75

khalid mahmud

Back-End Developer

PDF generated at: 20 May 2025 09:43:56 UTC

View this report on HackerRank

Score

75% • 112 / 150

scored in Backend Engineer Technical Test - Talvette in 58 min 45 sec on 20 May 2025 14:02:50 +06

Candidate Information

Email	mdkhalidmahmud1010@gmail.com
Test	Backend Engineer Technical Test - Talvette
Candidate Packet	View
Taken on	20 May 2025 14:02:50 +06
Time taken	58 min 45 sec/ 60 min
Work Experience	4 years
Invited by	Leonika
Invited on	20 May 2025 14:01:19 +06

Skill Distribution

No.	Skill	Score
1	Problem Solving Basic	0%

2	C# Basic	95%
3	.NET Basic	0%
4	.NET Intermediate	100%
5	REST API Intermediate	50%
6	SQL Basic	0%
7	SQL Intermediate	0%
8	Java Basic	100%
9	C++ Basic	100%

Tags Distribution

Data Structures	0%	Easy	80%
C#	95%	.NET	33%
Language Proficiency	0%	Problem Solving	0%
Medium	37%	.NET Core	100%
Garbage Collection	100%	Performance Optimization	100%

API	50%	MySQL	0%
SQL	0%	Java	100%
C++	100%	Overloading	100%
OOPS	100%	Polymorphism	100%
Abstraction	100%		

Questions

MCQ • 12.5 / 50

Status	No.	Question	Time Taken	Skill	Score	Code Quality
⊗	1	Prefix - 2 Multiple Choice	1 min 27 sec	Problem Solving (Basic)	0/5	-
⊗	2	Reading from Console Multiple Choice	2 min 45 sec	C# (Basic)	0/5	-
⊗	3	Modify Font Size When Printed Multiple Choice	1 min 2 sec	.NET (Basic)	0/5	-
✓	4	GC Optimization in .NET Multiple Choice	1 min 15 sec	.NET (Intermediate)	5/5	-

✓	5	Which of the following supports output caching in .NET 8 for improved performance? Multiple Choice	1 min 5 sec	-	5/5	-
✓	6	Modern Web APIs Multiple Choice	43 sec	REST API (Intermediate)	2.5/5	-
✗	7	MySQL: Space In From Clause Multiple Choice	1 min 8 sec	SQL (Basic)	0/5	-
✗	8	SQL Order of Operations Multiple Choice	25 sec	SQL (Basic)	0/5	-
✗	9	MySQL: Pipe In Select Clause Multiple Choice	11 sec	SQL (Intermediate)	0/5	-
✗	10	MySQL: Group Field Multiple Choice	1 min 34 sec	SQL (Intermediate)	0/5	-

Coding Challenge • 100 / 100

Status	No.	Question	Time Taken	Skill	Score	Code Quality
✓	11	How Will You Compare? Coding	6 min 16 sec	Java (Basic) +2	50/50	-

✓	12	Car Inheritance Coding	40 min 25 sec	Java (Basic)	+2	50/50	-

1. Prefix - 2

✗ Incorrect

Multiple Choice | Data Structures | Easy

Question description

Evaluate the prefix expression : * + 2 - 2 1 / - 4 2 + - 5 3 1

Note that the integers are all less than 10. For example, 5 3 1 is three separate integers.

Interviewer guidelines

(2+2-1)*(4-2)/(5-3+1)= 2

Candidate's Solution

Options: (Expected answer indicated with a tick)

- ○ 2 ✓
- ● 10
- ○ 12
- ○ 4

No comments.

2. Reading from Console

Incorrect

Multiple Choice | Easy | C# | .NET | Language Proficiency | Problem Solving

Question description

Consider the following code snippet:

```
using System;

namespace Solution
{
   class Solution
   {
      static void Main(string[] args)
      {
         Console.WriteLine(Console.Read());
         Console.ReadKey();
         Console.WriteLine(Console.ReadLine());
      }
   }
}
```

There is a single line input provide in the console, `hello`. The output of the code is:

Candidate's Solution

Options: (Expected answer indicated with a tick)

- h ello
- h llo

- ○ 104 ello ✓
- ○ 104 llo
- ● Compilation Error

No comments.

3. Modify Font Size When Printed

Incorrect

Multiple Choice | .NET | Easy

Question description

You have a webpage with size 15 fonts. But when someone prints the page, you want it to be printed in size 10 fonts. How can you do it?

Candidate's Solution

Options: (Expected answer indicated with a tick)

- ○ Using @media Rule ✓
- ○ Using @Print Rule
- ● Only way to do it is by creating a different print-friendly page and print that page.

Using <print> tag.

No comments.

4. GC Optimization in .NET

Correct

Multiple Choice | Medium | .NET Core | Garbage Collection | Performance Optimization

Question description

Considering the different ways .NET Core and .NET Framework handle garbage collection, which method is true for .NET Core when optimizing for high-throughput applications?

Interviewer guidelines

This question is considered medium difficulty as it requires understanding of both .NET Core and .NET Framework's garbage collection mechanisms, a grasp on how these mechanisms can be adjusted for application optimization, and the ability to recognize the impact of these adjustments on application performance.

Candidate's Solution

Options: (Expected answer indicated with a tick)

.NET Core uses a workstation garbage collection mode by default, which is optimized for desktop applications.

.NET Core allows for server garbage collection to be configured, which is better suited for high-throughput applications, utilizing multiple cores and segments for garbage collection.

- ○ Both .NET Core and .NET Framework use the same garbage collection mode and do not offer optimizations for high-throughput applications.
- ○ .NET Core exclusively uses concurrent garbage collection, which cannot be configured to optimize for different application types.

ⓘ No comments.

---

5. Which of the following supports output caching in .NET 8 for improved performance?

✓ Correct

Multiple Choice .NET

Question description

Caching

Interviewer guidelines

Which of the following supports output caching in .NET 8 for improved performance?
A) Response.WriteAsync()
B) app.UseStaticFiles()
C) OutputCacheAttribute
D) CacheMiddleware()

Candidate's Solution

Options: (Expected answer indicated with a tick)

- ○ Response.WriteAsync()

- app.UseStaticFiles()
- **OutputCacheAttribute** ✓
- CacheMiddleware()

No comments.

6. Modern Web APIs

Partially correct

Multiple Choice | Medium | API

Question description

Modern Web APIs provide many possibilities to developers. Which of the following are true?

Candidate's Solution

Options: (Expected answer indicated with a tick)

- [x] **Local storage can be used to store data across browser sessions without expiration.** ✓
- [ ] localStorage supports only string and integer types.
- [x] **FileReader allows reading files stored on the user's machine in a synchronous manner.**

- [ ] Both the Fullscreen API and the Geolocation API require explicit user permission.

No comments.

7. MySQL: Space In From Clause

Incorrect

Multiple Choice | MySQL | Easy

Question description

Choose the correct explanation of the next query:
Select all that apply.

```
SELECT * FROM users clients
```

Candidate's Solution

Options: (Expected answer indicated with a tick)

- [ ] Selects everything in the “users” table, then adds columns from the “clients” table.
- [x] **If the “users” table exists, selects everything from there. Otherwise, queries the “clients” table as a fallback.**
- [ ] Selects everything in the “users” table, then assigns the “clients” alias.

Selects everything in the “users” table, then adds rows from the “clients” table.

No comments.

8. SQL Order of Operations

Incorrect

Multiple Choice SQL Easy

Question description

Which is the correct order of execution of operations in SQL?

Candidate's Solution

Options: (Expected answer indicated with a tick)

SELECT, FROM, JOIN, WHERE, GROUP BY, HAVING, ORDER BY

SELECT, FROM, GROUP BY, WHERE, HAVING, ORDER BY

FROM, GROUP BY, WHERE, ORDER BY, HAVING, SELECT

SELECT, FROM, GROUP BY, HAVING, WHERE, ORDER BY

No comments.

9. MySQL: Pipe In Select Clause

Incorrect

Multiple Choice | MySQL | Medium

Question description

Select the expression that causes a MySQL error.

Candidate's Solution

Options: (Expected answer indicated with a tick)

- ○ SELECT level | depth FROM Categories
- ● **SELECT level || depth FROM Categories**
- ○ SELECT level ||| depth FROM Categories ✓
- ○ SELECT level |'|'| depth FROM Categories

No comments.

10. MySQL: Group Field

Incorrect

Multiple Choice | MySQL | Medium

Question description

Select the expression that causes a MySQL error.

Select all that apply.

Candidate's Solution

Options: (Expected answer indicated with a tick)

- ○ SELECT customer_id, ANY_VALUE(amount) FROM transactions GROUP BY customer_id
- ○ SELECT customer_id, RAND(amount) FROM transactions GROUP BY customer_id ✓
- ● **SELECT customer_id, SUBSTRING_INDEX(GROUP_CONCAT(amount ORDER BY RAND()), ',', 1) FROM transactions GROUP BY customer_id**
- ○ none of the above, all expressions are correct

No comments.

11. How Will You Compare?

Correct

Coding | Java | C++ | C# | Overloading | OOPS | Easy

Question description

Create a Comparator class that includes three overloaded compare methods:

- *boolean compare(int a, int b):* Return true if *a* = *b*, otherwise return false.
- *boolean compare(string a, string b):* Return true if *a* = *b*, otherwise return false.
- *boolean compare(int[] a, int[] b):* Return true if both of the following conditions hold true. Otherwise, return false:

- The length of *a* = the length of *b*.
- Elements *a[i]* = *b[i]* for all indices.

Note for C++ implementations: Use Vector<int> for the array parameters.

Constraints

- For strings, 1 ≤ length of *a,* length of *b* ≤ 2000
- For integers, 0 ≤ *a, b* ≤ 10000000
- For integer arrays, 1 ≤ length of *a,* length of *b* ≤ 10

▼ INPUT FORMAT FOR CUSTOM TESTING

Input from stdin will be processed as follows and passed to the function.

The first line contains an integer *T*, the number of test cases.
Each of the next *T* sets of lines is in one of the following formats:

- The first line contains the integer 1 representing the comparison type (1, 2, or 3 for int, string, or array comparisons, respectively). The next two lines contain strings *a* and *b*.
- The first line contains the integer 2 representing the overloaded function type. The next two lines contain integers *a* and *b*.
- The first line contains the integer 3 representing the overloaded function type. The next three lines contain the following:
  1. Two space-separated integers *n* and *m*, the lengths of arrays *a* and *b*.
  2. A line of *n* space-separated integers *a[i]*.
  3. A line of *m* space-separated integers *b[i]*.

▼ SAMPLE CASE 0

Sample Input 0

```
STDIN       Function
-----       --------
3       → T = 3 number of test cases.
1       → Comparison type 1
hello world →  a = "hello world"
hello world →  b = "hello world"
2       → Comparison type 2
3       → a = 3
4       → b = 4
3       → Comparison type 3
```

```
3 3       → a[] size n=3  b[] size m=3
1 2 3     → a = [1, 2, 3]
1 2 3     → b = [1, 2, 3]
```

Sample Output 0

```
Same
Different
Same
```

Explanation 0

There are *3* test cases:

Test Case	comparison type	a	b	Output	Explanation
1	*1*	*"hello world"*	*"hello world"*	*"Same"*	Both strings are the same.
2	*2*	*3*	*4*	"*Different*"	The two integers are different (*3 ≠ 4*).
3	*3*	*{1,2,3}*	*{1,2,3}*	*"Same"*	Both arrays have the same number of elements and each element *a[i] = b[i]*

▼ SAMPLE CASE 1

Sample Input 1

```
STDIN     Function
-----     --------
2         → T = 2 number of test cases.
3         → Comparison type 3
3 4       → a[] size=3 b[] size=4
1 2 3     → a = [1, 2, 3]
1 2 3 4   → b = [1, 2, 3, 4]
1         → Comparison type 1
```

```
HackerRank → a = "HackerRank"
HackerRank → b = "hackerRank"
```

Sample Output 1

```
Different
Different
```

Explanation 1

There are *2* test cases.

Test Case	comparison type	a	b	Output	Explanation
1	3	*{1, 2, 3}*	*{1, 2, 3, 4}*	"Different"	The arrays are different.
2	*1*	*HackerRank*	*hackerRank*	"*Different*"	The two strings are different.

Interviewer guidelines

▼ SOLUTION

```
/*Write your code here. DO NOT use access modifiers (e.g.: 'public') in your class declarations.*/
class Comparator {
    boolean compare(int a, int b) {
        if (a == b)
            return true;
        return false;
    }

    boolean compare(String a, String b) {
        if (a.equals(b)) {
            return true;
        }
        return false;
    }

    boolean compare(int[] a, int[] b) {
```

```
        if (a.length != b.length)
            return false;
        for (int i = 0; i < a.length; i++) {
            if (a[i] != b[i]) {
                return false;
            }
        }
        return true;
    }
}
```

Candidate's Solution

Language used: **C#**

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
/*Enter your solution*/

class Comparator{
    public  bool compare(int a,int b){
        if(a==b) return true;
        return false;
    }


    public  bool compare(string a,string b){
        return a.Equals(b);
    }

    public  bool compare(int[] a, int[] b){
        if(a.Length!=b.Length) return false;

        for(int i=0;i<a.Length;i++){
            if(a[i]!=b[i]) return false;
        }

        return true;
    }
}

class Solution {
    static void Main(String[] args) {
        Comparator comp = new Comparator();
        int testCases = Convert.ToInt32(Console.ReadLine());
```

```
33          while(testCases-- > 0){
34              int condition = Convert.ToInt32(Console.ReadLine());
35              if(condition == 1){
36                  string s1=Console.ReadLine();
37                  string s2=Console.ReadLine();
38                  if(comp.compare(s1,s2)){
39                      Console.WriteLine("Same");
40                  }
41                  else{
42                      Console.WriteLine("Different");
43                  }
44              }
45              else if(condition == 2){
46                  int num1=Convert.ToInt32(Console.ReadLine());
47                  int num2=Convert.ToInt32(Console.ReadLine());
48                  if(comp.compare(num1,num2)){
49                      Console.WriteLine("Same");
50                  }
51                  else{
52                      Console.WriteLine("Different");
53                  }
54              }
55              else if(condition == 3){
56                  Console.ReadLine();
57                  int[] arr1 = Console.ReadLine().Split(' ').Select( x=>
    Convert.ToInt32(x)).ToArray();
58                  int[] arr2 = Console.ReadLine().Split(' ').Select( x=>
    Convert.ToInt32(x)).ToArray();
59                  if(comp.compare(arr1,arr2)){
60                      Console.WriteLine("Same");
61                  }
62                  else{
63                      Console.WriteLine("Different");
64                  }
65              }
66          }
67      }
68  }
69
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED

TestCase 0	Easy	Sample	Success	1	0.0617 sec	23.7 KB
TestCase 1	Easy	Sample	Success	1	0.0436 sec	23.8 KB
TestCase 2	Easy	Sample	Success	9	0.0491 sec	23.6 KB
TestCase 3	Easy	Sample	Success	9	0.0414 sec	23.7 KB
TestCase 4	Easy	Hidden	Success	10	0.046 sec	23.7 KB
TestCase 5	Easy	Hidden	Success	10	0.04 sec	23.7 KB
TestCase 6	Easy	Hidden	Success	10	0.0471 sec	23.7 KB

No comments.

12. Car Inheritance

Correct

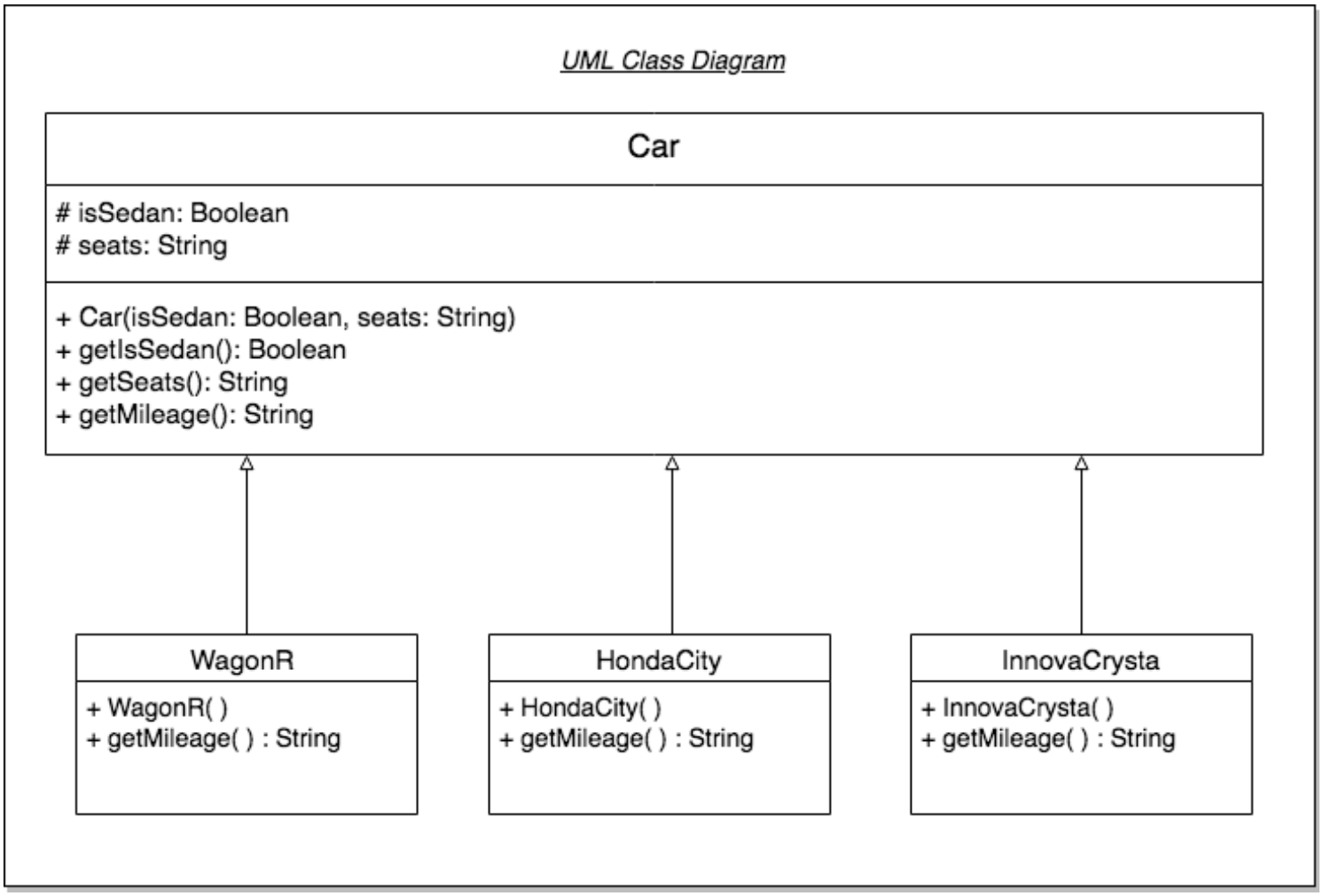
Coding | Polymorphism | Easy | Abstraction | Java | C++ | C#

Question description

Build on an abstract class and create instances of each derived class with specific variables. The program will verify the implementation by accessing the stored data.

The provided code performs the following tasks:

- Defines an abstract class called *Car* with implementations for the methods *getIsSedan()* and *getSeats()*, and an abstract method named *getMileage()*
- Instantiates a *WagonR, HondaCity*, or *InnovaCrysta* object based on input (0 for WagonR, 1 for HondaCity, and 2 for InnovaCrysta)
- Invokes the *getIsSedan()*, *getSeats()*, and *getMileage()* methods on the object



Details for each car are as follows:

- WagonR is not a sedan and has 4 seats.
- HondaCity is a sedan and has 4 seats.
- InnovaCrysta is not a sedan and has 6 seats.

Function Description

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

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

Constraints

- 0 ≤ type of car ≤ 2
- 5 ≤ *mileage* ≤ 30

▼ INPUT FORMAT FOR CUSTOM TESTING

The first line contains an integer that describes 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

```
STDIN   Function
-----   --------
0    →  type of car to instantiate = 0 (WagonR)
22   →  mileage = 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

```
STDIN   Function
-----   --------
1    →  type of car to instantiate = 1 (HondaCity)
12   →  mileage = 12
```

Sample Output

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

Interviewer guidelines

▼ SOLUTION

Java 17 Solution

```
// Write your code here.

/**
 * WagonR class
 */
class WagonR extends Car {
    private int mileage;

    public WagonR(int mileage) {
        super(false, "4");
        this.mileage = mileage;
    }

    @Override
    public String getMileage() {
        return mileage + " kmpl";
    }
}

/**
 * HondaCity class
 */
class HondaCity extends Car {
    private int mileage;

    public HondaCity(int mileage) {
        super(true, "4");
        this.mileage = mileage;
    }

    @Override
    public String getMileage() {
        return mileage + " kmpl";
    }
}

/**
 * InnovaCrysta class
 */
class InnovaCrysta extends Car {
```

```
    private int mileage;

    public InnovaCrysta(int mileage) {
        super(false, "6");
        this.mileage = mileage;
    }

    @Override
    public String getMileage() {
        return mileage + " kmpl";
    }
}
```

Candidate's Solution

Language used: **Java 21**

```
1  import java.util.Scanner;
2  import java.io.*;
3
4
5  abstract class Car {
6      protected boolean isSedan;
7      protected String seats;
8
9      public Car(boolean isSedan, String seats) {
10         this.isSedan = isSedan;
11         this.seats = seats;
12     }
13
14     public boolean getIsSedan() {
15         return this.isSedan;
16     }
17
18     public String getSeats() {
19         return this.seats;
20     }
21
22     abstract public String getMileage();
23
24     public void printCar(String name) {
25         System.out.println(
26           "A " + name + " is " + (this.getIsSedan() ? "" : "not ")
27             + "Sedan, is " + this.getSeats() + "-seater, and has a mileage
   of around "
28             + this.getMileage() + ".");
29     }
```

```
}
// Write your code h
// Write your code here.



/**
*   WagonR class
**/

 class WagonR extends Car{
    private int mileage;


    public WagonR(int mileage){
         super(false,"4");
        this.mileage = mileage;
    }

    @Override
    public String getMileage(){
     return this.mileage+" kmpl";
    }
}


/**
*   HondaCity class
**/

 class HondaCity extends Car{
    private int mileage;


     public HondaCity(int mileage){
         super(true,"4");
         this.mileage = mileage;
    }

    @Override
    public String getMileage(){
        return this.mileage+" kmpl";
    }
}

/**
```

```
76 *   InnovaCrysta class
77 **/
78
79  class InnovaCrysta extends Car{
80     private int mileage;
81
82
83      public InnovaCrysta(int mileage){
84          super(false,"6");
85          this.mileage = mileage;
86     }
87
88     @Override
89     public  String getMileage(){
90         return this.mileage+" kmpl";
91     }
92 }
93
94
95 public class Solution {
96
97     public static void main(String[] args) throws IOException {
98         BufferedReader bufferedReader = new BufferedReader(new
   InputStreamReader(System.in));
99         int carType = Integer.parseInt(bufferedReader.readLine().trim());
100        int carMileage = Integer.parseInt(bufferedReader.readLine().trim());
101
102        if (carType == 0){
103            Car wagonR = new WagonR(carMileage);
104            wagonR.printCar("WagonR");
105        }
106
107        if(carType == 1){
108            Car hondaCity = new HondaCity(carMileage);
109            hondaCity.printCar("HondaCity");
110        }
111
112        if(carType == 2){
113            Car innovaCrysta = new InnovaCrysta(carMileage);
114            innovaCrysta.printCar("InnovaCrysta");
115        }
116    }
117 }
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
TestCase 0	Easy	Sample	Success	1	0.0527 sec	37.8 KB
TestCase 1	Easy	Sample	Success	1	0.0543 sec	37.9 KB
TestCase 2	Easy	Sample	Success	1	0.0817 sec	37.6 KB
TestCase 3	Easy	Hidden	Success	7	0.0529 sec	37.5 KB
TestCase 4	Easy	Hidden	Success	8	0.0608 sec	37.4 KB
TestCase 5	Easy	Hidden	Success	8	0.0554 sec	37.5 KB
TestCase 6	Easy	Hidden	Success	8	0.0577 sec	37.4 KB
TestCase 7	Easy	Hidden	Success	8	0.0553 sec	37.5 KB
TestCase 8	Easy	Hidden	Success	8	0.0513 sec	37.5 KB

No comments.