73.3
Pass

Leonardo Felicio de Melo

Back-End Developer (.NET)

PDF generated at: 2 Oct 2024 23:25:30 UTC

View this report on HackerRank

Score

73.3% • 110 / 150

scored in Workana Back-End Developer (.NET) in 147 min on 2 Oct 2024 14:54:40 MDT

Candidate Information

Email	leonardo.melo.dev@hotmail.com
Test	Workana Back-End Developer (.NET)
Candidate Packet	View
Taken on	2 Oct 2024 14:54:40 MDT
Time taken	147 min/ 150 min
Work Experience	3 years
Invited by	Lucas

Suspicious Activity detected

Code similarity

Code similarity
1 question

Skill Distribution

No.	Skill	Score

1	SQL Basic	100%
2	.NET Basic	20%
3	C# Basic	100%

Tags Distribution

Database	100%	SQL	100%
Easy	73%	PostgreSQL	100%
Simple Queries	100%	Relationships	100%
Aggregation	100%	.NET	20%
Back-End Development	20%	.NET API	20%
.NET MVC	20%	Entity Framework	20%
C#	100%	Inheritance	100%

Questions

Status	No.	Question	Time Taken	Skill	Score

✓	1	SQL: Fitness Tracker Activity Report DbRank	5 min 57 sec	SQL (Basic)	50/50
✓	2	.NET Calendar API Back-end Developer	1 hour 58 min 37 sec	.NET (Basic)	10/50
✓	3	C#: Computer Inheritance Coding	7 min 45 sec	C# (Basic)	50/50 ⚑

1. SQL: Fitness Tracker Activity Report

✓ Correct

DbRank Database SQL Easy PostgreSQL Simple Queries Relationships Aggregation

Question description

Create a query for a fitness tracker. It should return a list of all the activities that have occurred in the current month and a summary of the segments of those activities.

Each activity contains a set of segments that are used to calculate activity metrics such as steps taken and calories burned.

The result should have the following columns: *name | dt | segments | average_segment_steps | total_steps | total_calories*.

- *name* - activity name
- *dt* - activity date and time
- *segments* - total number of activity segments
- *average_segment_steps* - average number of activity steps, rounded up to the nearest integer.
- *total_steps* - total number of activity steps
- *total_calories* - total calories burned during activity

The result should be sorted in ascending order by *dt*.

Note:

- Only activities in the current month should be included in the result.
- The current month is September.

▼ SCHEMA

activities			
name	**type**	**constraint**	**description**
id	INT	PRIMARY KEY	Activity ID
name	VARCHAR(255)		Activity name
dt	DATETIME		Activity date and time

segments			
name	**type**	**constraint**	**description**
activity_id	INT	FOREIGN KEY (activity_id => activities.id)	Activity ID
steps	SMALLINT		Segment steps
calories	SMALLINT		Segment calories

▼ SAMPLE DATA TABLES

activities

id	name	dt
1	Running	2022-08-28 00:24:13
2	Hiking	2022-09-14 06:15:50
3	Walking	2022-09-01 15:47:08

segments		
activity_id	steps	calories
1	1308	115
1	1931	98
1	522	112
1	1460	64
1	1598	58
1	1031	63
1	1480	22
1	2243	107
2	1230	35
2	733	25
2	2108	92
2	1831	54
2	1651	79

2	757	66
2	634	94
3	1184	111
3	1968	74
3	1048	104
3	1203	119
3	1441	58

▼ EXPECTED OUTPUT

name	dt	segments	average_segment_steps	steps	calories
Walking	2022-09-01 15:47:08	5	1369	6844	466
Hiking	2022-09-14 06:15:50	7	1278	8944	445

Interviewer guidelines

```
SELECT
  name,
  dt,
  COUNT( * )          AS segments,
  CEIL( AVG( steps ) ) AS average_segment_steps,
  SUM( steps )         AS steps,
  SUM( calories )      AS calories
  FROM
    activities a
      LEFT JOIN segments s
        ON a.id = s.activity_id
  WHERE
    MONTHNAME( dt ) = 'September'
```

```
GROUP BY
  id
ORDER BY
  dt
```

Candidate's Solution

Language used: **MySQL**

```
/*
Enter your query below.
Please append a semicolon ";" at the end of the query
*/

SELECT
    a.name,
    a.dt,
    COUNT(s.activity_id) AS segments,
    CEIL(AVG(s.steps)) AS average_segment_steps,
    SUM(s.steps) AS total_steps,
    SUM(s.calories) AS total_calories
FROM
    activities a
JOIN
    segments s ON a.id = s.activity_id
WHERE
    MONTH(a.dt) = 9
    AND YEAR(a.dt) = 2022
GROUP BY
    a.id
ORDER BY
    a.dt;
```

Time taken: **0.03 sec**

ⓘ **No comments.**

2. .NET Calendar API

✓ Partially correct

Back-end Developer · .NET · Easy · Back-End Development · .NET API · .NET MVC · Entity Framework

Question description

A company is launching a new service that provides scheduling appointments and meetings of an individual, similar to popular services such as Gmail Calendar, Outlook, etc. A 'calendar' needs to be created, and as part of this challenge, you are required to come up with a service to maintain this calendar.

As step 1, create a service that supports REST APIs for creating, deleting, and updating events in a calendar. An event will have details such as event name, scheduled time, scheduled location, members, etc.. A few more APIs required would be to fetch the event details, the events in a particular location, sort the events as per the time, finding all events for a particular organizer, etc. A detailed explanation about the APIs and data is given below.

Each event object is a JSON object with the following keys -

1. *name* - Name of the event. [STRING]
2. *time* - Scheduled time for the event in UTC (GMT + 0). [EPOCH INTEGER]
3. *location* - Location of the event. [STRING]
4. *members* - String of member names separated by a comma. [ARRAY OF STRINGS]
5. *eventOrganizer* - Name of the organizer of the event. [STRING]
6. *id* - Unique ID of the event as generated by the system. [INTEGER]

▼ EXAMPLE

```
{
  "name": "Agenda discussion",
  "time": 1573843210,
  "location": “Miami”,
  "members": “Any,Jay”
  “eventOrganizer”: “Sam”,
  "id": "1"
}
```

▼ APIS

The following APIs need to be implemented:

1. *Adding a new event* - POST request should be created to add a new event. The API endpoint would be */calendar*. The request body contains the details of the event. HTTP response should be 201.
2. *Deleting any event by id* - DELETE request to endpoint */calendar/{id}* should delete the event. If the item does not exist return not found.
3. *Editing the event* - PUT request to endpoint */calendar/{id}*. The request body would contain the id of the event and the information that needs to be edited. If the item does not exist return not found.
4. *Getting all events* - GET request to endpoint */calendar* should return all the events in the system. The HTTP response code should be 200. If no event exists, return the empty array.
5. *Getting all events of the organizer* - GET request to endpoint */calendar/query?eventOrganizer={eventOrganizer}* should return the entire list of events organized by this organizer. The HTTP response code should be 200. For empty response return empty array.
6. *Getting event by id* - GET request to endpoint */calendar/query?id={id}* should return the details of the event with this unique id. The HTTP response code should be 200.
7. *Getting all events by location* - GET request to endpoint */calendar/query?location={location}* should return the entire list of events happening at that location. The HTTP response code should be 200.
8. *Getting event by name* - GET request to endpoint */calendar/query?name={name}*should return the details of the event with this name. The HTTP response code should be 200.
9. *Sort the event as per the time* - GET request to endpoint */calendar/sort* should return the events sorted in descending order of time.

Candidate's Submission

Testcase	Test file	Status	Score

TestCalendarCheckNonExistentApi	TestResults.xml	⊗ Failed	0 / 5.0
TestCalendarUpdate_Ok	TestResults.xml	⊗ Failed	0 / 5.0
TestCreateCalendar_Ok	TestResults.xml	✓ Success	5.0 / 5.0
TestDeleteCalendar_Ok	TestResults.xml	⊗ Failed	0 / 5.0
TestGelCalendarsByLocation_Ok	TestResults.xml	⊗ Failed	0 / 5.0
TestGetCalendar_Ok	TestResults.xml	⊗ Failed	0 / 5.0
TestGetCalendarsByEventOrganizer_Ok	TestResults.xml	⊗ Failed	0 / 5.0
TestGetCalendarsByName_Ok	TestResults.xml	⊗ Failed	0 / 5.0
TestGetCalendars_Ok	TestResults.xml	✓ Success	5.0 / 5.0
TestGetSortedCalendar	TestResults.xml	⊗ Failed	0 / 5.0

Review logs: output log
View candidate code

No comments.

3. C#: Computer Inheritance

Correct

Coding C# Easy Inheritance

Question description

Implement inheritance as described below.

Create an abstract class Computer that has the following:

1. A member variable *type* [string]
2. A member variable *model* [string]
3. A member variable *cpu* [string]
4. A member variable *isTurnedOn* [boolean] (the default status value is false)
5. A constructor function that takes 3 parameters and assigns them to *type*, *model*, and *cpu* respectively
6. A member function *GetComputerType()* that returns the value *type*
7. A member function *GetComputerModel()* that returns the value *model*
8. A member function *GetComputerCpu()* that returns the value *cpu*
9. A member function *GetComputerStatus()* that returns the value *isTurnedOn*
10. A member function *SwitchComputerStatus()* that toggles the *isTurnedOn* value

Create a class PersonalComputer that inherits from the above class Computer. It has the following:

1. A constructor function that takes 2 parameters, *model* and *cpu*. It calls the base class constructor with *type* value 'PersonalComputer', *model*, and *cpu* respectively.

Create a class Notebook that inherits from the above class Computer. It has the following:

1. A constructor function that takes 2 parameters, *model* and *cpu*. It calls the base class constructor with *type* value 'Notebook', *model*, and *cpu* respectively.

Your implementation of the function will be tested by a stubbed code on several input files. Each input file contains parameters for the function calls. The functions will be called with those parameters, and the result of their executions will be printed to the standard output by the stubbed code.

▼ INPUT FORMAT FOR CUSTOM TESTING

The first line contains 3 space-separated strings for building the PersonalComputer object, where first is the model of the computer and second is the CPU of the computer respectively.

The second line contains 3 space-separated strings for building the Notebook object, where first is the model of the computer and second is the CPU of the computer respectively.

▼ SAMPLE CASE 0

Sample Input For Custom Testing

```
Asus Intel_i7
MSI AMD_Ryzen_3_3200G
```

Sample Output

```
PersonalComputer info: type - PersonalComputer, model - Asus, CPU - Intel_i7
PersonalComputer is turned off
Switching
PersonalComputer is turned on
Switching
PersonalComputer is turned off
Notebook info: type - Notebook, model - MSI, CPU - AMD_Ryzen_3_3200G
Notebook is turned off
Switching
Notebook is turned on
Switching
Notebook is turned off
```

Explanation

First, a computer object is created with *type* "PersonalComputer", *model* "Asus", *cpu* "Intel_i7", and *isTurnedOn* false. Then, all 4 functions are called: *GetComputerType*, *GetComputerModel*, *GetComputerCpu*, and *GetComputerStatus*. The result is printed to the standard output.

Then, the function *SwitchComputerStatus* is called, followed by *GetComputerStatus*, and the result is printed to the standard output. Finally, the function *SwitchComputerStatus* is called again, followed by *GetComputerStatus*, and the result is printed to the standard output.

The same operations are performed for the second computer object.

▼ SAMPLE CASE 1

Sample Input For Custom Testing

```
Lenovo Intel_Core_i3_9100F
Acer AMD_Ryzen_3_3200G
```

Sample Output

```
PersonalComputer info: type - PersonalComputer, model - Lenovo, CPU - Intel_Core_i3_9100F
PersonalComputer is turned off
Switching
PersonalComputer is turned on
Switching
PersonalComputer is turned off
Notebook info: type - Notebook, model - Acer, CPU - AMD_Ryzen_3_3200G
Notebook is turned off
Switching
Notebook is turned on
```

```
Switching
Notebook is turned off
```

Explanation

First, a computer object is created with *type* "PersonalComputer", *model* "Lenovo", *cpu* "Intel_Core_i3_9100F", and *isTurnedOn* false. Then, all 4 functions are called: *GetComputerType*, *GetComputerModel*, *GetComputerCpu*, and *GetComputerStatus*. The result is printed to the standard output.

Then, the function *SwitchComputerStatus* is called, followed by *GetComputerStatus*, and the result is printed to the standard output. Finally, the function *SwitchComputerStatus* is called again, followed by *GetComputerStatus*, and the result is printed to the standard output.

The same operations are performed for the second computer object.

Interviewer guidelines

▼ SOLUTION

C# Solution

```
/*
    abstract class Computer {
         protected string type;
         protected string model;
         protected string cpu;
         protected bool isTurnedOn;
         public Computer(string _type,string _model,string _cpu){
             type = _type;
             model = _model;
             cpu = _cpu;
         }
          public string GetComputerType(){
              return type;
          }
          public string GetComputerModel(){
             return model;
         }
          public string GetComputerCpu(){
              return cpu;
          }
          public bool GetComputerStatus(){
```

```
        return isTurnedOn;
    }
     public void SwitchComputerStatus(){
         isTurnedOn = !isTurnedOn;
     }

}
class PersonalComputer : Computer {
    public PersonalComputer(string _model,string _cpu): base("PersonalComputer",_model,_cpu)
    {}
}
class Notebook: Computer {
    public Notebook(string _model,string _cpu): base("Notebook",_model,_cpu)
    {}
}
```

Candidate's Solution Language used: **C#**

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;

namespace Solution
{
using System;

namespace Solution
{
    public abstract class Computer // Tornar a classe Computer abstrata
    {
        private string Type;
        public string Model;
        public string Cpu;
        public bool IsTurnedOn;

        // Construtor da classe base
        public Computer(string type, string model, string cpu)
        {
            Type = type;
            Model = model;
            Cpu = cpu;
            IsTurnedOn = false; // Inicializa com um valor padrão
        }

        public string GetComputerType()
        {
            return this.Type;
        }

        public string GetComputerModel()
        {
            return this.Model;
        }

        public string GetComputerCpu()
        {
            return this.Cpu;
        }

        public bool GetComputerStatus()
        {
            return this.IsTurnedOn;
        }

        // Método para alternar o estado do computador
        public void SwitchComputerStatus()
        {
            IsTurnedOn = !IsTurnedOn;
        }
    }

    // Classe PersonalComputer que herda de Computer
    public class PersonalComputer : Computer
    {
        // Construtor da classe PersonalComputer
        public PersonalComputer(string model, string cpu)
            : base("PersonalComputer", model, cpu)
        {
        }
    }

    // Classe Notebook que herda de Computer
    public class Notebook : Computer
    {
        // Construtor da classe Notebook
        public Notebook(string model, string cpu)
```

```
            : base("Notebook", model, cpu)
        {
        }
    }

    class Solution
    {
        static void Main()
        {
            Type baseType = typeof(Computer);
            if (!baseType.IsAbstract)
                throw new Exception($"{baseType.Name} type should be abstract");

            string str = Console.ReadLine();
            string[] strArr = str.Split(' ');
            Computer personalComputer = new PersonalComputer(strArr[0], strArr[1]);

            var computerType = personalComputer.GetComputerType();
            var computerModel = personalComputer.GetComputerModel();
            var computerCPU = personalComputer.GetComputerCpu();
            var computerStatus = personalComputer.GetComputerStatus() ? "on" : "off";

            Console.WriteLine($"PersonalComputer info: type - {computerType}, model - {computerModel}, CPU - {computerCPU}");
            Console.WriteLine($"PersonalComputer is turned {computerStatus}");

            Console.WriteLine("Switching");
            personalComputer.SwitchComputerStatus();
            computerStatus = personalComputer.GetComputerStatus() ? "on" : "off";
            Console.WriteLine($"PersonalComputer is turned {computerStatus}");

            Console.WriteLine("Switching");
            personalComputer.SwitchComputerStatus();
            computerStatus = personalComputer.GetComputerStatus() ? "on" : "off";
            Console.WriteLine($"PersonalComputer is turned {computerStatus}");

            str = Console.ReadLine();
            strArr = str.Split(' ');
            Computer notebook = new Notebook(strArr[0], strArr[1]);

            computerType = notebook.GetComputerType();
            computerModel = notebook.GetComputerModel();
            computerCPU = notebook.GetComputerCpu();
            computerStatus = notebook.GetComputerStatus() ? "on" : "off";

            Console.WriteLine($"Notebook info: type - {computerType}, model - {computerModel}, CPU - {computerCPU}");
            Console.WriteLine($"Notebook is turned {computerStatus}");

            Console.WriteLine("Switching");
            notebook.SwitchComputerStatus();
            computerStatus = notebook.GetComputerStatus() ? "on" : "off";
            Console.WriteLine($"Notebook is turned {computerStatus}");

            Console.WriteLine("Switching");
            notebook.SwitchComputerStatus();
            computerStatus = notebook.GetComputerStatus() ? "on" : "off";
            Console.WriteLine($"Notebook is turned {computerStatus}");
        }
    }
}

    class Solution
    {
        static void Main()
        {
            Type baseType = typeof(Computer);
            if (!baseType.IsAbstract)
                throw new Exception($"{baseType.Name} type should be abstract");

            string str = Console.ReadLine();
            string[] strArr = str.Split(' ');
            Computer personalComputer = new PersonalComputer(strArr[0], strArr[1]);

            var computerType = personalComputer.GetComputerType();
            var computerModel = personalComputer.GetComputerModel();
            var computerCPU = personalComputer.GetComputerCpu();
            var computerStatus = personalComputer.GetComputerStatus() ? "on": "off";

            Console.WriteLine($"PersonalComputer info: type - {computerType}, model - {computerModel}, CPU - {computerCPU}");
            Console.WriteLine($"PersonalComputer is turned {computerStatus}");

            Console.WriteLine("Switching");
            personalComputer.SwitchComputerStatus();
            computerStatus = personalComputer.GetComputerStatus() ? "on": "off";
            Console.WriteLine($"PersonalComputer is turned {computerStatus}");

            Console.WriteLine("Switching");
            personalComputer.SwitchComputerStatus();
            computerStatus = personalComputer.GetComputerStatus() ? "on": "off";
            Console.WriteLine($"PersonalComputer is turned {computerStatus}");

            str = Console.ReadLine();
            strArr = str.Split(' ');
            Computer notebook = new Notebook(strArr[0], strArr[1]);

            computerType = notebook.GetComputerType();
            computerModel = notebook.GetComputerModel();
            computerCPU = notebook.GetComputerCpu();
            computerStatus = notebook.GetComputerStatus() ? "on": "off";

            Console.WriteLine($"Notebook info: type - {computerType}, model - {computerModel}, CPU - {computerCPU}");
            Console.WriteLine($"Notebook is turned {computerStatus}");

            Console.WriteLine("Switching");
            notebook.SwitchComputerStatus();
            computerStatus = notebook.GetComputerStatus() ? "on": "off";
            Console.WriteLine($"Notebook is turned {computerStatus}");

            Console.WriteLine("Switching");
            notebook.SwitchComputerStatus();
            computerStatus = notebook.GetComputerStatus() ? "on": "off";
            Console.WriteLine($"Notebook is turned {computerStatus}");
        }
    }
}

```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 0	Easy	Sample	Success	1	0.0429 sec	21.1 KB
Testcase 1	Easy	Sample	Success	1	0.0413 sec	21.1 KB
Testcase 2	Easy	Hidden	Success	7	0.074 sec	21.1 KB
Testcase 3	Easy	Hidden	Success	7	0.0461 sec	21.1 KB
Testcase 4	Easy	Hidden	Success	7	0.0429 sec	21.1 KB
Testcase 5	Easy	Hidden	Success	7	0.0411 sec	21.1 KB
Testcase 6	Easy	Hidden	Success	7	0.0464 sec	21.1 KB
Testcase 7	Easy	Hidden	Success	7	0.0441 sec	21.2 KB
Testcase 8	Easy	Hidden	Success	6	0.0594 sec	21 KB

No comments.