Exercises for Java Advanced Features: Coding

1. Create 3 classes: Person, Developer, JavaDeveloper.
    a. Person is a parent class, Developer inherits from Person, JavaDeveloper inherits from Developer
    b. Create constructor for every class that will call constructor of the super class. Each constructor should display an information, that it has been called.
    c. Create an object of type JavaDeveloper. What information will be displayed and in what order?
    d. Using an object of type JavaDeveloper call a method that is defined in Developer class. What access modifier should it have?
    e. *Overload method from the Person class in JavaDeveloper class to accept additional parameters.
2. Warehouse
    a. User should be able to: add, display all of the details, update, delete an item
    b. Use composition and collections (The warehouse has products/items)
    c. Add possibility to display summaries, like sum of all of the products, their prices.
    d. *Add possibility to update number of items in a specific way, e.g.:
    “pliers:30”
    “scissors:+4”
3. Personal information
    a. Create a file containing any personal data (name, surname, phone number). Data of individual persons should be in the following lines.
    b. Download data from a file and create objects of people based on them (in any way - Regex, String.split ...).
    c. Enter the created objects into ArrayList or Map (<line number>: <Person>).
    d. Present the obtained data.
4. Let’s buy a vehicle
    a. Create class Person
    b. Create class Parser
    c. Create interface or an abstract class Vehicle
    d. Create classes Car and Bike, that will implement/inherit Vehicle.
    e. User will provide all of the details about the Person (buyer) using command-line (e.g. “John Smith born 3/24/1984”. Provided information will be parsed using Regex within Parser class. Parser class should receive char sequence and return an object of type Person or Null if provided details will not have required information.
    f. Created person will then buy a bike and car. Information about what and when was bought should be displayed.
    g. Brand and model of car/bike should be stored using variable of type Enum.
5. Create a List that stores the character, e.g. "*".
    a. Use the list to draw a horizontal line.
    b. Draw a vertical line
    c. Draw a square full of asterisks.
    d. * Inside the loop, allow the user to select "add" / "delete" "row" / "column" - display the effect after each selection.
6. Create a Map where the key will be the employee and the value - his manager.
    a. The key and value are of the String type
    b. The key and value are classes of type "Employee" and "Manager"

    c. * The key is of type "Manager", the value is a list storing the type "Employee"
    d. * Let the employee be dismissed, display the result
    e. * Allow to employ a new employee, display the result
7. We're planning vacations
    a. User provides information about the country and cities that he is going to visit. You can use a nested while loop to gather information and a HashMap to store it.
    b. Display the created plan.
        i. *if city occurs on the list twice it should be displayed as "back and forth"
    c. Store created plan using JSON.
8. We're going on vacation
    a. Load created in the previous exercise plan (JSON). Display it.
    b. Create a simple interaction with a user. Every time, when user will press Enter (or write "next") next city (or country) should be displayed.
    c. Add a possibility to remove a city/country from the list (user was not able to visit it)
9. * Statistics
    a. Create a file that will contain any text. It can be generated using lorem ipsum library.
    b. Create statistics of words contained in the text.
    c. Display the number of occurrences of individual words in the form <word>: <number of occurrences>
    d. * as above sorted
10. *Create a factory that includes Manager, employees (Worker) and *Director.
    a. loops, conditions - adding employees, displaying currently working (what they do), getting orders from the user.
    b. OOP - all employees (including the director), inherit from the common class (Employee). Each employee may have a work tool (e.g. Hammer, Laptop, PointingFinger).
    c. collections - appropriate grouping of data
    d. * threads - use the while loop only to receive orders from the user, use the threads to display status / modify each object.
    e. ** additional functionalities – e.g. displaying information about who and how much has earned since the beginning of work (counting every few seconds), "deduct in succession" - sorted list of employees (by name), employee data can be loaded from the file at the beginning.
    f. ** support for the Director class - displaying information "everybody works - great!", "where is <name> ?!" (e.g. if the employee has left earlier).