CIB UK/PT CSharp Developer Test

The test has 3 questions:

1
Type - Coding
Languages Allowed: C#, Java 7, Java 8, JavaScript (Node.js), PHP, Python 2, Python 3, Ruby

2
Type - Coding
Languages Allowed: C++, C++14, C#, Java 7, Java 8

3
Type - Subjective

Instructions

1. This is a programming test
2. You are free to choose any language from the list and code

In this challenge, you will write an HTTP GET method to retrieve information from a movie database. You will be given a search term, and your function must return the value of the total field in the returned JSON object.

Question # 1

Function Description

Complete the function getNumberOfMovies in the editor below. The function must return the value of the total field in the returned JSON object.

getNumberOfMovies has the following parameter(s):

substr: the string to search for in the movie database

It must query https://jsonmock.SGT.com/api/movies/search/?Title=[substr] and return the total number of movie titles having the substring substr in their title. The query response is a JSON object with the following five fields:

- page: The current page.
- per_page: The maximum number of results per page.
- total: The total number of movies having the substring substr in their title.
- total_pages: The total number of pages which must be queried to get all the results.
- data: An array of JSON objects containing movie information where the Title field denotes the title of the movie.

Constraints

0 < |substr| < 20

Input Format for Custom Testing

Input from stdin will be processed as follows and passed to the function.
The only line contains the string *substr*.

Sample Case 0

Sample Input 0

```
maze
```

Sample Output 0

```
97
```

Explanation 0

The value of *substr* is *maze*, so our query is
https://jsonmock.SGT.com/api/movies/search/?Title=maze and the response is:


```
{
   page:1,
   per_page:10,
   total:97,
   total_pages:10,
   data:[
      {
         Poster:"https://images-na.ssl-images-
amazon.com/images/M/MV5BMjUyNTA3MTAyM15BMl5BanBnXkFtZTgwOTEyMTkyMjE@._V1
_SX300.jpg",
         Title:"The Maze Runner",
         Type:"movie",
         Year:2014,
         imdbID:"tt1790864"
      },
      {
         Poster:"https://images-na.ssl-images-
amazon.com/images/M/MV5BMjE3MDU2NzQyMl5BMl5BanBnXkFtZTgwMzQxMDQ3NTE@._V1
_SX300.jpg",
         Title:"Maze Runner: The Scorch Trials",
         Type:"movie",
         Year:2015,
         imdbID:"tt4046784"
```

```
      },
      {
         Poster:"https://images-na.ssl-images-
amazon.com/images/M/MV5BMjExOTkxMTIzN15BMl5BanBnXkFtZTgwNjcxNzY2NTE@._V1
_SX300.jpg",
         Title:"Into the Grizzly Maze",
         Type:"movie",
         Year:2015,
         imdbID:"tt1694021"
      },
      {
         Poster:"https://images-na.ssl-images-
amazon.com/images/M/MV5BMTIwNDg4MjIyMV5BMl5BanBnXkFtZTcwNDEwMzkxMQ@@._V1
_SX300.jpg",
         Title:"Hercules in the Maze of the Minotaur",
         Type:"movie",
         Year:1994,
         imdbID:"tt0110018"
      }
   ]
}
```

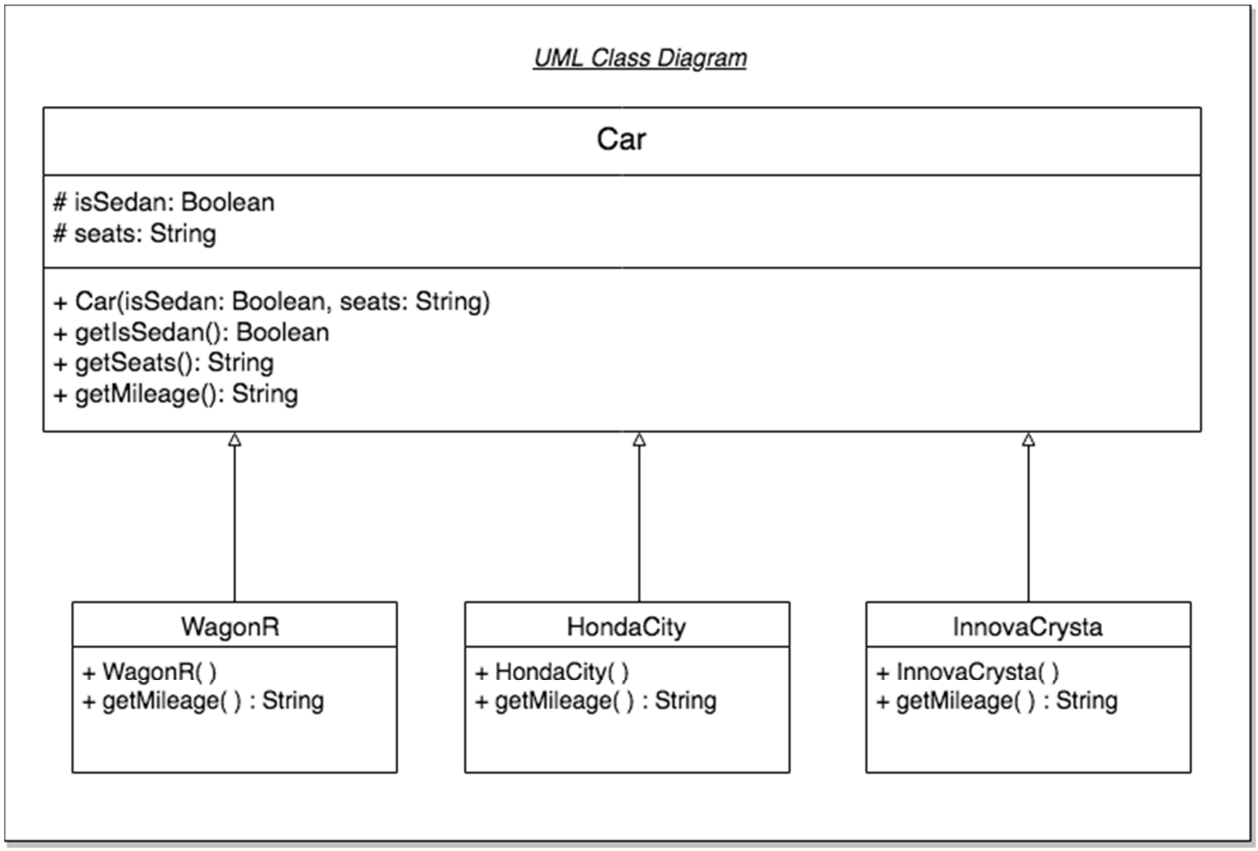

We then return the value of the *total* field as our answer, *97*.

Question #2

In this challenge, you will be asked to build on an abstract class and initialize an instance of each class with a variable. The program will then test your implementation by retrieving the data you stored.

The locked code in the editor does the following:

1. Declares an abstract class named Car with the implementations for getIsSedan() and getSeats() methods, as well as an abstract method named getMileage().
2. Creates *WagonR*, *HondaCity*, or *InnovaCrysta* object based on input (0 for WagonR, 1 for HondaCity and 2 for InnovaCrysta).
3. Calls the getIsSedan(), getSeats(), and getMileage() methods on the object.



The details for each car are provided below -

1. WagonR is not a sedan and has 4 seats.
2. HondaCity is a sedan and has 4 seats.
3. InnovaCrysta is not a sedan and has 6 seats.

Complete the code in the editor below to implement the following:

1. Three classes named, WagonR, HondaCity, and InnovaCrysta that inherit from the Car class.
2. One integer argument is provided to the constructor which is the mileage of the car.
3. Each class must implement the getMileage() method which returns a string in the form of <mileage> kmpl where <mileage> is the value provided to constructor.

Constraints

- The integer in first line will be in between 0 and 2 inclusive.
- The integer in the second line (i.e. mileage) will be in between 5 and 30 inclusive.

Input Format For Custom Testing

The first line contains an integer describing the type of car to instantiate.
The second line contains an integer, the mileage of the car.

Sample Case 0

Sample Input For Custom Testing

```
0
22
```

Sample Output

```
A WagonR is not Sedan, is 4-seater, and has a mileage of around 22 kmpl.
```

Sample Case 1

Sample Input For Custom Testing

```
1
12
```

Sample Output

```
A HondaCity is Sedan, is 4-seater, and has a mileage of around 12 kmpl.
```

Question #3

Consider the following C# code below, please identify and correct all the possible bugs and bad programming approaches in the code.

Please rewrite the code, you can change everything in the code, as long the functionality to Add and Delete people is preserved and the existing Person properties are, at least, maintained.

Warning: We recommend that you try to run the code in Visual Studio or another IDE. Include also the use cases that you have used to validate your code.

Tip: there are at least 15 necessary changes.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace TestCSharp
{
   public class Person
   {
      public string Name { get; set; }
      public int? Age { get; set; }

      public void CalcAge(DateTime _birthDate)
      {
         this.Age = _birthDate.Year - DateTime.Now.Year;
      }
   }

   public class PersonManager
   {
      public static List<Person> DB { get; set; }

      public void AddPerson(string name, string birthDate)
      {
         try
         {
            Person p = null;
            p.Name = name;
            p.CalcAge(birthDate);
            DB.Add(p);
         }
         catch (Exception e)
         {
            throw e;
         }
      }

      public void DeletePerson(int id)
      {
         try
         {
            //Write code to delete person from list
         }
         catch (Exception e) { }
      }
   }
}
```