# Exercises: Objects and Classes

Problems for exercises and homework for the [“Programming Fundamentals” course @ SoftUni](https://softuni.bg/courses/programming-fundamentals).

You can check your solutions here: <https://judge.softuni.bg/Contests/210/Objects-and-Classes-Exercises>.

## Count Working Days

Write a program that **reads two dates** in formatdd-MM-yyyy and prints the **number of working days** between these two dates **inclusive**. Non-working days are:

* All days that are **Saturday** or **Sunday**.
* All days that are **official holidays** in Bulgaria:
  + New Year Eve (**1 Jan**)
  + Liberation Day (**3 March**)
  + Worker’s day (**1 May**)
  + Saint George’s Day (**6 May**)
  + Saints Cyril and Methodius Day (**24 May**)
  + Unification Day (**6 Sept**)
  + Independence Day (**22 Sept**)
  + National Awakening Day (**1 Nov**)
  + Christmas (**24**, **25** and **26 Dec**)

All days not mentioned above are **working** and should count.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 11-04-2016  14-04-2016 | 4 |
| 11-04-2016  22-04-2016 | 10 |
| 20-12-2015  31-12-2015 | 7 |

### Hints

* Read **start date** and **end date** from Console.
* **Create** two objects of type DateTime – startDate and endDate.
* Create an **array of type** DateTime and add **all official holidays** in it.
* Loop from startDate to endDate. Add **1 day** at each iteration.
* Get the **current da**y in the loop and check whether is **Saturday**, **Sunday** or it is **contained** **in the holidays array**. If it is not, increment the workingDaysCounter.

## Advertisement Message

Write a program that **generate random fake advertisement message** to extol some product. The messages must consist of 4 parts: laudatory **phrase** + **event** + **author** + **city**. Use the following predefined parts:

* **Phrases** – {“Excellent product.”, “Such a great product.”, “I always use that product.”, “Best product of its category.”, “Exceptional product.”, “I can’t live without this product.”}
* **Events** – {“Now I feel good.”, “I have succeeded with this product.”, “Makes miracles. I am happy of the results!”, “I cannot believe but now I feel awesome.”, ”Try it yourself, I am very satisfied.”, “I feel great!”}
* **Author** – {“Diana”, “Petya”, “Stella”, “Elena”, “Katya”, “Iva”, “Annie”, “Eva”}
* **Cities** – {“Burgas”, “Sofia”, “Plovdiv”, “Varna”, “Ruse”}

The format of the output message is: **{phrase} {event} {author} – {city}**.

As an input you take the **number of messages** to be generated. Print each random message at a separate line.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 3 | Such a great product. Now I feel good. Elena – Ruse  Excelent product. Makes miracles. I am happy of the results! Katya – Varna  Best product of its category. That makes miracles. Eva - Sofia |

### Hints

* Hold the phrases, events, authors and towns in 4 arrays of strings.
* Create Random object and **generate** **4 random numbers** each in its range:
  + phraseIndex 🡪 ­[0, phrases.Length)
  + eventIndex 🡪 [0, events.Length)
  + authorIndex 🡪 [0, authors.Length)
  + townIndex 🡪 [0, towns.Length)
* Get one **random element** from each of the four arrays and **compose a message** in the required format.

## Intersection of Circles

Create class Circle with properties Center and Radius. The center is a point with coordinates X and Y (make a class Point). Write a method bool Intersect(Circle c1, Circle c2) that tells whether the two given circles **intersect or not**. Write a program that tells if two circles intersect.

The input lines will be in format: **{X} {Y} {Radius}**. Print as output “Yes” or “No”.

### Examples

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Visualization** |
| 4 4 2  8 8 1 | No |  |
| 3 3 2  4 3 6 | Yes |  |
| 1 1 4  4 2 5 | Yes |  |

### Hints

* Calculate d = **distance between the circle centers**.
* If the d ≤ r1 + r2 (the sum of radiuses**) 🡪** the circles **intersect** (or one of the circles is inside the other or the circles have one common point when d = r1 + r2).
* If the d > r1 + r2 **🡪** the circles do **not intersect** (they have not common shared point).

## Average Grades

Define a class Student, which holds the following information about students: **name**, **list of grades** and **average grade** (calculated property, read-only). A single grade will be in range [2…6], e.g. 3.25 or 5.50.

Read a **list of students** and print the students that have **average grade ≥ 5.00** ordered **by name** (ascending), then by **average** **grade** (descending). Print the student name and the calculated average grade.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 3  Ivan 3  Todor 5 5 6  Diana 6 5.50 | Diana -> 5.75  Todor -> 5.33 |
| 6  Petar 3 5 4 3 2 5 6 2 6  Mitko 6 6 5 6 5 6  Gosho 6 6 6 6 6 6  Ani 6 5 6 5 6 5 6 5  Iva 4 5 4 3 4 5 2 2 4  Ani 5.50 5.25 6.00 | Ani -> 5.58  Ani -> 5.50  Gosho -> 6.00  Mitko -> 5.67 |

### Hints

* Create class Student with properties Name (string), Grades (double[]), and property AverageGrade (calculated by LINQ as Grades.Average(), read-only).
* Make a **list of students** and **filter with LINQ** all students that has average grade >= 5.00.
* Print the filtered students **ordered by name** in ascending order, then by **average grade** in descending order.

## Book Library

To model a **book library**, define classes to hold a **book** and a **library**. The library must have a **name** and a **list of books**. The books must contain the **title**, **author**, **publisher**, **release date**, **ISBN-number** and **price.**

Read a **list of books**, add them to the library and print the **total sum of prices by author**,ordered **descending by price** and **then by author’s name lexicographically**.

Books in the input will be in format **{title} {author} {publisher} {release date} {ISBN} {price}**.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 5  LOTR Tolkien GeorgeAllen 29.07.1954 0395082999 30.00  Hobbit Tolkien GeorgeAll 21.09.1937 0395082888 10.25  HP1 JKRowling Bloomsbury 26.06.1997 0395082777 15.50  HP7 JKRowling Bloomsbury 21.07.2007 0395082666 20.00  AC OBowden PenguinBooks 20.11.2009 0395082555 14.00 | Tolkien -> 40.25  JKRowling -> 35.50  OBowden -> 14.00 |

### Hints

* Create classes Book and Library with all the mentioned above properties:****
* **Create** an object of type Library.
* **Read the input** and create a Bookobject for each book in the input.
* Create a **LINQ** query that will **sum the prices by author** and print the result.

## Book Library Modification

Use the classes from the previous problem and make a program that **read a list of books** and **print all titles** **released after given date** ordered **by date** and then **by** **title lexicographically**. The date is given if format “**day-month-year**” at the last line in the input.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 5  LOTR Tolkien GeorgeAllen 29.07.1954 0395082999 30.00  Hobbit Tolkien GeorgeAll 21.09.1937 0395082888 10.25  HP1 JKRowling Bloomsbury 26.06.1997 0395082777 15.50  HP7 JKRowling Bloomsbury 21.07.2007 0395082666 20.00  AC OBowden PenguinBooks 20.11.2009 0395082555 14.00  30.07.1954 | HP1 -> 26.06.1997  HP7 -> 21.07.2007  AC -> 20.11.2009 |