the amount of discount. (e.g. the above Shoe has a new price of 4.15.-) (2p)

11. Add an assert to this method that terminates the program, if the discount amount is greater than 1 (100%). Do not forget to enable asserts in your project. (2p)

12. Create a **`Store class`**, which can contain an arbitrary number of Products. There should be a possibility of adding new Products, and removing existing ones. (2p)

13. Instantiate a new **`Store`** in the main method (1p) and read Products from a file to it. The contents of the file are written below. If the bestBefore date is to be reached in 3 days, instead of creating an ordinary product instantiate a DiscountProduct. The amount of discount is 30%.(3p)

The **input file** contents are: :

```
bread,2.50,2017-03-26,3
butter,2.30,2017-03-20,14
milk,1.80,2017-02-26,30
cheese,3.10,2016-12-25,110
sugar,2.70,2016-12-10,300
flour,1.50,2016-11-15,180
```

14. Extend the **`Store class`** with a method that writes those products on the screen alphabetically ordered that are cheaper than 10.-. Call this method from the main class. (2p with stream 1p without stream).

15. Extend the **`Store class`** with a method that writes the names of all products on the screen with capital letters in ascending order by original price. Call this method from the main class. (2p with stream 1p without stream).

16. Add a marker annotation type to your project. The name of it should be “MarkerField”. This annotation can be used on fields and Methods and it is visible till runtime. It also appears in the documentation of the classes. Mark the name field of the Product class with this annotation. (2p)

17. Add a static method to the Main class that prints out how many days have passed since your birthday. (You do not have to use your real birthday if you would not like). Call this method from the main method. (2p)