# 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 format **dd-MM-yyyy** and prints the **number of working days** between these two dates **inclusive**. Consider that **official holidays** are 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, 26 Dec**). Consider also **leap years** have 1 extra day.

### 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 holidays**
* Loop from **startDate** to **endDate** while adding **1 day** to each iteration
* Get **current da**y in the loop and check whether is **Saturday**, **Sunday** or it is **contained** **in the array** of holidays. If it is not increase the **workingDaysCounter** with 1

## Advertisement Message

Write a program that generate random advertisement message for some product. The messages must contain 4 parts: Laudatory phrase, event, author, author’s city

Use the predefined arrays with various samples for all 4 parts.

* **Phrases** – {“Excellent product”, “Such a great product”, “I always use that product”, “Best product of its category”}
* **Events** – {“Now I feel good.”, “I have succeeded to change.”, “That makes miracles.”, “I cannot believe but now I feel awesome”, ”Try it yourself, I am very satisfied”.}
* **Author** – {“Diana”, “Petya”, “Stella”, “Elena”, “Katya”, “Iva”, “Annie”, “Misha”}
* **Cities** – {“Burgas”, “Sofia”, “Plovdiv”, “Varna”, “Ruse”}

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

As an Input you receive the **number of messages** you must generate and print. **Each on new line**.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 2 | Such a great product. Now I feel good. Elena – Ruse  Excelent product. That makes miracles. Katya - Varna |

### Hints

* Create **Random** object and **generate** **4 random numbers** each in range
  + [0, phrases.Length)
  + [0, events.Length)
  + [0, authors.Length)
  + [0, cities.Length)
* Get **1 element from each of the four arrays** and **compose a message** in the required format

## Circles Intersection

Create class **Circle** with properties **Center** and **Radius**. (Center is a **point** with coordinates **X** and **Y**). 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}.**

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 4 4 2  8 8 1 | No |
| 3 3 2  4 3 6 | Yes |
| 1 1 4  2 2 7 | Yes |

### Hints

* if **distance between centers** is bigger than the **sum of circle’s radiuses** circles do **not intersect**.

## Average Grades

Define a class **Student**, which contains the following information about students: **name**, **list of grades** and **average grade**. A single grade will be in range [2-6]

Read **list of students** and print all the students that have **average grade ≥ 5.00** ordered **by name**

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 5  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  Gosho -> 6.00  Mitko -> 5.67 |

### Hints

* Create class **Student** with properties **Name**, **Grades** and method **AverageGrade**
* **AverageGrade** might be calculated using LINQ – **Grades.Average()**
* Make a **list of students** and **filter** all students that has average grade >= 5.00.
* Print those students **ordered by name** in ascending order

## Book Library

There is a **book library**. Define classes respectively for a **book** and a **library**. The library must contain a **name** and a **list of books**. The books must contain the **title, author, publisher, release date, ISBN-number and price.**

Read **list of books**, add them to the library and print the **total prices by author** ordered **descending by price**.

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

### Examples

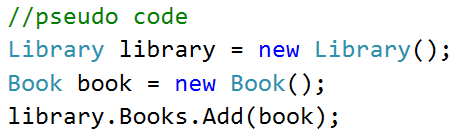
|  |  |
| --- | --- |
| **Input** | **Output** |
| 5  LOTR Tolkien GeorgeAllen 29.07.1954 0395082999 30.00  Hobbit Tolkien GeorgeAll 21.09.1937 0395082888 10.25  HP1 JKRowling Bloomsbury 26.06.1997 0395082777 15.50  HP7 JKRowling Bloomsbury 21.07.2007 0395082666 20.00  AC OBowden PenguinBooks 20.11.2009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

### Hints

* Create classes **Book** and **Library** with all the required properties
* **Create** object of type **Library**



* **Read input** and **create** new **Book objec**t **with the data from the input**
* **Add** the newly created **book to the library**



* Create a **LINQ** query that will **sum prices by author** and print the result

## Book Library Modification

Use the classes from the previous task and make a program that **read a list of books** and **print all titles** that are **released after given date** ordered **by date**

### 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 |